Advanced HP OpenView Network Node Manager

(Configuring Management of Topologies and Events)

U5089S C.00

HP Training

Student guide

invent

© Copyright 1983-2004 Hewlett-Packard Development Company, L.P.

The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

This is an HP copyrighted work that may not be reproduced without the written permission of HP. You may not use these materials to deliver training to any person outside of your organization without the written permission of HP.

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

HEWLETT-PACKARD COMPANY
3404 E. Harmony Road
Fort Collins, CO 80528 U.S.A.

Use of this manual and flexible disk(s), tape cartridge(s), or CD-ROM(s) supplied for this pack is restricted to this product only. Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

Contains software from AirMedia, Inc.

© Copyright 1996 AirMedia, Inc.

Trademark Notices
Java™ is a U.S. trademark of Sun Microsystems, Inc.
Microsoft®, Windows XP®, Windows® 2000, Windows®, and MS Windows® are U.S. registered trademarks of Microsoft Corporation.
Netscape™ and Netscape Navigator™ are U.S. trademarks of Netscape Communications Corporation.
Oracle® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.
Oracle7™ is a trademark of Oracle Corporation, Redwood City, California.
OSF/Motif® and Open Software Foundation® are trademarks of Open Software Foundation in the U.S. and other countries.
Pentium® is a U.S. registered trademark of Intel Corporation.
UNIX® is a registered trademark of The Open Group.

Printed in US

Advanced HP OpenView Network Node Manager
Configuring Management of Topologies and Events
Student guide
August 2004

Contents

1. Introduction

Module Objectives 1-1
Course Outline 1-2
NNM Product Structure 1-3
Product Structure 1-5
Network Management SPIs 1-7
Networks that Require Extended Topology for Discovery 1-9
What Extended Topology Discovers 1-11
Extended Topology Monitors Overlapping IP Address Domains 1-13
Extended Topology Monitors OSPF 1-14
Extended Topology Manages HSRP Groups 1-15
Extended Topology Discovers and Manages IPv6 Networks 1-16
Providing Root Cause Analysis 1-17
Difficult Diagnostics: Do It All In Your Head 1-18
Make the Computer Do the Work 1-19
Intelligent Diagnostics for Networks 1-20
Intelligent Diagnostics Components 1-21
Intelligent Diagnostics - Where Can It Lead? 1-22
Comparing Event Reduction Mechanisms 1-23
Home Base 1-25
Configuring and Monitoring NNM 1-27
Management By Exception 1-28
The Web Based Alarm Browser 1-29
The Dynamic View Window 1-31
A Walk-Through of Dynamic Views 1-33
Lab Exercises: Introduction 1-35

2. Discovering Connectivity with Extended Topology

Module Objectives 2-1
Comparing NNM and Extended Topology 2-2
Understanding Recurring Discovery 2-3
Temporarily Unresponsive Nodes 2-4
Extended Topology Support Classes 2-6
Validated Device List 2-8
Device Support Rules 2-9
Extended Topology Integration Model 2-12
Discovery Process Overview 2-14
Lab: Extended Topology Discovery 2-15

3. Enabling Extended Topology

Module Objectives 3-1
Before Enabling Extended Topology 3-2
SNMP Configuration 3-3
Starting Extended Topology Analysis 3-4
Automatic Zone Partitioning 3-7
Automatic Zone Configuration 3-8
Manually Starting Extended Topology Discovery 3-11

Contents

Incremental Discovery . . . 3-13
Configuring Discovery Cycles . . . 3-15
Viewing Discovery Status . . . 3-17
Lab: Extended Topology Discovery . . . 3-19

4. Distributing Extended Topology

Module Objectives . . . 4-1
DIM Characteristics of Extended Topology . . . 4-2
Extended Topology and Replication . . . 4-3
Views in a DIM Environment . . . 4-4
Lab Exercises . . . 4-6

5. Scaling netmon Discovery and Polling

Module Objectives . . . 5-1
Scalability of netmon Status Polling . . . 5-2
Controlling the Set of Managed Interfaces . . . 5-3
Automatically Unmanaging Interfaces . . . 5-5
Configuring Interface Unmanagement . . . 5-7
Filter Examples . . . 5-9
Handling DNS Problems . . . 5-11
Testing DNS Resolution . . . 5-12
Restricting Forward Lookups . . . 5-14
Restricting NNM's Reverse Name Lookups . . . 5-16
Lab Exercises . . . 5-18

6. Controlling Extended Topology Discovery

Module Objectives . . . 6-1
Limiting Extended Topology Discovery . . . 6-2
bridge.noDiscover Filtering . . . 6-4
Zone Discovery . . . 6-5
Configuring Zones . . . 6-7
Zone Configuration Test . . . 6-10
Zone Example with Routed Core . . . 6-11
Extended Topology Support Tools . . . 6-13
Discovery Stages . . . 6-14
dumpDiscoStatus.ovpl . . . 6-16
dumpAgentProgress.ovpl . . . 6-18
Viewing Extended Topology Data . . . 6-20
Deployment Tips: Pre-Discovery . . . 6-22
Deployment Tips: During Discovery . . . 6-25
Deployment Tips: Post-Discovery . . . 6-27
Database Support for Extended Topology Data . . . 6-29
XPL Logging . . . 6-30
Lab Exercises . . . 6-31

7. Managing Overlapping IP Address Domains

Contents

Module Objectives 7-1
Why Use Private IP Addresses? 7-3
What Is Network Address Translation (NAT)? 7-4
How Does NAT Work? 7-5
Receiving a NAT'd Response 7-6
Types of NAT Configuration 7-7
NNM Management Features 7-9
How Can NNM Manage Through NAT? 7-10
Private IP Address Management Example 7-11
NNM Shows Overlapping Addresses 7-13
Overlapping IP Address Terminology 7-14
Overlapping IP Support Limitations 7-16
Traffic Flows To/From Customer Networks 7-17
Overlapping Address Domains 7-18
Configuration Overview 7-19
Configure Overlapping Address Domains 7-21
dupip.conf Commands and Attributes 7-23
Create a Seedfile For Each OAD 7-25
Check Syntax Using ovdupip 7-27
Load Changes Into Extended Topology 7-30
Deploy Changes to Running Configuration 7-31
Rediscover a Single Zone 7-32
Deleting an OAD 7-33
OAD View 7-34
OAD View Popup Selections 7-36
Viewing OAD Node Details 7-37
Neighbor View in an OAD Environment 7-38
VLAN View with OADs 7-39
HSRP View with OADs 7-40
OAD in Topology Summary 7-41
OADs in the Alarm Browser 7-42
OAD Discovery Architecture 7-43
Troubleshooting Overlapping Address Domains 7-44
Troubleshooting Overlapping Address Domains 7-45
Lab Exercises 7-46

8. Active Problem Analyzer

Module Objectives 8-1
Active Problem Analyzer 8-3
NNM AE Polling and Status Engines 8-5
Active Problem Analysis for HSRP 8-6
Status Polling Engine 8-8
Analysis Engine 8-10
APA Handles Overlapping Address Domains 8-12
Choosing Your General Poller 8-14
Returing Status to ovw 8-16
Enabling and Disabling APA 8-18
APA Demand Poll 8-19

Contents

Demand Poll Results 8-20
Configuring APA 8-22
Using Topology Filters 8-25
Controlling HSRP Polling 8-28
Polling Protocols for Network Devices 8-30
Controlling Polling Protocols 8-31
APA Event-Triggered Polling 8-34
Visualizing APA Not Monitored 8-36
Setting Objects to Not Monitored 8-38
Events and Status 8-40
Comparing NNM and APA Status Handling 8-43
Neighbor Analysis: Connected Ports 8-45
Neighbor Analysis: Unconnected Ports 8-47
Important Nodes Filter 8-49
APA Configuration Polling 8-51
What Triggers a Configuration Poll? 8-53
Interface Renumbering 8-54
APA Interface Renumbering Analysis 8-55
Interface Renumbering Display 8-56
APA Statistics 8-57
Lab Exercises 8-59

9. Configuring Extended Topology Discovery of OSPF

Module Objectives 9-1
Open Shortest Path First Background 9-2
Running OSPF Discovery 9-4
Extended Topology OSPF View 9-6
Common OSPF Errors 9-7
Lab Exercises 9-9

10. Configuring Extended Topology Discovery of HSRP

Module Objectives 10-1
HSRP Background Information 10-2
HSRP Failover 10-4
HSRP Failover 10-6
HSRP Failover 10-7
HSRP Failover 10-8
HSRP Failover 10-9
Collect and Display HSRP Router Information 10-10
HSRP View Details 10-13
HSRP View 10-14
HSRP Operation 10-15
HSRP Interaction with netmon 10-17
Validating Your Results 10-19
Lab Exercises 10-21

11. Introduction to Event Reduction

Contents

Module Objectives 11-1
Fundamental Objective: Event Reduction 11-2
The Need for Event Correlation 11-3
Correlation Example 11-5
ECS and HP OpenView 11-6
What is a Correlation? 11-8
Event Flows and Streams 11-10
Architecture of ECS in NNM 11-12
Improving on What Correlations Do 11-13
What If I Have Unique Needs? 11-14
What Are My Options? 11-16
Composer May Be Your Answer 11-17
What Does It Take? 11-18
Composer Is a Super-Correlation 11-19
Correlator Evaluation 11-20
Choosing an Event Reduction Mechanism 11-21
Available Manuals 11-23
Lab Exercises 11-24

12. Configuring Event Correlation

Module Objectives 12-1
ECS Configuration Files 12-3
The ECS Event Configuration GUI 12-4
Enabling and Disabling Correlations 12-7
Composer: Configuring NNM-Shipped Correlators 12-9
Internal Event Correlators: NodeIF 12-10
Internal Event Correlator: IntermittentStatus 12-11
Internal Event Correlator: Chassis 12-12
Multiple Reboots Correlator 12-13
NNM Supplied Correlations 12-14
Modifying Event Correlations 12-16
Modifying Parameters 12-19
Simple Data Types 12-21
netmon Accelerated Polling 12-23
Connector Down Correlation 12-26
Configuring Secondary Failures 12-28
ConnectorDown Correlation with NNM Extended Topology 12-31
PairWise Event Correlation 12-33
Pattern Delete 12-36
Repeated Events Correlation 12-38
Event De-Duplication 12-40
Scheduled Maintenance Correlation 12-42
Copying a Correlation 12-44
Configuring ECS from the Command Line 12-46
Troubleshooting ECS 12-48
Lab Exercises: Enabling ECS 12-50

13. Introduction to Composer Development

Contents

Module Objectives . . . 13-1
Correlation Composer Overview . . . 13-2
Planning the Correlation . . . 13-4
Correlator Development Process . . . 13-6
Creating Correlator Requirements . . . 13-8
Composer User Interface Modes . . . 13-9
Operator Access to Composer . . . 13-10
Starting the Composer Developer Interface . . . 13-12
Setting the Event Type . . . 13-14
Selecting a Template . . . 13-16
Correlator Store Files . . . 13-18
Opening a Correlator Store File . . . 13-20
Exclusive Access to Correlator Stores . . . 13-22
Configuring Operator Access . . . 13-24
Development and Runtime Correlator Stores . . . 13-26
Configuring a NameSpace . . . 13-27
Configuring the Security File . . . 13-29
Configuring Deployment Files . . . 13-33
Deploying Correlators . . . 13-35
Performance Implications . . . 13-37
Correlator Development Cautions . . . 13-38
Lab Exercises . . . 13-40

14. Creating a Basic Correlator

Module Objectives . . . 14-1
Suppress Correlator Template . . . 14-2
Suppress Example - Problem Statement . . . 14-3
Suppress Example - Sample PDU . . . 14-4
Creating a Correlator . . . 14-5
Select Incoming Events . . . 14-7
Alarm Signature . . . 14-9
Operators . . . 14-11
Suppress Example - Solution . . . 14-13
Suppress Example - Definition Window . . . 14-14
Suppress Example - Resulting Alarm Browser . . . 14-15
Suppress Example Update - Problem Restatement . . . 14-16
Event Contents and Varbinds . . . 14-17
Suppress Example Update - Sample PDU . . . 14-18
Suppress Example Update - Solution . . . 14-19
Suppress Example Update - Definition Window . . . 14-20
Lab Exercises . . . 14-21

15. Using Variables in Correlators

Module Objectives . . . 15-1
Variable Types . . . 15-2
Scope of Variables . . . 15-4
Advanced Filter . . . 15-5

Contents

Variable Example 15-6
Variable Example: Definition Window 15-8
Configuring a Lookup Variable 15-10
Editing the Datastore 15-11
Variable Example Update 15-12
Variable Example Update - Definition Window 15-13
Extract Patterns 15-15
Pattern Matching 15-16
Extract Variable Assignment 15-19
Extract Example 15-21
Extract Variable Definition 15-22
Extract Example 15-23
Extract Example - Definition Window 15-25
Creating a Combine Variable 15-27
Enhance Correlator Template 15-28
Enhance Example 15-29
Enhance Example - Definition Window 15-31
New Event Creation 15-33
Lab Exercises 15-35

16. Using Additional Correlators

Module Objectives 16-1
Rate Correlator Template 16-2
Message Key 16-3
Automatic Variables 16-5
Rate Example 16-6
Rate Example - Definition Window 16-7
Rate Example - New Event Creation 16-8
Repeated Correlator Template 16-9
Repeated Example 16-11
Repeated Example - Definition Window 16-13
Transient Correlator Template 16-15
Transient Example 16-16
Transient Example - Definition Window 16-18
Lab Exercises 16-20

17. Relating Events from Multiple Sources

Module Objectives 17-1
Multi-Source Correlator Template 17-2
Set Definition 17-3
MultiSource Example 17-4
MultiSource Example - BSC Definition 17-6
MultiSource Example - BTS Definition 17-8
Another MultiSource Example 17-10
Another MultiSource Example - Trap A & B Definitions 17-12
Another MultiSource Example - Trap C Definition 17-14
Another MultiSource Example - New Event 17-16

Contents

Lab Exercises 17-17

18. Using Callbacks and Built-In Functions

Module Objectives. 18-1
Variables that Access Functions 18-2
Variable and Function Evaluation. 18-4
Callback Example - Problem Statement 18-6
Passing Parameters to Functions 18-7
Callback Example - Callback Specification. 18-9
Callback Example Update - Problem Statement 18-11
Callback Example Update- Display Correlated Events 18-12
Built-In Functions 18-13
Built-in Example - Problem Statement. 18-15
Built-in Example - Design 18-16
Concept of Keys. 18-17
storeStr () Syntax 18-18
retrieveStr () Syntax. 18-19
Built-in Example - Design 18-20
Built-in Example - Definition Window 18-22
Built-in Example - New Event 18-24
GetByIndex() to Access Multiple Return Values. 18-25
getByIndex Example 18-27
Example: OV_MultipleReboots 18-29
Load Perl Script or C Library 18-30
Lab Exercises 18-32

19. Best Practices and Tools

Module Objectives. 19-1
Best Practices 19-2
Migrating a Correlator Store File 19-4
Viewing Previous Correlator Store Revisions. 19-5
Merging Correlator Store Files 19-7
Analyzing Events 19-10
Capturing an Event Stream. 19-13
Replaying an Event Stream. 19-16
Tracing Events 19-18
Function Debugging Tips. 19-20
Lab Exercises 19-22

20. Combining Correlators

Module Objectives. 20-1
Examine the OV_NodeIF Correlators 20-2
Examine the OV_NodeIF Correlators 20-3
Examine the OV_NodeIF Correlators 20-4
Examine the OV_NodeIF Correlators 20-5
Examine the OV_NodeIF Correlators 20-6
Advanced Function Example - Problem Statement 20-7

Contents

Advanced Function Example - Design 20-10
Concept of Feedback 20-11
Feedback Example 20-12
Lab Exercises 20-13

21. Configuring syslog Messages for SNMP

Module Objectives 21-1
Converting syslog Messages to SNMP Traps 21-3
syslog Deployment Models 21-5
Architecture in NNM 21-7
Default Trap Mappings 21-9
Configuration Prerequisites 21-12
Configuration Overview 21-14
NNM syslog Main GUI 21-15
Extract Patterns 21-18
Pattern Matching 21-19
Extract Variable Assignment 21-22
Extract Example 21-24
Testing Extract Patterns 21-25
Sending an SNMP Message on a Condition 21-27
Suppressing a syslog Pattern 21-28
The Syslog to NNM Template 21-29
Modifying a Condition 21-31
Add a Condition 21-33
Adding and Modifying Varbinds 21-34
Deploying syslog Mappings 21-35
Testing Syslog Monitoring 21-37
syslog Traps and Overlapping IP Addresses 21-39
Troubleshooting Tips 21-40
Removing Syslog Integration 21-43
Lab Exercises 21-45

A. Viewing Your Environment with Dynamic Views

Module Objectives A-1
Accessing Dynamic Views A-2
Using Home Base A-3
Using Alarms to Launch Dynamic Views A-4
Features and Cues of the Views A-5
A Hierarchy of Views A-6
Internet View A-8
Network View A-9
Segment View A-10
Extended Topology VLANs View A-11
Change Displayed Labels A-13
Active Tables A-14
Expand Neighbors A-15
Poster Printing A-16

Contents

Troubleshooting Dynamic Views A-17
Lab Exercises A-18

B. Securing Dynamic Views

Module Objectives B-1
User View of Dynamic View Security B-2
What is a Tomcat Realm? B-3
Using Tomcat Realms for Security B-4
Verify a Realm for Dynamic Views B-5
Add Roles B-6
Add Users B-8
Restart ovas to Read Configuration B-9
Using MD5 Password Encryption B-10
Lab Exercises B-12

C. Using Problem Diagnosis

Module Objectives C-1
Overview of Problem Diagnosis C-2
Major Components C-3
Problem Diagnosis Server C-4
Problem Diagnosis Probe C-5
Starting the User Interface C-6
Selecting Endpoints C-7
Path List C-8
Path Map C-10
Partial Paths C-11
Path Detail C-12
Current Path Detail C-14
Trek Detail C-16
Detecting Network Brownouts C-17
Lab Exercises C-18

D. Configuring Problem Diagnosis

Module Objectives D-1
Server Tasks D-2
Installing the Server D-3
Starting and Stopping the Server D-4
Linking the Server to a Probe D-5
Configuring the Server Port D-8
Configuring Brownout Detection D-10
Probe Tasks D-12
Installing a Probe D-13
Configuring a Probe D-15
Starting and Stopping the Probe D-18
Disabling a Probe D-20
Linking the Probe to a Server D-22
Troubleshooting a Probe D-25

Contents

Uninstalling Problem Diagnosis Software D-27
Lab Exercises D-28

E. Constructing Advanced Filters

Module Objectives E-1
Filter Overview E-3
Filters Streamline NNM Data Flow E-4
Using Object Attributes E-6
Looking Inside the filters File E-8
Defining Object Sets E-10
Attribute Value Assertions E-12
AVA Operators E-15
Building Filter Expressions E-17
Special Pattern Matching in NNM Filters E-19
Regular Expressions E-21
Pattern Matching Examples E-25
Testing Your Filters E-26
Example filters File E-28
Lab Exercises E-30

F. Installing and Configuring NNM on Linux

Module Objectives F-1
Installation Overview F-2
System Requirements F-3
Pre-Installation Steps F-5
Installing the NNM Software F-8
After Installing F-10
Configuring Oracle F-11
Verifying NNM F-12
Troubleshooting and Known Issues F-14
Removing NNM F-16
Lab Exercises: Installing NNM on Linux F-18

G. Device Managment Details

Module Objectives G-1
Status Determination for Switches G-2
Enable or Disable SNMP Polling for Unconnected Switch Ports G-4
Example Admin Down Interface G-6
Dynamic Handling of Unconnected Ports G-7
Managing Cisco Boards G-9
Discovering Cisco Boards G-10
Monitoring Cisco Boards G-11
Board Visualization G-13
Board Count in Topology Summary G-14
Aggregated Ports G-15
Monitoring Aggregated Ports G-17
Visualizing Cisco Aggregated Ports G-20

Contents

Aggregated Ports in ovet_topodump.ovpl G-22
Nortel MultiLink Trunk Support G-24
Switch Stack Device Features G-25
NNM Stack Support G-26
ProCurve Switch Stack Support G-27
Stacked Switch Management G-28
Sample Stack Visualization G-29
Visualizing Layer 3 Edges G-30
NNM 7.0 Handling of Layer 3 Connectivity G-31
Visualization Without Connectivity Information G-32
NNM 7.5 Addresses Layer 3 Edges G-33
Visualization When Connectivity Info is Available G-34
Configuring Layer 3 Edge Discovery G-35
Visualization Example G-36
Duplicate IP Address Support G-37
Anycast Address Support G-39
Backup Address Support G-41
Other Duplicate IP Address Support G-43
Simple Extended Topology Object Model G-45
1: Address Down, Interface Up, Node Up G-47
2: Interface Down, Node Up G-48
3: Address Down, Interface Down, Node Up G-49
4: Two Connected Interfaces Down G-50
5: Node Down G-51
Neighbor Analysis Algorithm G-52
Neighbor Analysis: One Connection G-53
Neighbor Analysis: Two Connections G-54
Neighbor Analysis: OAD NextHop G-55
Neighbor Analysis: End Node Down G-56
Lab Exercises G-57

H. IPv6 in Extended Topology

Module Objectives H-1
IPv6 Background H-2
IPv6 Features H-3
IPv6 Address Types H-4
IPv6 Addresses H-5
Abbreviating IPv6 Addresses H-6
IPv6 Address Scoping H-7
IPv6 Prefixes H-8
IPv6 Prefix Example H-9
IP Transition Strategies H-10
IPv6 Resources H-11
Extended Topology and IPv6 . . . . . H-12
IPv6 Management Example H-13
Managing Coexistence with Extended Topology H-14
IPv6 Views H-16
IPv6 Status Propagation H-18

Contents

IPv6 Network View H-20
Focused IPv6 Network View H-21
IPv6 Node View H-22
IPv6 Interface View H-23
IPv6 Prefix Group View H-24
IPv6 System Requirements H-25
IPv6 Router Requirements H-26
IPv6 Management Limitations H-28
IPv6 Discovery H-29
Configuring IPv6 . . . . . H-30
Configuring IPv6 Seed File H-31
Configuring IPv6 Discovery H-33
Consistent Hostnames H-34
IPv6 Status Monitoring H-36
Configuring IPv6 Polling H-37
Configuring Prefix Names H-39
IPv6 Logfiles H-40
Lab Exercises H-41

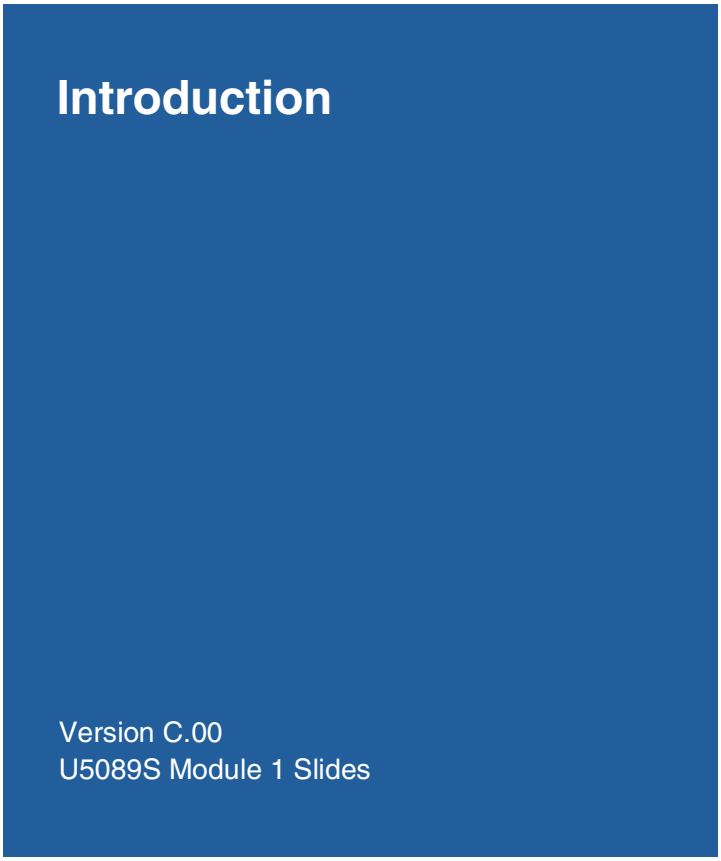
I. Lab Solutions

Chapter 1, Introduction I-2
Chapter 2, Discovering Connectivity with Extended Topology I-4
Chapter 3, Enabling Extended Topology I-5
Chapter 6, Controlling Extended Topology Discovery I-10
Chapter 5, Scaling netmon Discovery and Polling I-31
Chapter 8, Active Problem Analyzer I-32
Chapter 9, Configuring Extended Topology Discovery of OSPF I-34
Chapter 10, Configuring Extended Topology Discovery of HSRP I-50
Chapter 11, Introduction to Event Reduction I-56
Chapter 12, Configuring Event Correlation I-58
Chapter 13, Introduction to Composer Development I-68
Chapter 14, Creating a Basic Correlator I-69
Chapter 15, Using Variables in Correlators I-74
Chapter 16, Using Additional Correlators I-90
Chapter 17, Relating Events from Multiple Sources I-101
Chapter 18, Using Callbacks and Built-In Functions I-107
Chapter 19, Best Practices and Tools I-116
Chapter 21, Configuring syslog Messages for SNMP I-118
Chapter A, Viewing Your Environment with Dynamic Views I-120
Chapter B, Securing Dynamic Views I-122
Chapter C, Using Problem Diagnosis I-126
Chapter D, Configuring Problem Diagnosis I-127
Chapter E, Constructing Advanced Filters I-130
Chapter G, Device Managment Details I-135
Chapter H, IPv6 in Extended Topology I-138

Contents

1 Introduction

Module Objectives



At the completion of this module the student will be able to:

- Describe the value of event reduction.
- Describe the purpose of event correlation.
- List methods of reducing alarms in the Alarm Browser.
- Describe which protocol services are managed by Extended Topology.

Course Outline

Course Outline

- This advanced NNM configuration course covers
  - Event reduction and root cause analysis
    - Configuring ECS parameters for correlations
    - Designing and creating your own correlators in Composer
  - Managing specific protocols with Extended Topology
    - OSPF
    - HSRP
    - IPv6
  - Troubleshooting Extended Topology discovery
  - Recent NNM improvements
    - Home Base and dynamic views
    - Problem Diagnosis

- This course assumes you are already familiar with
  - the NNM event subsystem, pmd, and configuring events
  - NNM discovery (seed files, filters, how it works)
  - Networking protocols and services of interest

©2004 Hewlett-Packard Development Company, L.P.

NNM Product Structure

NNM Product Structure

Product packaging/pricing allows increased flexibility for you to buy the product that fits your needs:

NNM Starter Edition

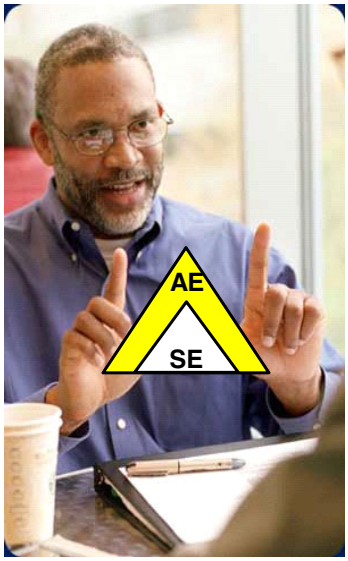
Entry-level product designed for smaller networks needing basic network (primarily layer 3 routers/hubs/PCs) from a single management station.

NNM Advanced Edition

Designed for all sizes of networks requiring:

- •advanced management of switches/VLANs,
- •sophisticated root-cause analysis, and
- •distributed management,for large networks spanning multiple sites/departments.

This is the “platform” for even greater advanced capabilities delivered through the add-on NNM Smart Plug-ins.



U5089S ©2004 Hewlett-Packard Development Company, L.P. 3

NNM Extended Topology, formerly available separately, is one piece of functionality in NNM Advanced Edition (NNM AE), providing layer 2 and VLAN management capabilities. Additional features like HSRP, OSPF are sold separately as a part of the advance routing protocol SPI. The new products in NNM are:

- NNM SE 250 Node Pack. You can add as many 250 node packs as you need (no top-end limit). There is not an UNLIMITED SE or intermediate level.
- NNM AE 250, 1000, 5000 Node Pack and an UNLIMITED Node pack. You can mix any packs and NNM adds all available licenses. Again you can add as many 250 node packs or 5000 node packs as you need.
- NNM SE 250 node pack to NNM AE 250 node pack upgrade.

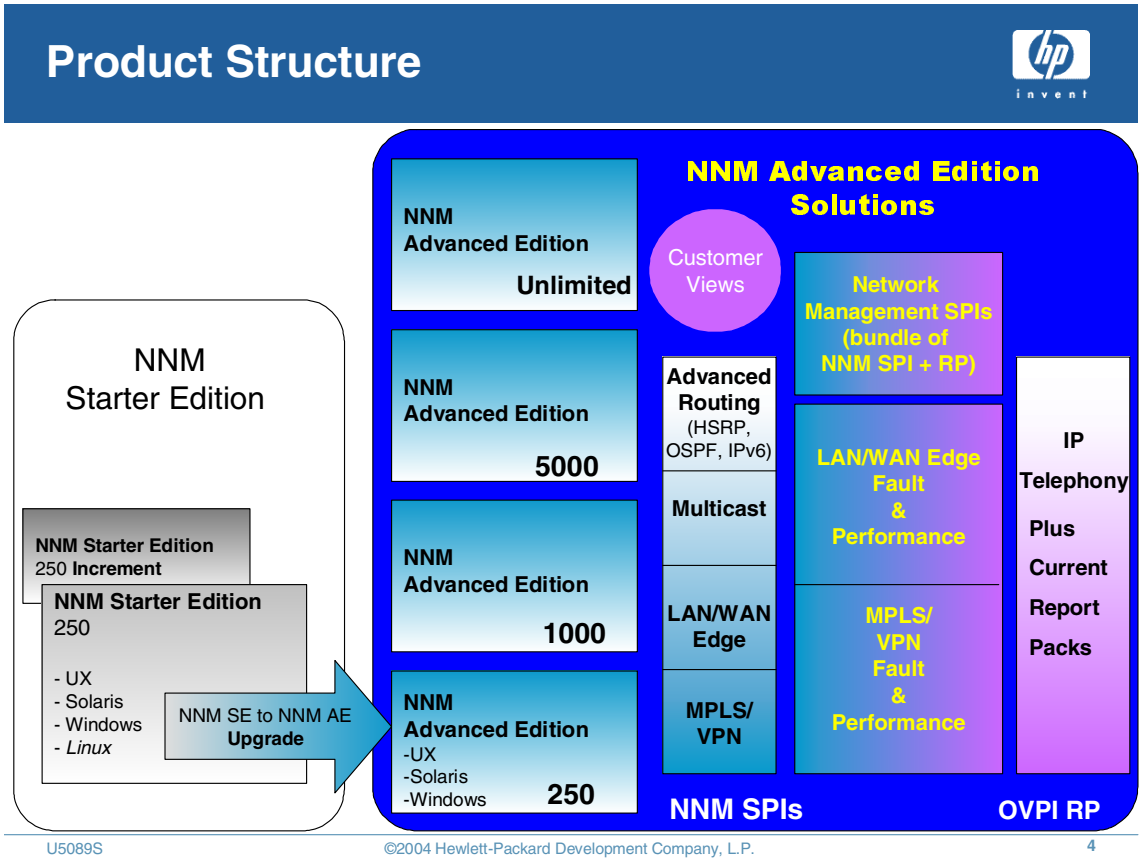
In addition to the technical changes, HP has adapted to your business needs in the market.

- With more license levels available, you can purchase just the amount of network management that your environment requires.
- Starter Edition and Advanced Edition provide perfect fit for different target environments.
- Advanced Edition is the foundation for more advanced capabilities.
- With cost-control measures in place throughout industry, a single NNM station can manage thousands more objects. This scalability lowers your total cost of ownership by reducing the

need for multiple servers.

- In today's over-complex world, too many point-focused products become a confusing fog. By combining the key technologies for management in the network -- NNM, Extended Topology, and Problem Diagnosis -- HP has created a simplified world where your management needs are all met in one place with ease.
- And HP has made these benefits available while still keeping the overall cost structure significantly lower than the competition's.

Product Structure



NNM Advanced Edition

Network Node Manager Advanced Edition is designed for all sizes of networks requiring advanced management of switches, sophisticated root-cause analysis, and distributed management for large networks spanning multiple departments.

Network Node Manager Advanced Edition easily integrates new services, technologies and increased usage demands to optimize your total cost of ownership and operation. Out-of-the-box automation and intelligence helps your staff understand the components of your network services and their relationships with network devices in complex switched environments for increased staff efficiencies.

Intelligent Diagnostics for Networks provides unique advanced root-cause analysis, dramatic event reduction and troubleshooting for layer 2 and 3 networks to reduce your mean time to repair cycles.

Features

- Beyond advanced root-cause analysis

- Intelligent Diagnostics for Networks (ID for Networks) for unique advanced root-cause analysis, dramatic event reduction, troubleshooting for layer 2 and layer 3 networks
- Out of the box correlators with correlation composer for easy fine tuning
- Intelligent multi-threaded poller and problem state analyzer
- Dynamic views show the relationship between devices in complex switched environments and network services
- Broad range of device and protocol support (IPv6, IP Telephony, HSRP, OSPF, Multicast)

Related Products

These new NNM SPIs and Network Management SPIs are Advanced Edition only and extend the capabilities of NNM Advanced Edition for specific areas.

- NNM SPI for LAN/WAN Edge
- Network Management SPI for LAN/WAN Edge (includes NNM SPI plus report pack as discounted bundle)
- NNM SPI for MPLS IP VPN
- NNM SPI for advanced routing
- NNM SPI for multicast
- IP telephony management solutions
- Performance Insight

  Performance Insight for Networks isolates performance problems in complex networks. OpenView Performance Insight provides integrated path views of switched networks for complete diagnosis of connection problems.

Network Node Manager Starter Edition

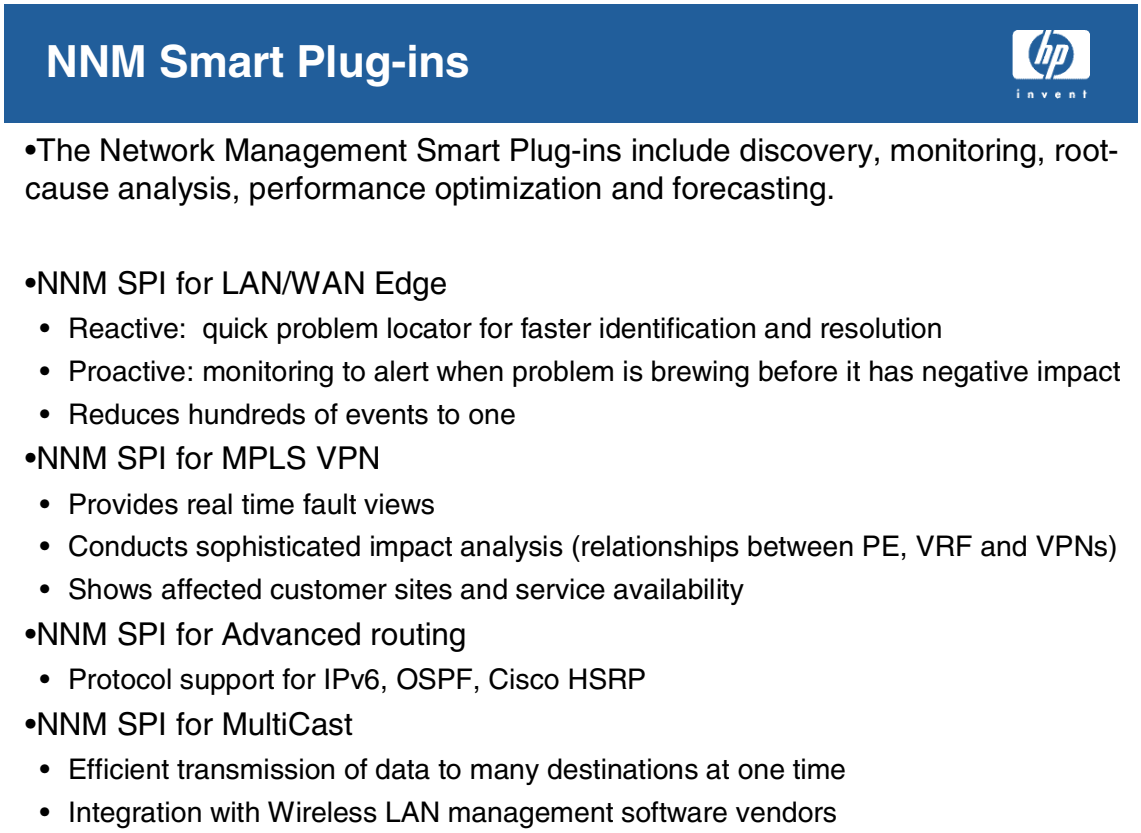
NNM Starter Edition is for smaller networks needing basic network management (layer 3 routers/hubs/PCs) from a single management station. It has built-in intelligence that helps your staff understand all the components and relationships in your network. Easy-to-use tools are provided to help you identify and resolve problems fast. These tools fit all levels of expertise while still providing the flexibility to easily custom fit to your unique business needs.

The Starter Edition is a subset of the Advanced Edition. NNM SE provides an easy upgrade path to NNM Advanced Edition for larger, more complex switched network environments.

Features

- Intuitive Home Base GUI for summary and quick access to events and maps
- Accurate discovery and mapping of your network and relationships
- Targeted views gets you to the heart of the problem
- Filters and correlates events to cut through the noise and get to the real problem
- Correlation composer for easy fine tuning of out of the box correlators
- Easy upgrade to advanced capabilities when you need them

Network Management SPIs



As your network service requirements expand, you can add capabilities to Network Node Manager Advanced Edition with the Network Node Manager Smart Plug-ins (SPIs), which are available separately.

Network Node Manager SPI for **Advanced Routing** extends the capabilities of Network Node Manager Advanced Edition to intelligently diagnose dynamic networks for IPv6, OSPF (Open Shortest Path First) and Cisco Hot Standby Routing Protocol (HSRP). This SPI takes advantage of Network Node Manager's rich capabilities to easily integrate new services, technologies and increased usage demands.

Network Node Manager SPI for **LAN/WAN Edge** is your enterprise solution to monitoring the Frame Relay connectivity between you and your service provider. It analyzes the hundreds of events that occur during connectivity failures and pinpoints the problem as being local, remote or due to potential configuration issues. In addition, it notifies you of the customers or sites affected by an outage. It can provide this information for any switch supporting the Frame Relay MIB (RFC 1315).

Network Node Manager SPI for **MPLS VPN** monitors MPLS service availability and shows who is affected when a network interface goes down. This SPI will assist in prioritizing failures with real-time fault views and sophisticated degradation analysis by helping you understand the critical relationships between customer sites, the provider edge and VPNs.

Network Node Manager SPI for **Multicast** enables a network operator to view the topology of a

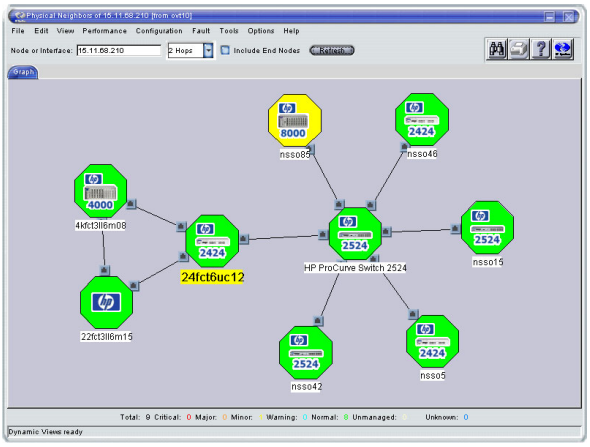
multicast environment and its status. It helps your operators initiate fault isolation for multicast applications by automatically discovering and displaying maps of multicast routing topology and relationships, measuring multicast traffic rates, and generating SNMP alarms based on multicast activity.

New technologies, such as **IP Telephony** environments, add new management challenges to your network operation. To meet these new needs, HP OpenView provides an integrated IP Telephony management solution for leading IP Telephony vendor infrastructures. Network Node Manager, HP OpenView Performance Insight and HP OpenView Operations integrate with products from leading vendors to provide the capability for fast fault isolation and repair of IP Telephony infrastructures and to generate specific network performance reports for optimal use of IP Telephony resources.

Networks that Require Extended Topology for Discovery

Networks that Require Extended Topology for Discovery

- Heterogeneous layer 2 switched network management (LAN & WAN)
- Private IP Addresses (overlapping address domains)
- Targeted views for quickly identifying root cause
  - View switched environment and complex relationships between devices
  - View network services such as OSPF and VLAN
- Enhances NNM views: neighbor, station, internet
- Superior root-cause analysis via improved path detection



U5089S ©2004 Hewlett-Packard Development Company, L.P. 6

If you have a connectivity problem, you want to see a layer 2 and layer 3 view (view of your switched and IP environment). HP OpenView Network NNM AE's Extended Topology functionality discovers layer 2 device information and displays device connectivity. Extended Topology manages heterogeneous switched layer 2 environments as well as routed layer 3 environments. Extended Topology provides improved layer 2 connectivity visualization, including port aggregation (trunking), meshes, and switch board and port information in all dynamic views.

NNM's Extended Topology Functionality

NNM's Extended Topology functionality augments NNM by discovering and displaying additional device connectivity information that you can use to diagnose network problems. It includes, but is not limited to, the following features:

- Management of heterogeneous switched layer 2 environments and routed layer 3 environments.
- Additional and enhanced dynamic views.
- Views from protocols and technology running on top of your network, such as OSPF and VLAN.

- Launching targeted views from events for rapid problem resolution.
- Discover and monitor network domains that use Static NAT.

Focused Visualization

Extended Topology provides targeted views that allow greater clarity of information by targeting how the visualization of specific components in the network is presented. Views are included that add value by visualizing potentially complex relationships between devices in an easy to understand context. You can also visualize protocols running on top of your network, such as OSPF and VLANs.

Improved Root Cause Analysis

Extended Topology improves root cause event correlation via a richer layer-2 topology path agent. With NNM SE in a VLAN/switched environment, the layer 2 connectivity and meshing information does not follow the layer 3 IP hierarchy. The ConnectorDown correlation may not be accurate in finding the root cause. By using the Extended Topology physical topology to build critical paths, root cause analysis is more accurate.

What Extended Topology Discovers

- Extended Topology accesses information from:
  - Standard MIBs
  - Vendor-specific MIBs
  - Other unique methods
- It models and displays “layer-two” connectivity
- Information about
  - VLANs
  - Overlapping Address Domains
  - Frame Relay (requires a SPI)
  - OSPF (requires a SPI)
  - HSRP (requires a SPI)
  - IPv6 (requires a SPI)

U5089S ©2004 Hewlett-Packard Development Company, L.P. 7

Extended Topology uses a variety of mechanisms to discover and lay out an accurate layer 2 topology for supported devices. Extended Topology discovers using standard SNMP MIBs, vendor specific MIBs and non-SNMP methods where appropriate. Extended Topology stores this information in the database and leverages it in a variety of ways including using layer 2 topology to build critical routes used to determine root cause for network connector failures.

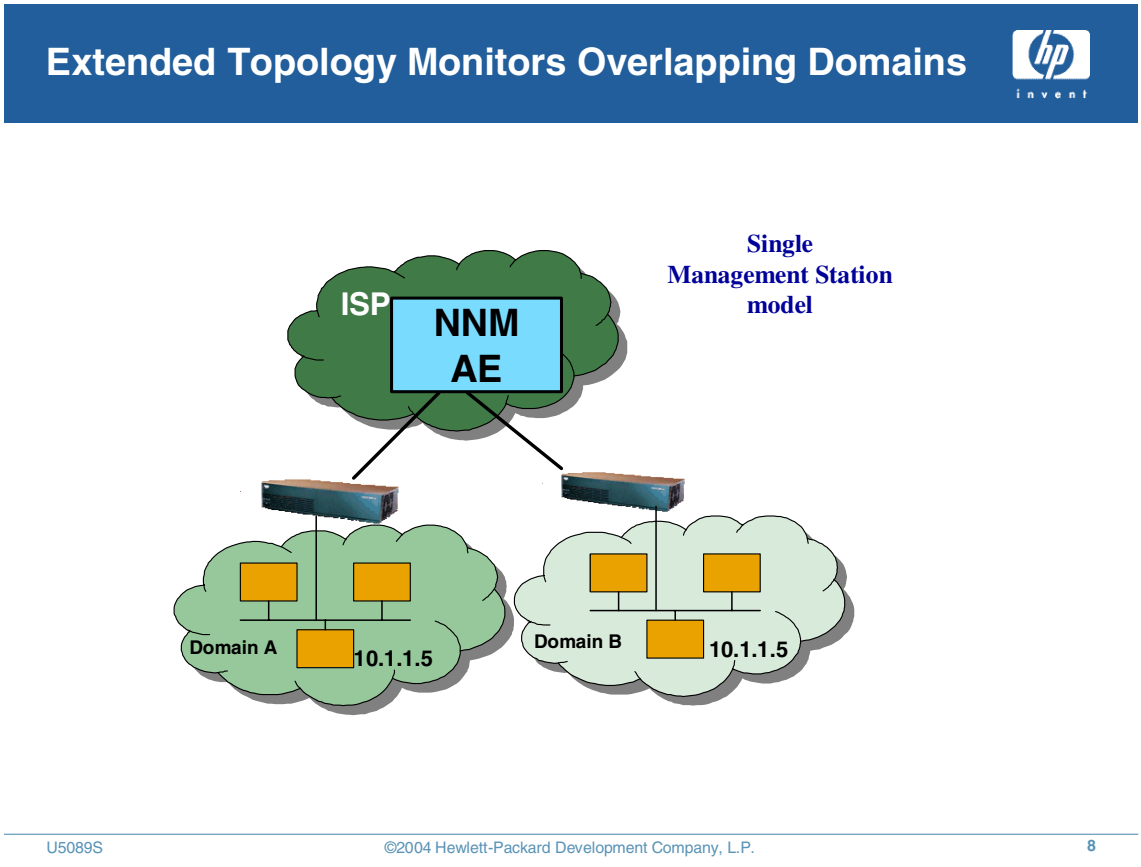
Extended Topology supports a specific list of switches. On an on-going basis HP adds support for additional switches and update the list.

Extended Topology discovery obtains the following information:

“Layer-two” connectivity	Extended Topology discovers layer 2 connection information and uses this information to map managed nodes and their neighboring port relationships. Extended Topology calculates layer 2 network paths from the Extended Topology management station through managed devices. Among other things, this results in better diagnostic information being delivered to the HP OpenView ECS (Event Correlation Services) used by NNM.

VLAN information	Extended Topology discovers VLAN information from managed devices. This data can be displayed in VLAN-specific views, or overlaid on other views (such as the Neighbor view). VLAN views tell you which switches are participating and provide a physical connectivity overlay view.
ATM information	Extended Topology discovers ATM information such as Virtual Path Identifier (VPI) and Virtual Channel Identifier (VCI). You can view this information by moving your mouse pointer over an interface that supports ATM. Extended Topology can work with any vendor that talks ILMI. (ILMI is the management interface between ATM switches.)
OSPF information	During an OSPF discovery, Extended Topology discovers which area OSPF devices are located in, and how the areas relate to one another. The OSPF view displays this information. OSPF discovery is a separate process from the Extended Topology discovery. You run OSPF discovery by using a manual discovery procedure.Our solution is vendor-independent, for any device which supports OSPF MIBs. (Requires Advanced Routing SPI.)
IPv6 information	Running IPv6 discovery results in Extended Topology discovering global, site-local, and link-local addresses. All routers must be dual-stacked for IPv6 discovery to function properly. To prepare Extended Topology for IPv6 discovery, you must run the `$OV_BIN/setupExtTopo.ovpl` script. (Requires Advanced Routing SPI.)
HSRP information	Extended Topology discovers and displays HSRP information from managed devices that support the HSRP protocol. While HSRP discovery is automatic, there are important preliminary steps you need to take to assure correct HSRP discovery and monitoring. (Requires Advanced Routing SPI.)
Static NAT and private IP addresses	Extended Topology discovers and monitors network domains that use Static NAT, even when these addresses overlap. You can use Extended Topology to distinguish these overlapping addresses by configuring each address group into an OAD (overlapping address domain).

Extended Topology Monitors Overlapping IP Address Domains

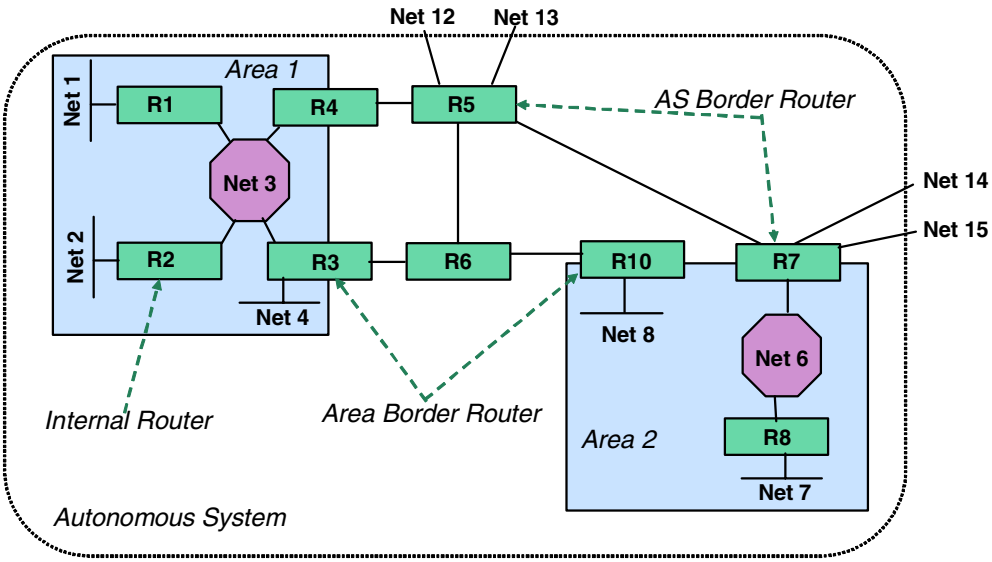


NNM Extended Topology functionality supports management of networks with overlapping IPv4 addresses. Extended Topology provides support for connectivity discovery, monitoring and event handling from networks with overlapping IP addresses. It identifies events by the unique device and domain from which they originate.

NNM allows operators to receive events from overlapping IP domains without being confused about the correct source and to navigate easily to views of that specific domain – from a single management station. This reduces Mean Time to Repair by getting to the device quickly and understanding which customer is affected and prioritizing accordingly.

Extended Topology Monitors OSPF

Extended Topology Manages OSPF



©2004 Hewlett-Packard Development Company, L.P.

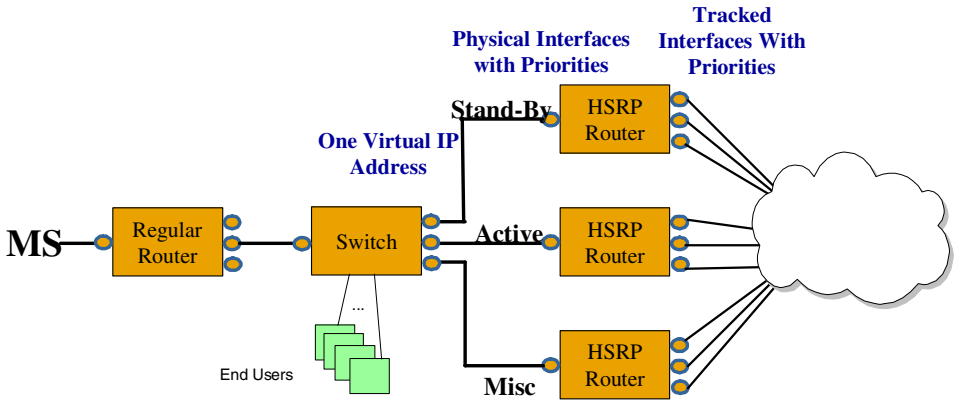
Open Shortest Path First (OSPF) is a routing protocol that allows routers to collect and share information to build a topology of the network. As a “link-state” routing protocol, it provides more accurate routing and faster response to changes than older “distance-vector” routing such as RIP.

Extended Topology manages OSPF environments by collecting internal routing information from routers and creating views showing areas and area border routers. This requires the Advanced Routing SPI.

Extended Topology Manages HSRP Groups

Extended Topology Manages HSRP Groups

•Network redundancy for IP networks



U5089S ©2004 Hewlett-Packard Development Company, L.P. 10

The Hot Standby Routing Protocol (HSRP) allows near-100 percent network uptime by providing network redundancy for IP networks. User traffic is immediately and transparently rerouted when there is a first-hop failure in a network edge device or access circuit.

Routers in an HSRP group share an IP address and MAC address. Extended Topology allows you to monitor the health of your HSRP service overall, as well as the individual interfaces in the network.

Extended Topology can be configured to automatically monitor the status of HSRP groups and to track active and standby interfaces in the group.

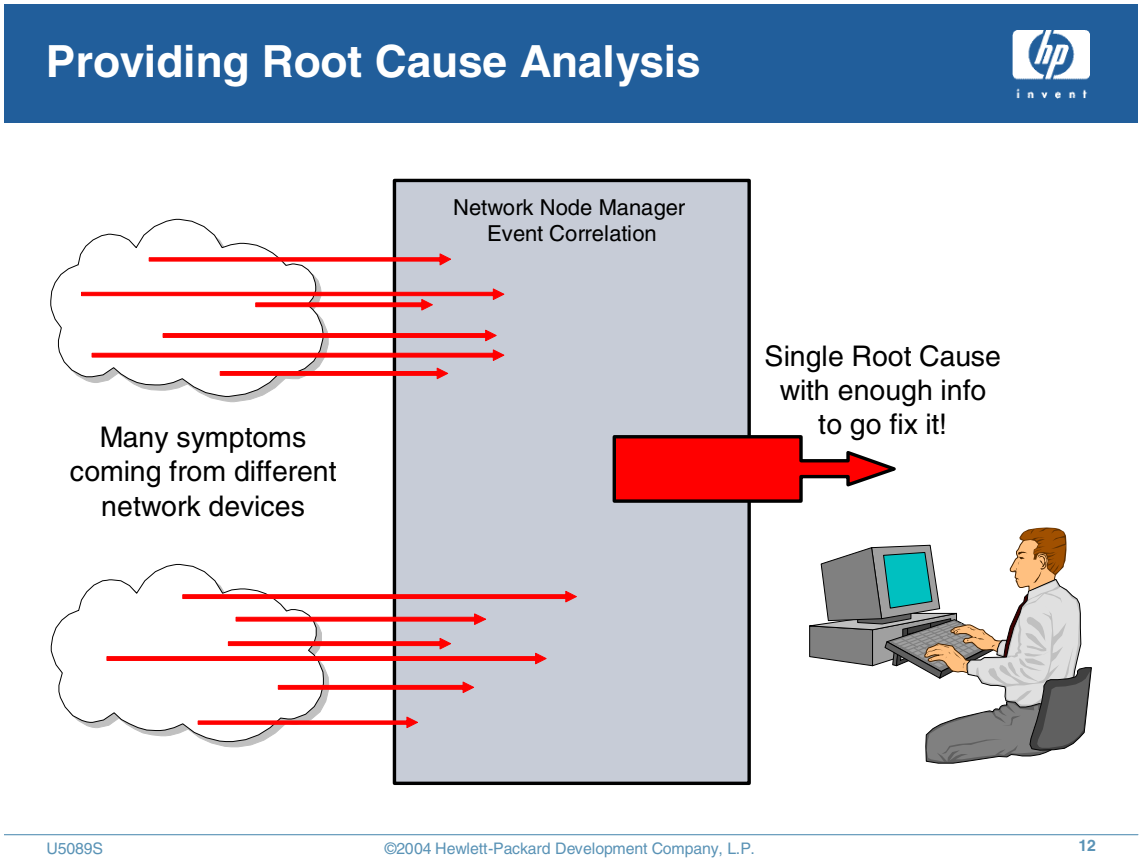
This functionality requires the Advanced Routing SPI.

Extended Topology Discovers and Manages IPv6 Networks

Many areas of the world are exhausting their IPv4 address space and turning to IPv6 for additional addressing and security.

Extended Topology can manage both IPv4 and IPv6 networks from a single management station using different views. You can configure Extended Topology (with the Advanced Routing SPI) to discover IPv6 information through SNMP. Once a device is discovered, Extended Topology monitors status using ICMPv6 (ping).All status change events are integrated to the NNM event system, allowing automatic actions as well as visualization of your IPv6 environment.

Providing Root Cause Analysis



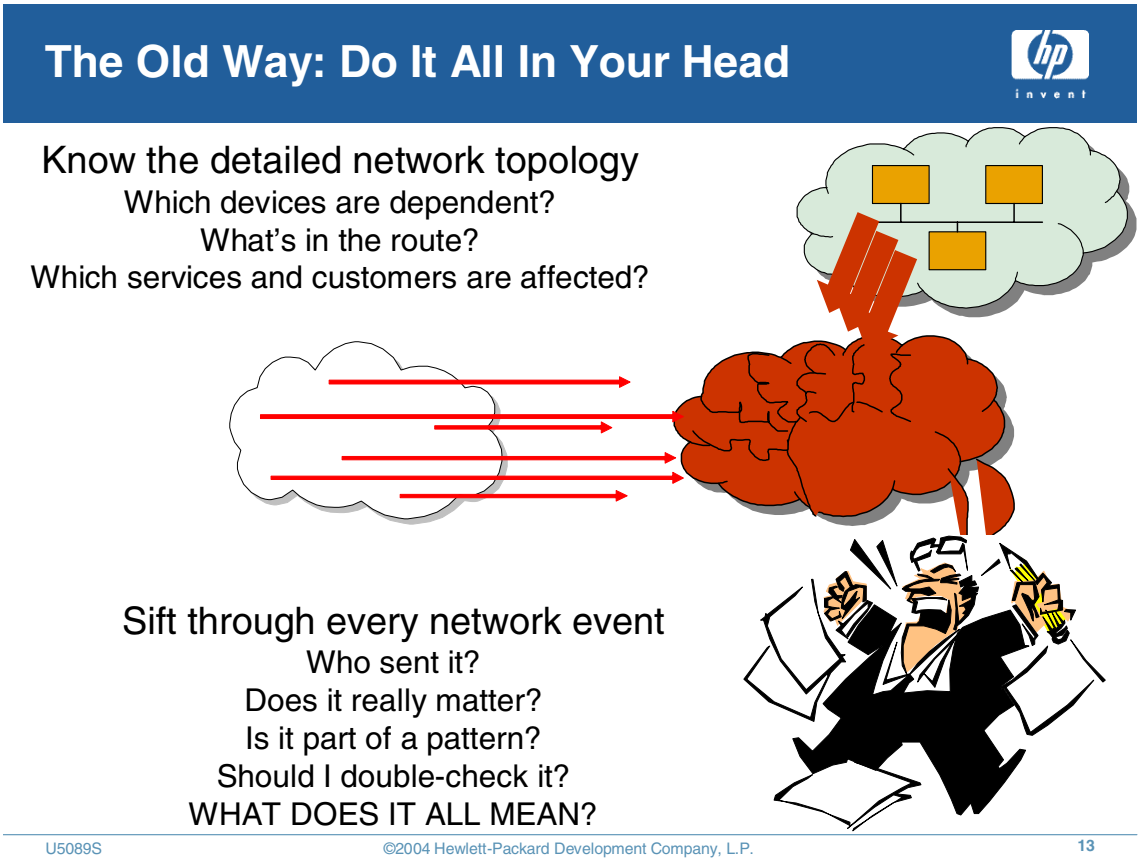
Problem

A condition in the network that generates masses of events within a small time frame, such as a back hoe digging into a line, becomes what is know as an event storm. Network failures can also cause redundant or repeated patterns of events. Event storms can overwhelm a network operator so that they might miss seeing important alarms. Missed information delays detecting and fixing the problem.

Solution

The solution is to isolate important events, analyze them and generate useful information from them to generate a higher level alarm. This is the purpose and function of event correlation.

Difficult Diagnostics: Do It All In Your Head



Consider the "old school" of network management where an administrator-guru had to keep the entire network topology in his head and remember how all the devices were really interconnected. Then when network events arrived (and they came in storms), he had to think about each one to determine its relevance.

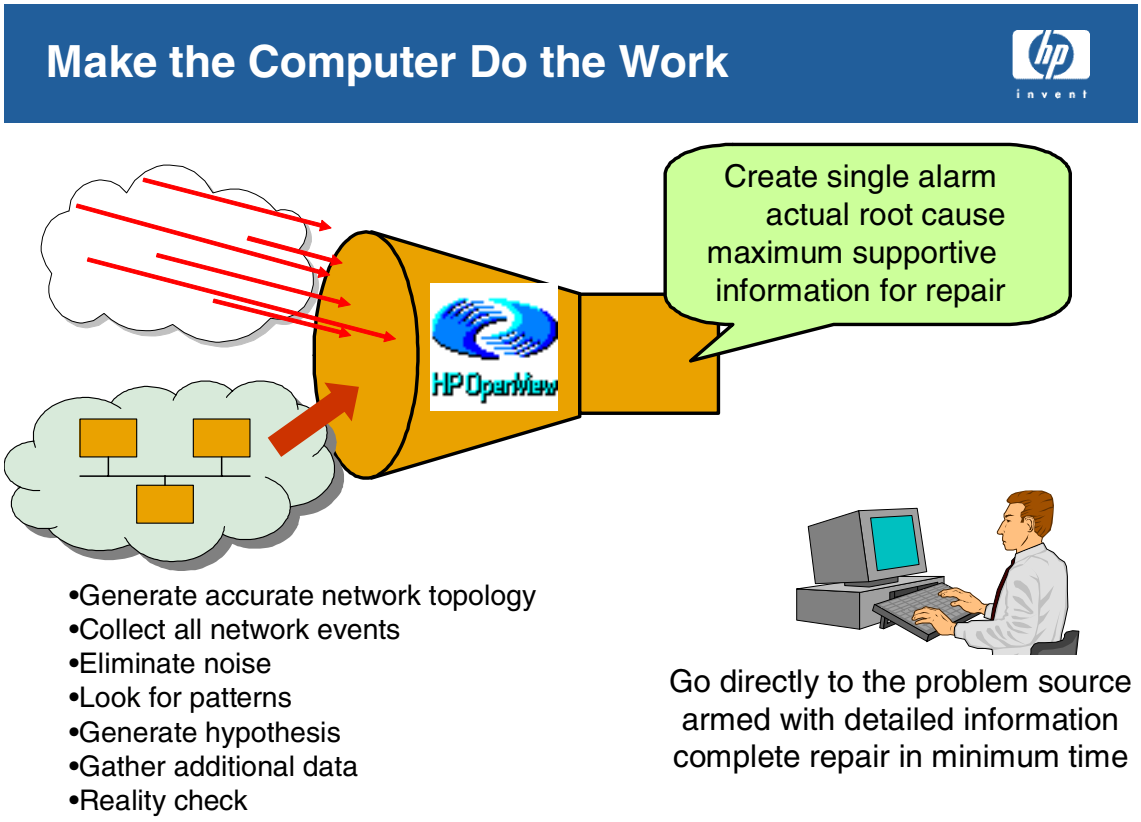
It was up to the administrator to recognize patterns of network events that pointed to a specific problem or outage, then track down what services would be affected. The Mean Time to Repair just kept getting longer and longer.

While this may have worked when networks consisted of routed layer 3 environments, with the overwhelming complexity of devices, protocols, and services in the last 5 years, the job became -- overwhelming!

No one could succeed with this level of complexity and keep their network up and running and maximize the overall benefits that networking promised. Coping by adding more people or more systems only increased the Total Cost of Ownership, not reduced it. Service Level Agreements were violated and business and productivity were lost.

NNM's outstanding fault diagnostic features have taken a quantum leap ahead with NNM.

Make the Computer Do the Work



Rather than forcing the network administrator to do all the work of keeping the topology in his mind and watching all the network events, NNM handles all the complexity for you.

With Extended Topology, NNM charts the actual layer 2 connectivity of your devices and recognizes meshed paths and when a device outage truly does degrade your network availability.

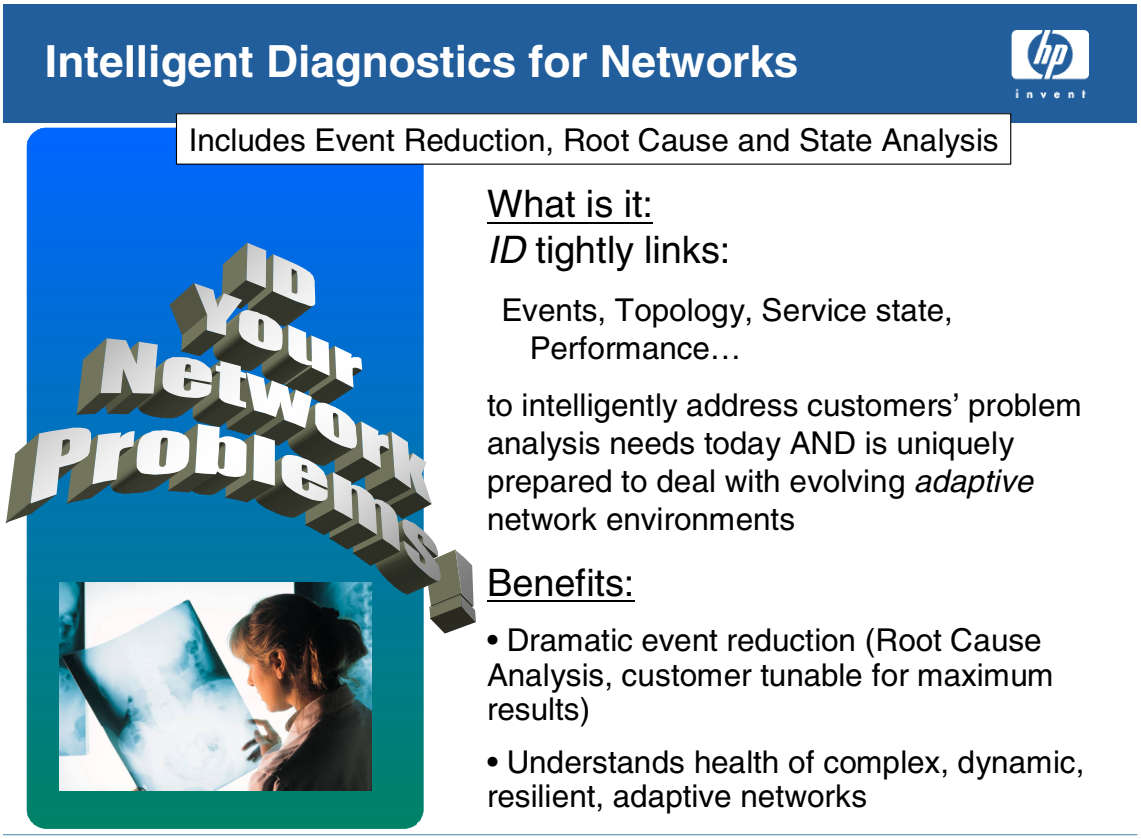
In addition, NNM understands how services like HSRP operate on your network. When one interface on one device goes down, NNM verifies the overall health of the service remaining. It even automatically waits for Spanning Tree reconfigurations to quiet down before verifying problems to avoid giving you any false alarms.

While the topology subsystem has a much more intricate understanding of your network, the event system has taken a significant lesson in what it all means. Rather than continually informing you of every tiny minutiae, the event system correlates the patterns, eliminates the noise, generates probable causes, and even verifies the data before alerting you to take action.

This allows you to go directly to the root problem and fix it with a significantly lower Mean Time To Repair.

With your less-stressed environment, you can even teach NNM's flexible event system about unique devices you may use in your network and how to interpret various messages they may emit. Transferring your knowledge into NNM's intelligence base lowers your Total Cost of Ownership.

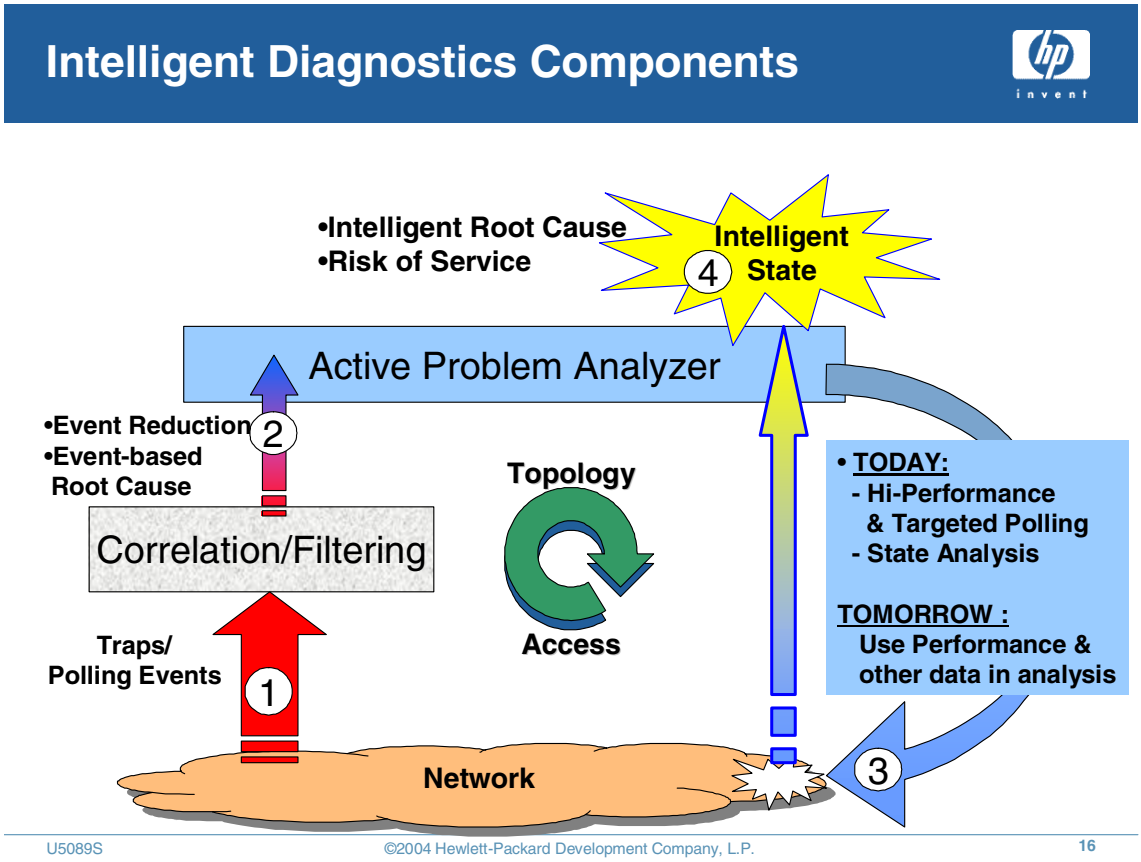
Intelligent Diagnostics for Networks



HP calls this integration of detailed topology understanding and flexible event resolution Intelligent Diagnostics. By intelligently understanding the state of your network, and being able to learn about new devices and heuristics, NNM takes the analysis of network problems from a manual task to a completely automated process.

- Integrates NNM's leading-edge components
  - — ECS
  - — Composer
  - — Active Problem Analyzer
- Teach the event subsystem the heuristics you've been applying
  - — Flexibility in describing symptoms
  - — Interprets based on current network state
- Adapts to network evolution without needing complete rediscovery
- Tied into state analysis for accuracy and verification
- Only issues root cause alarms. All internal updates are silent.

Intelligent Diagnostics Components



The Active Problem Analyzer is the brains of Intelligent Diagnostics for Networks and can intelligently handle new protocols and data sources.

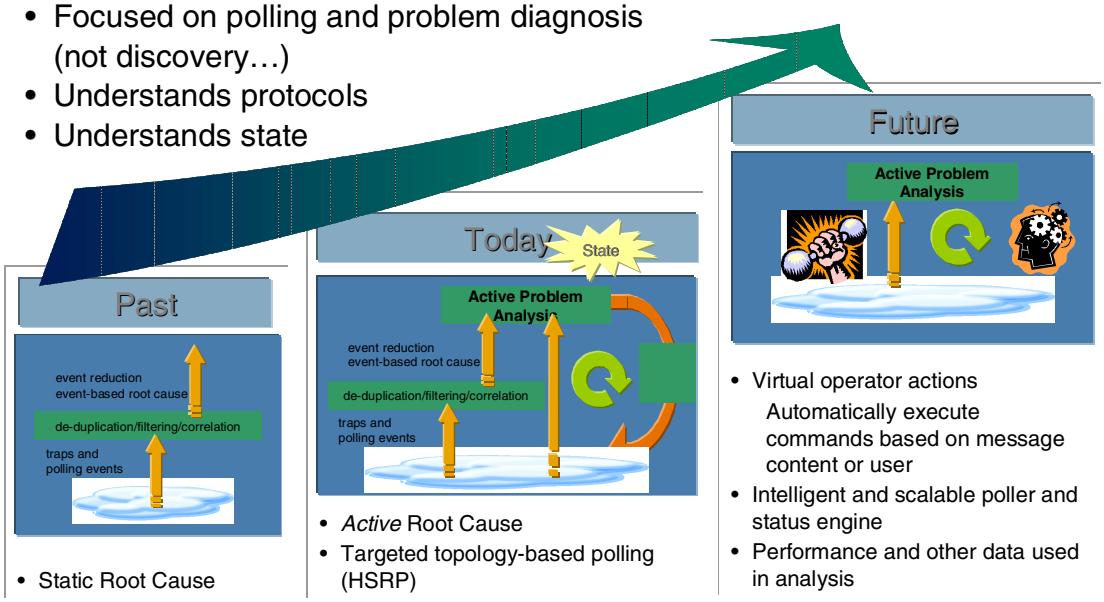
Benefits of this advanced root cause, event reduction and state analysis are

- Improved event reduction. Huge reduction in “noise” events
- Meaningful, descriptive events
- Understands health of complex, dynamic, resilient networks
- Predictive analysis of service – risk of future failure
- Easy end-user fine-tuning of behavior
- Investment protection
- Increased accuracy of recommendation from software
- Decreased staffing requirements (hours, expertise)
- Reduced time to repair (go directly to real problem)

Intelligent Diagnostics - Where Can It Lead?

Intelligent Diagnostics - Fastest MTTR

Intelligent diagnostics
- Focused on polling and problem diagnosis (not discovery…)
- Understands protocols
- Understands state



- Static Root Cause
- *Active* Root Cause
- Targeted topology-based polling (HSRP)
- Virtual operator actions
  Automatically execute commands based on message content or user
- Intelligent and scalable poller and status engine
- Performance and other data used in analysis

©2004 Hewlett-Packard Development Company, L.P.

Comparing Event Reduction Mechanisms

What Reduction Mechanism is the Best Choice for a Specific Problem?

Before attempting to develop a correlation or de-duplication, consider:

- What the operator really wants to see
- Level of complexity in the mechanisms

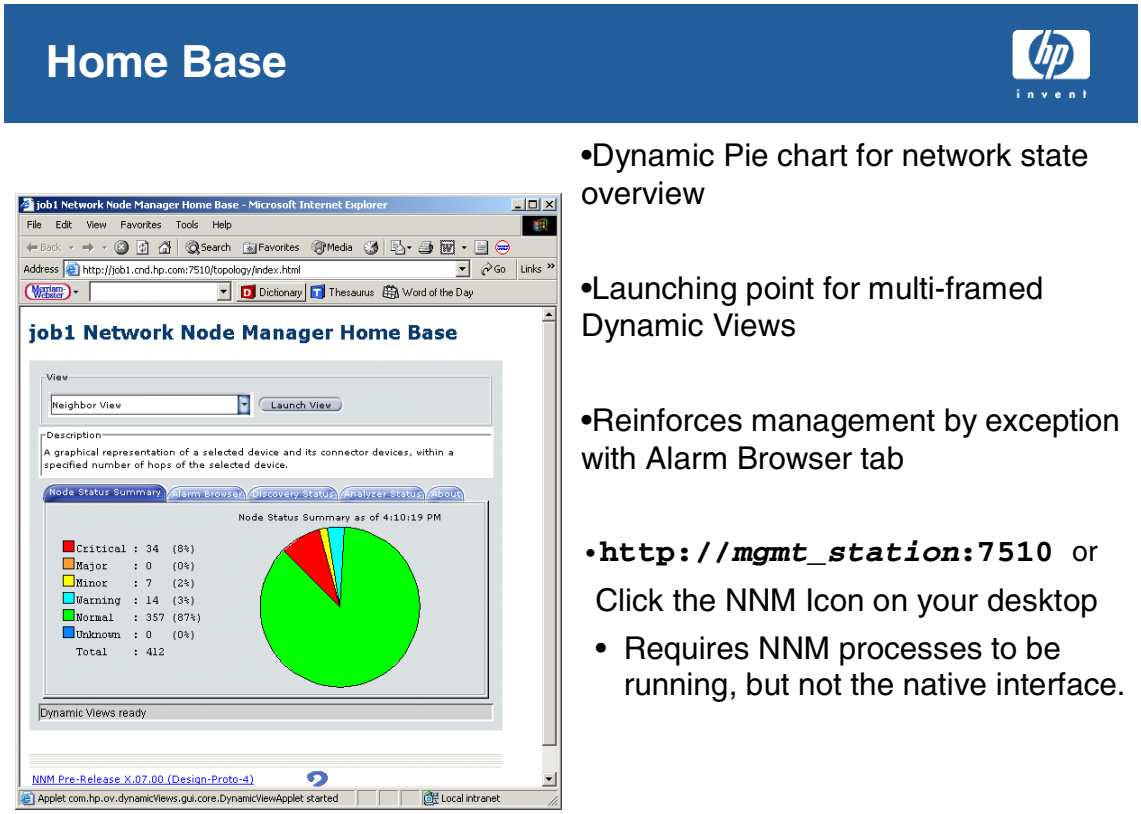
The following list of mechanisms is a rank order of complexity in terms of developing an event reduction, the simplest to develop being first.

1. Log Only or Ignore
2. De-duplicate
3. Composer correlator
4. ECS correlation

Log only and de-duplication are mechanisms that operate on a single event type, independent of other events. Composer correlators and ECS correlations are more powerful in that they can be

designed and developed to identify a pattern of events and reduce that pattern to a single root cause.

Home Base



Network Node Manager's graphical user interface, "Home Base," is specifically designed to provide all users a simple starting point for using the product. It provides an easy-to-understand summary of your network's status and quick access to detailed event data and targeted maps. Home Base brings the power of Network Node Manager to even novice users.

You can access Home Base from your browser using the following URL:

`http://hostname:7510/`

In Windows operating system, double-clicking the Network Node Manager icon in the HP OpenView program group or on your desktop starts Home Base.

This is a launching point for many of the Dynamic Views. In addition to launching views from Home Base, you can select tabs that cause Home Base to display additional information about your network. Home Base contains access to:

- Dynamic Views

  You can use the Launch View button to launch the requested view. As you select different views, the description field changes.

  The views available from Home Base vary depending on whether you purchased the NNM Starter or Advanced Edition, and whether you installed additional Network Solution SPIs (Smart Plug-ins). Here are a few of the views available from Home Base:

— `Node view`: This command opens a menu that you can use to configure a topology view based upon the various filters available. A second selection criteria is the status of the managed devices that pass the selected filter.

— `Neighbor view`: This view shows you a graphical representation of a node and its directly-connected neighbors, regardless of their IP address (layer 3).

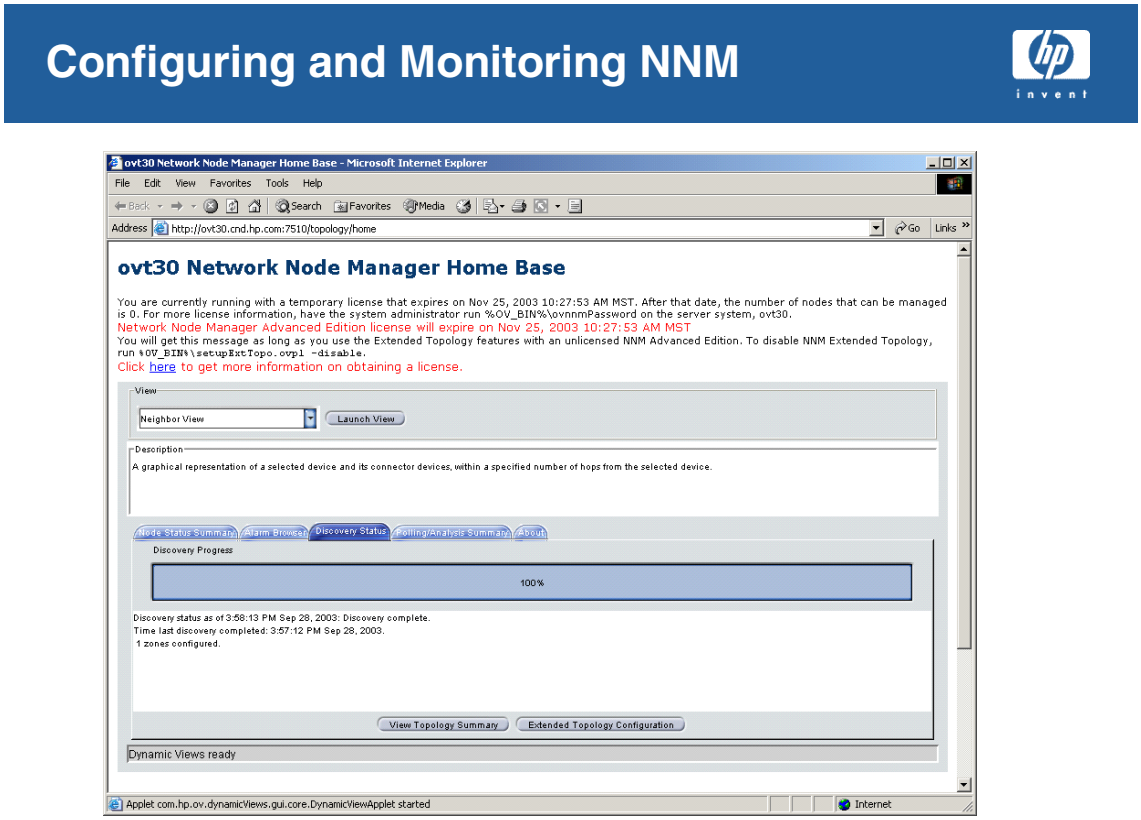
See the *Using Extended Topology* manual for more information about Home Base.

- The Extended Topology discovery status progress bar
- Alarm categories which can open the web-based Alarm Browser
- The pie chart is dynamically updated to reflect the nodes in the network; the numbers change showing discovery is on.

As you work, keep Home Base open, even if minimized. Closing Home Base closes all dynamic views that are using its context to run.

Note that the forward/back button or refresh will kill the JVM, and hence the window. Since multiple windows are generated by the same JVM, all windows will go away. To avoid this problem, launch new views in new windows.

Configuring and Monitoring NNM

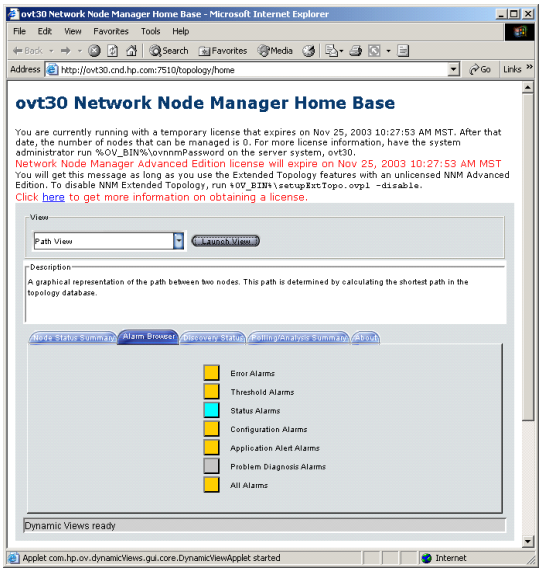


 ©2004 Hewlett-Packard Development Company, L.P.

Home Base also provides access to tools for configuring and monitoring the Extended Topology functionality of NNM Advanced Edition. This is found in the Discovery tab and the Polling/Analysis Summary tab.

Management By Exception

Management By Exception



- Open the Alarm Browser from Home Base.
  - The Alarm Browser reflects the current state of the network and services.
- Operator sees
  - the fewest possible number of alarms
  - with the most useful information
  - to solve the root problem the fastest.
- Launch the Dynamic View most related to the problem described in the alarm.
- Access troubleshooting tools from the Dynamic View interface

 ©2004 Hewlett-Packard Development Company, L.P.

To review the outstanding exceptions in your environment (alarms):

1. Start Home Base and view the Alarm Browser tab.
2. From there click on a category and browse alarms.
3. Select the alarm and launch the Dynamic View related to the problem to see the surrounding environment.
4. Executing troubleshooting tools from the view.

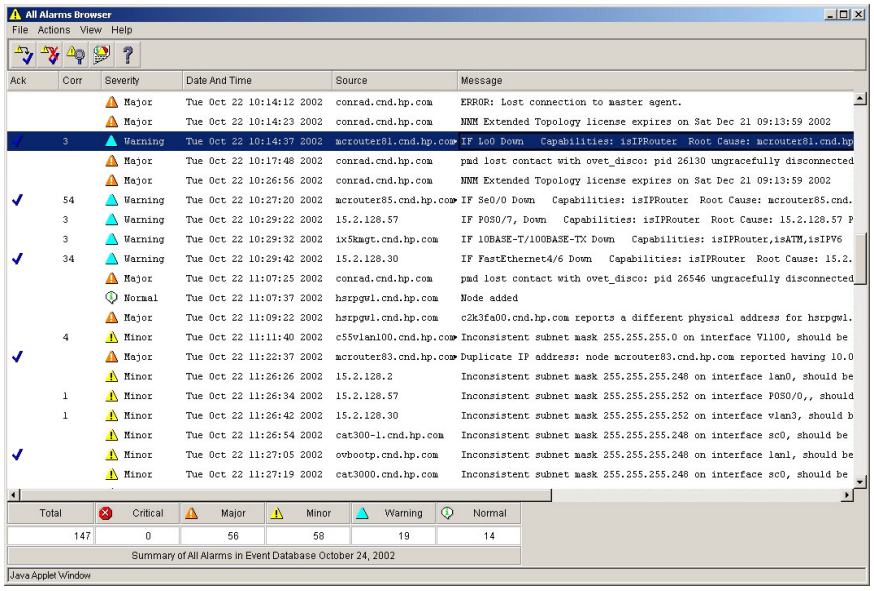
To resolve a problem by first locating the device:

This is referred to as management by out-of-band exception, such as a phone call.

1. Browse to Home Base.
2. From there, launch new view windows. You can drill down to the set of nodes (the subnet or neighbors of a node).
3. Once you locate the view you desire, use Dynamic Views to highlight VLANs, find by IP address, expand what you are viewing, or navigate to other network management tools.

The Web Based Alarm Browser

The Web Based Alarm Browser



 ©2004 Hewlett-Packard Development Company, L.P.

The Alarm Browser is a view of the received alarms contained in the Binary Event Store. When an Alarm category button is colored in Home Base, you know that you have outstanding alarms which need attention. The color in the `Alarm Categories` window is the color of the most severe alarm.

Starting the Alarm Browser

You can start the Java Alarm Browser from Home Base, the Launcher, or the Network Presenter.

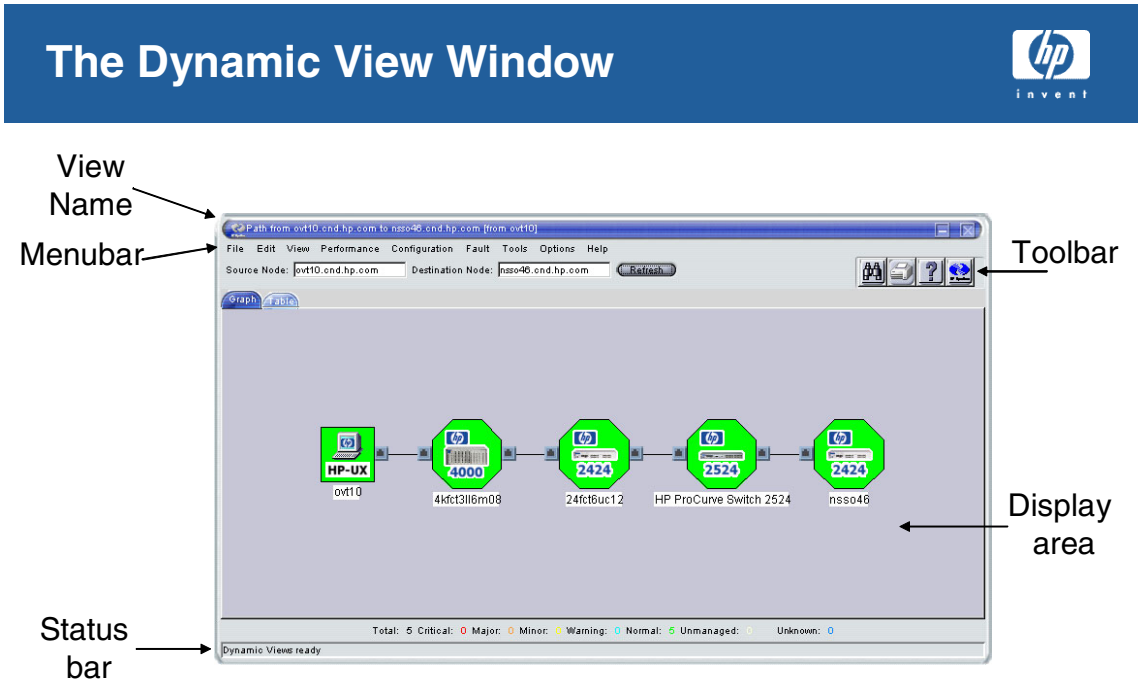
- From Home Base, select the Alarms tab.
- From the Launcher, select the `Information and Reports` tab. `Faults and Alarms -> NNM Alarm Browser`.
- From the `Fault` menu in the Network Presenter window, you can launch the NNM Alarm Browser.

This displays the NNM Alarm Browser window. Selecting `All Alarms` displays the window in the slide.

Using the Alarm Browser

From the Alarm Browser, you can launch a view related to the event using `Actions:Views`. You can view the correlated events from the web interface using the `Actions:Alarm Details` menu item. Select the `Correlations` tab.

The Dynamic View Window



Dynamic views contain a menu bar and tool bar similar to `ovw` and the Network Presenter. You can select a node and make a menu selection to launch a web-based or native application. If no objects are selected that qualify for the application, the menu selection is greyed out. The choices let you test network connectivity, launch new views, or use NNM tools to perform deeper analysis.

For security reasons, some menu choices are available only when the browser is running on the NNM management station, or when the system has been configured to permit remote web access on a per-user basis.

Extended topology menu items are displayed only if Extended Topology is enabled. These include `Options:Extended Topology Configuration`, as well as `Tools:Views->VLAN View`, etc.

Online manuals are available in the Help menu.

You can see what patches and Java versions are running on your system by selecting `Help:About` from a dynamic view.

The Display Area shows the current visualization of your network. Diamond shapes indicate routers. Octagons indicate switches. Squares indicate end nodes.

The Dynamic Views utilize a signed applet, allowing this applet to create files, launch applications, and manage/unmanage devices. When you access a dynamic view, the system prompts you to allow an applet signed by HP to run.

`"Warning - Security"`

```
"Do you want to trust the signed applet distributed by "Hewlett-Packard"? Publisher
authenticity verified by: "VeriSign, Inc.".
```

Click [Grant Always] and you will not be prompted with this dialog again.

A Walk-Through of Dynamic Views

A Walk-through of Dynamic Views

- Access from Home Base or from Alarm Browser
- A family of browser-based views
- Content is dynamically selected
- Node status is dynamically updated
- No running ovw required

- Basic views
  - Internet view
  - Network view
  - Segment view
  - Station view
- Extended Topology views
  - VLAN views
  - OSPF view
- Enhanced views
  - Always available, better data when Extended Topology is enabled.
  - Neighbor view
  - Node view
  - Path view
  - Problem Diagnosis view

U5089S ©2004 Hewlett-Packard Development Company, L.P. 24

Now we'll spend a couple of minutes on each of NNM's "Dynamic Views". In this discussion, the term "Dynamic Views" describes the family of browser-based views, whose content is created as a result of your on-the-spot choices. This dynamic creation of content based on your requests distinguishes these views from other NNM views, not to mention the views you might see in other management products.

The status of nodes in a Dynamic View is kept up-to-date. Once open these views maintain dynamic status colors and are visually flagged when the status changes to help you during troubleshooting.

Here's a quick summary of NNM's views. You can access them through Home Base or the Tools menu. We'll discuss them all in more detail later.

Path View	You can use Path View to show the path between two nodes. The Path view contains a graphical representation of the path.
Neighbor View	A graphical representation of a selected device and connector devices related to it, within a specified number of hops of the selected device. By default, the Neighbor view only shows the selected device and connector devices.

Station View	A graphical representation of the NNM collection stations and management stations discovered in your topology. This view is useful only in a distributed management environment. (See the Guide to Scalability and Distribution for HP OpenView Network Node Manager manual for information about distributed environments.)
Internet View	A graphical representation of the networks in your topology. With the Internet view, you can view the general status of your network and locate problems in your network.
Network View	A table presenting the segments in a specific network. The table contains information about status of the segment and lists nodes that are on that segment.
Segment View	A graphical representation of the nodes on a specific segment in the network. A segment can also be referred to as a node's collision domain or shared media.
Node View	View the nodes that pass a severity and a filter as well as their connectivity. By setting severity to “Minor”, you can have an “Exception View” which behaves like an alarm browser, showing you all the nodes which have at least one down interface. As new nodes are found that pass the filters, they are dynamically added to the display.
Problem Diagnosis View	View historical information about the performance of a critical path in your network. Requires Extended Topology to be enabled.
VLAN View	See a list of the VLANs in your environment and the switches that participate in them. Requires Extended Topology to be enabled.
Overlapping Address Domain View	List the overlapping domains (groups of private IP addresses) and the addresses of devices in each group. See the private IP address and the management IP address of each device. Requires Extended Topology to be enabled.
OSPF View, HSRP View	View specific services and protocols in your environment. Require Extended Topology to be enabled and purchase of NNM SPIs.

Lab Exercises: Introduction

Lab Exercise: Introduction

- Define Terms
- Review current classroom map

©2004 Hewlett-Packard Development Company, L.P.

1. Describe the steps and interfaces involved in management by exception.

2. Name four ways to reduce the number of messages in the Alarm Browser.

3. How does Event Correlation assist NNM in addition to keeping the Alarm Browser uncluttered?

4. Name three protocols that Extended Topology manages. What are the product requirements?

Lab Exercises

Objective: The purpose of this lab is to build your ability to launch a user interface using either the traditional Graphical User Interface (GUI), or the Web-based User Interface.

NOTE: All of the executables mentioned in the lab solutions are in the OpenView binary directory, unless stated otherwise. The OpenView binary directory is:

`\Program Files\HP OpenView\bin` (on Windows)
`/opt/OV/bin` (on UNIX)

1. From Home Base, start the Internet view.

2. Go to the Alarm Browser from Home Base and launch a Neighbor View.

3. Review the current lab environment.

2 Discovering Connectivity with Extended Topology

Module Objectives

Discovering Connectivity with Extended Topology

Version C.00
U5089S Module 2 Slides

At the completion of this module, you will be able to:

- Describe the differences between NNM and Extended Topology discovery.
- Describe the integration between NNM and Extended Topology discovery.
- Describe how information is propagated for unresponsive nodes during discovery.

Comparing NNM and Extended Topology

	NNM	Extended Topology
Discovery	•Continuous discovery of managed devices and layer-3 connectivity •Results are available continuously	•Recurring discovery of layer-2 connectivity and data from managed devices •Results only available when finished
Device Support	•Based on strong layer-3 management standards •Supports any device that implements one of several standard MIBs	•Weaker industry standards for layer-2 management •Device support based on built-in knowledge, product-family similarities, and MIB support

©2004 Hewlett-Packard Development Company, L.P.

You'll find a few differences between NNM discovery and Extended Topology. There are two fundamental reasons for this:

- NNM has a unique “continuous” model of discovery that lets it maintain a current model of topology at all times. This is why you can watch as NNM builds up a topological model before your eyes. On the other hand, Extended Topology, like most network management tools, uses periodic discovery instead. It capitalizes on NNM's strong device discovery, and adds sophisticated connectivity discovery. One side-effect of this model is that the results of Extended Topology's discovery become available when the process completes, and not before.
- NNM is strongly based in industry standards, especially SNMP and widely supported MIBs. However, many modern network devices rely less on standards for management. NNM Extended Topology enters the field with strong support for many popular devices, and with the ability to support yet more devices as the need arises.

NNM discovery means finding additional nodes and devices in the network. Extended Topology discovery means re-querying those same nodes to find their connectivity.

Understanding Recurring Discovery

•Extended Topology captures data about your network as it exists during discovery

- Devices or connections that are down during discovery appear in the Extended Topology data, but information may be incomplete
- Post-discovery changes in topology are not reflected in Extended Topology views until the next discovery

©2004 Hewlett-Packard Development Company, L.P.

The intermittent or occasional nature of Extended Topology discovery (compared to NNM's continuous discovery) has some subtle side effects you should be aware of.

Many things can change in NNM between one Extended Topology discovery and the next, including topology and SNMP configuration. Post-discovery changes in topology (physical or NNM's database) are not dynamically reflected in the data that Extended Topology provides. All of these changes are reflected into Extended Topology data when Extended Topology runs its *next discovery cycle*. Extended Topology discovery should be run as often as necessary to assure that all Extended Topology data is up-to-date.

If Extended Topology gets a device from NNM but can't get SNMP information, it does not try to re-poll it. Information is carried over from a previous discovery if the node previously responded to SNMP queries. Also, if a link was down during the last discovery, the path information in a Node Down alarm may be misleading.

NNM does continuous, rolling discovery and monitoring. You do NNM discovery completely before starting Extended Topology discovery. NOTE: Extended Topology only manages nodes that are in the NNM topology. You must manage all the nodes you care about for Extended Topology with NNM first.

Extended Topology preloads the MIBs for supported devices. You do not need to (and cannot) load additional MIBs for Extended Topology.

Temporarily Unresponsive Nodes

Situations may arise that prevent a previously discovered device from responding with device information during later discovery cycles. If a particular device does not respond to SNMP when queried (for example, due to scheduled maintenance, problem with an intervening connector device, network bandwidth problems resulting in time-outs), it is represented in topology with information from when it had responded in previous discoveries.

NNM has some “memory” of the state of discovery, so that if a node responded in the recent past (with, for example, attributes, connectivity, VLAN and HSRP group associations, etc.), but not in the current discovery, NNM retains detailed information on that node from previous recent discoveries to provide a more complete and accurate topology picture that can improve with time.

A node which consistently doesn't respond to SNMP will no longer have its details represented using the cached data from its most recent response after 8 days (which allows for the device to be deleted if it does not respond to status polls for 7 days).

Post-discovery changes in the network are not dynamically reflected in the data that Extended Topology provides. They instead get incorporated during the next discovery cycle. You should be aware of this behavior, and some of its implications. For example:

- If a device was inaccessible during recent Extended Topology discoveries, but responded with device information during the most recent discovery cycle, a VLAN view will display VLAN information for it.
- If a device became inaccessible during some past Extended Topology discovery, and continued

to be unresponsive during several consecutive discovery cycles, a VLAN view will not display the device or VLAN information about the device.

- A Neighbor view will omit a neighboring device that happened to be inaccessible during several consecutive discovery cycles.

This list is not comprehensive, but gives you a sense of how periodic discovery can affect the data you observe later.

Extended Topology Support Classes

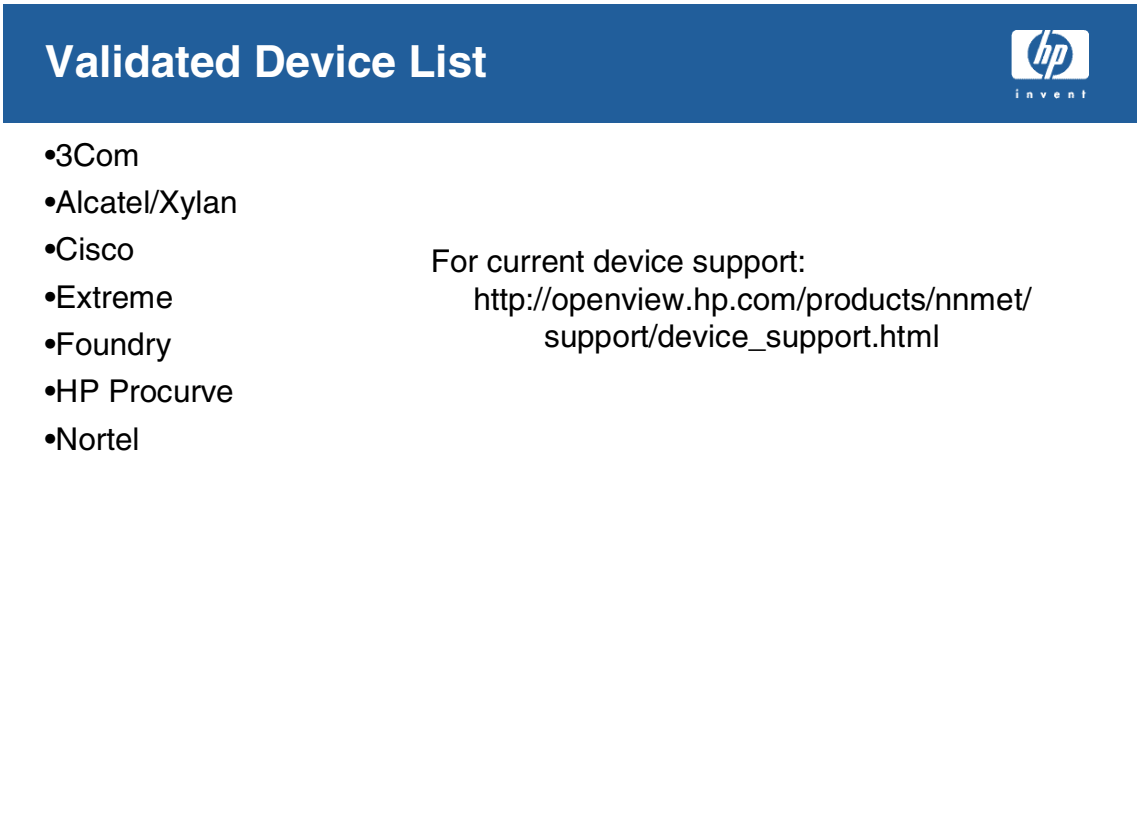
Extended Topology correctly maps and queries a mix of devices, based on both MIB support and built-in intelligence. Extended Topology dives into proprietary MIBs, so HP must certify devices.

As far as Extended Topology is concerned, there are four kinds of devices:

1. Devices which are known to be supported:
   - There is a continually-growing list of devices actually validated by R&D. Refer to the web site.
   - Use defect support process if a defect is exposed
2. Devices which are likely to be handled correctly:
   - Family member of device on validated list, and/or
   - Supports capability expected to work (for example MIB, CDP, etc.)
   - Use enhancement request process if full support is necessary
3. Devices which may or may not be handled correctly:
   - Not enough is known about such a device to determine one way or another
   - Use enhancement request process if full support is necessary
4. Devices which are suspected or known not to be supported:

- Known not to have capability or has failed validation
- Use enhancement request process if full support is necessary

Validated Device List



This table is a partial list of the devices that have been tested with Extended Topology and validated to work correctly.

For the most current information, and for information on how to submit a request for full support of a device, visit the indicated URL.

Device Support Rules

Device Support Rules

- L2 connectivity (non ATM/FR)
- Untagged VLANs
- ATM/ILMI connectivity
- OSPF neighborhood
- Meshes

- Which MIBs to enable on managed devices:
www.openview.hp.com/products/nnmet/support/device_requirements.html

©2004 Hewlett-Packard Development Company, L.P.

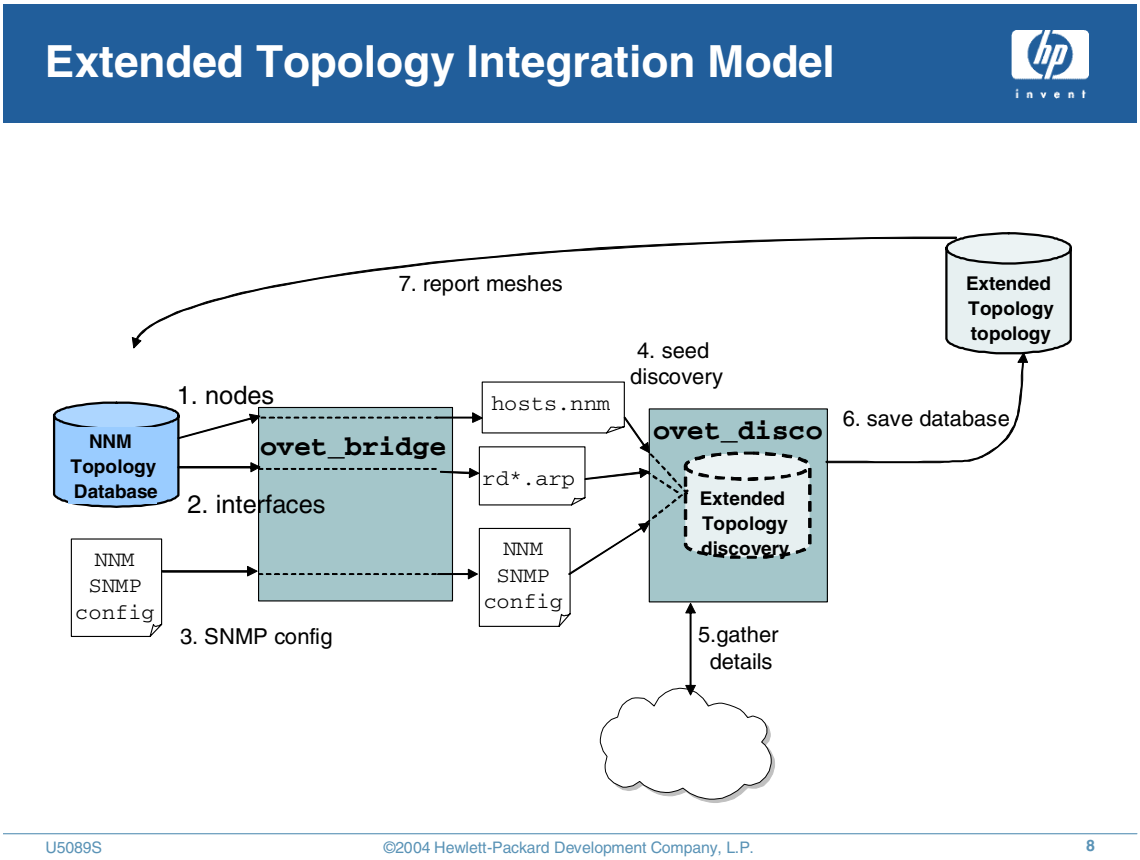
For devices that have not been validated, you can use the rules in the following table to see if it is likely that Extended Topology will handle the device correctly.

Relationship Type	Vendor	MIBs/Protocol support expected
L2 connectivity (non ATM/FR)	Cisco	Standard MIBs for node and interface details, CDP MIB, Cisco Stack MIB, Bridge MIB per VLAN for layer 2 as implemented by Cisco
	Alcatel	Xylan-port, Xylan-vlan MIBs, Standard Bridge MIB (for ageing time), Standard MIBs for node and interface details
	3Com	3Com stack configuration MIB, 3Com VLAN related MIBs (have problems), Standard Bridge MIB, Standard MIBs for node and interface details
	HP	Standard Bridge MIB, standard MIBs for node and interface details, CDP MIB from Cisco
	Others	Standard Bridge MIB, standard MIBs for node and interface details
Untagged VLANs	Cisco	Standard MIBs for node and interface details, CDP MIB, Cisco Stack MIB, Bridge MIB per VLAN for layer 2 as implemented by Cisco
	Alcatel	Xylan-port, Xylan-vlan MIBs, Standard Bridge MIB (for ageing time), Standard MIBs for node and interface details
	3Com	3Com stack configuration MIB, 3Com VLAN related MIBs, Standard Bridge MIB, Standard MIBs for node and interface details
ATM / ILMI connectivity	Cisco	ILMI protocol must be enabled on ATM switches, ATM MIB (RFC 1695)
	Any other vendor	ILMI protocol must be enabled on ATM switches, ATM MIB (RFC 1695)
OSPF Neighborhood	Any vendor	RFC 1850 or RFC 1253 (OSPF MIB)

Relationship Type	Vendor	MIBs/Protocol support expected
Mesh	N/A	Meshes are determined algorithmically based on connectivity data, and are not device specific.

When Extended Topology finds conflicts in the information returned from various MIBs on a device, CDP always wins.

Extended Topology Integration Model



Extended Topology discovery occurs in three major stages:

1. First is the automated discovery by `ovet_disco`, with data seeded from NNM. You can check the overall status of this stage via a web page.
2. Following the initial discovery, there is a continuous process of calculating and validating path information by `ovet_disco`. Extended Topology is also aware of changes in NNM's topology, and can launch a new discovery when a threshold of changes have occurred.
3. If you want to use the OSPF view, you need to run a manually initiated process for OSPF discovery. This process requires some minimal preliminary configuration (seed file).

Detailed Flow

1. `ovet_bridge` downloads nodes from `ovtopmd` and places them into the `hosts.nnm` file.
2. `ovet_bridge` downloads interfaces from `ovtopmd` and creates and IP to MAC mapping file `rd0.arp` to be used by Extended Topology discovery components. (One file per Overlapping Address Domain.)
3. `ovet_bridge` calls the export SNMP configuration script which downloads SNMP community configuration, timeout and retry values, and alternate SNMP ports from NNM and places it in

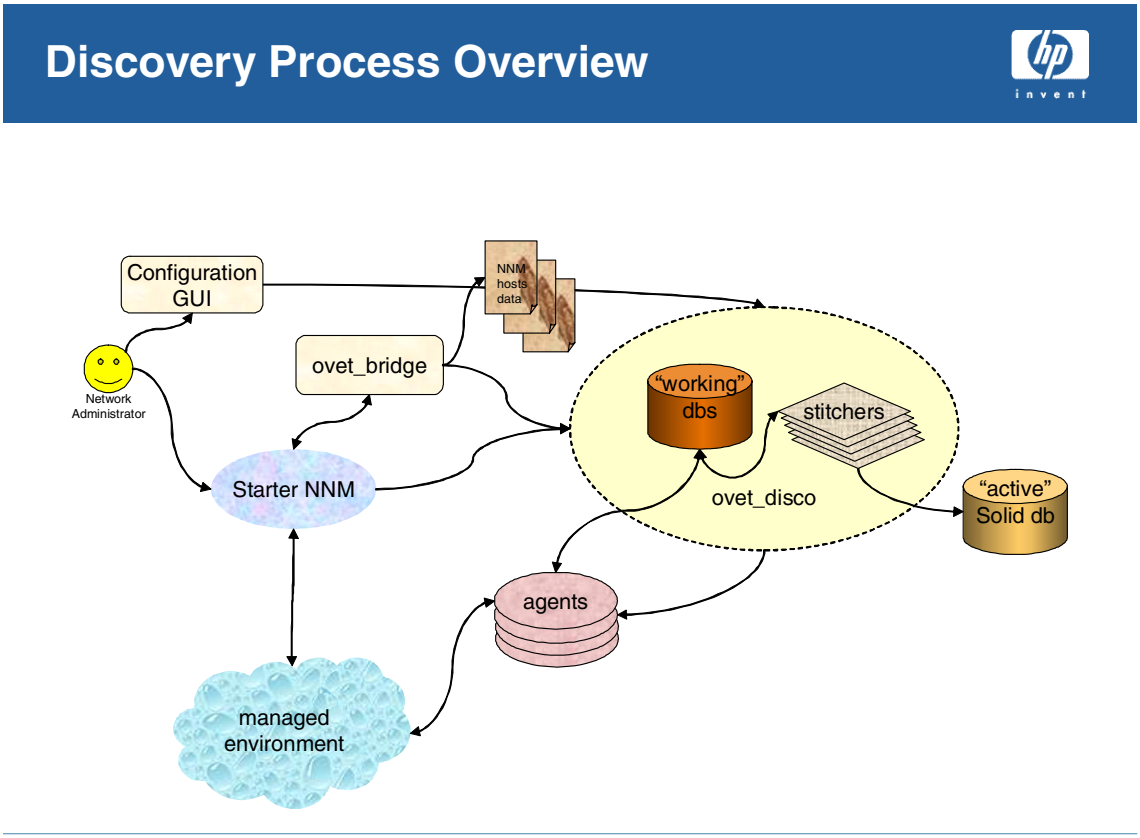
the Extended Topology SNMP configuration file. For more information about NNM and Extended Topology community names, see the man pages for `ovsnmp.conf_db` and `setupExtTopol.ovpl`.

4. The `ovet_disco` component uses the entries in `hosts.nnm` as a seed for finding connectivity. It does not find new nodes.
5. `ovet_disco` discovers connectivity between nodes and sends the nodes, interfaces, VLANS, and connections to the database.
6. `ovet_bridge` which is listening for topology updates finds the NNM object Ids corresponding to the nodes and interfaces and updates the database with the NNM object Ids.
7. The `Path Engine` begins computation of paths and sends them to `netmon`.
8. `ovet_bridge` continues to monitor changes in the network by receiving events from `pmd` and informing `ovet_disco` of every change.
9. When `ovet_disco` sees that it has crossed a threshold number of changes, it informs `ovet_bridge` to re-dump all the files and restart discovery.

What Extended Topology Gathers

In addition to device-specific and protocol-specific information, Extended Topology discovery gathers much of the same standard information that is gathered during `netmon` discovery. This allows the Dynamic Views to present the information when they are running from only the Extended Topology database.

Discovery Process Overview



The discovery subsystem is composed of finders, agents, helpers, stitchers, and an NNM adapter (`ovet_bridge`). DISCO performs discovery on a scheduled, threshold, or on-demand basis, generating tables in the Extended Topology database for the use of views and event analysis.

`ovet_disco` -- is the central point of Extended Topology operation. It is the execution engine for stitchers and stitchers in turn drive `ovet_disco`'s behavior. It dispatches work to agents and collects responses. `ovet_disco` manages internal databases.

finder – process that provides information about the existence of network elements to the rest of the discovery subsystem.

`ovet_agent` -- each Extended Topology agent process has device- or protocol-specific knowledge. Extended Topology agents are responsible for doing the SNMP data collection from the managed nodes dispatched to them by `ovet_disco`, and returning the collected data to `ovet_disco`.

helper – process focused on a specific protocol (e.g., SNMP, ICMP, DNS) that responds to requests from agents to communicate with devices.

stitcher – thread running in `ovet_disco`. Makes topological sense of the gathered SNMP data.

Lab: Extended Topology Discovery

Lab Exercises

•In this lab you will

- Compare netmon and Extended Topology discovery
- Describe Extended Topology discovery

©2004 Hewlett-Packard Development Company, L.P.

Review Questions

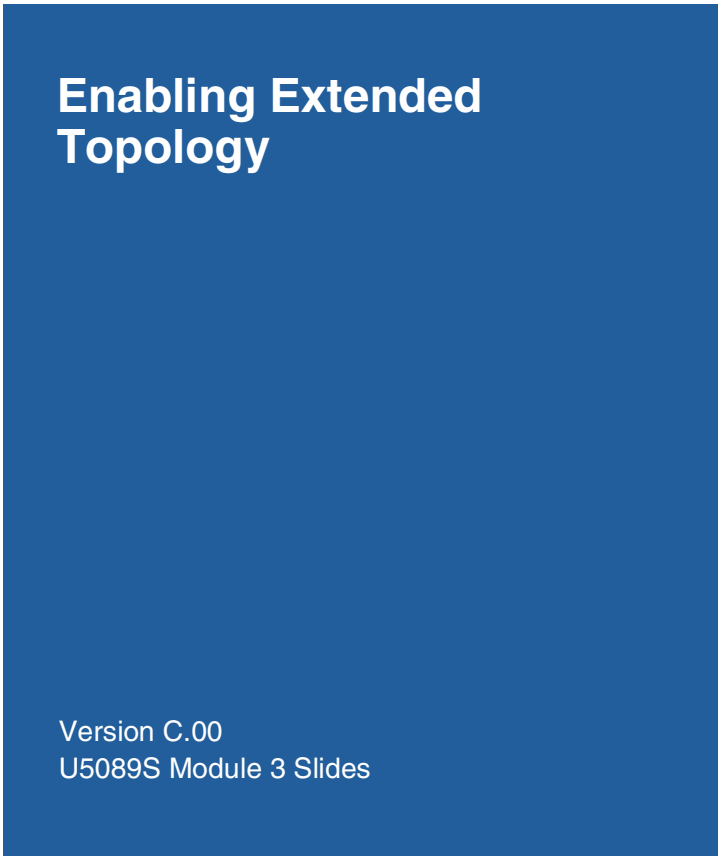
1. Describe the differences between `netmon` discovery and Extended Topology discovery.

2. What processes are involved in Extended Topology discovery?

3. Which file lists the nodes from `netmon` discovery that are passed to Extended Topology discovery?

3 Enabling Extended Topology

Module Objectives



At the completion of this module, you will be able to:

- Activate and verify automatic zone configuration.
- Configure protocols for Extended Topology discovery.
- Update SNMP configuration
- Enable NNM Extended Topology (start discovery)
- Manage NNM Extended Topology processes
  - — Check process status
  - — Configure Extended Topology discovery schedule
  - — Check discovery status
  - — Start an Extended Topology discovery
  - — Stop a running discovery

Before Enabling Extended Topology

- Extended Topology depends on NNM for inventory
- Set up your SNMP configuration.
- Set up discovery and/or topology filters on NNM
- Let NNM finish discovering your network
- Check the results in NNM to assure completeness and SNMP access

©2004 Hewlett-Packard Development Company, L.P.

To ensure that your first Extended Topology discovery is as useful as possible, it's best to have everything ready before starting. The main point to remember is that Extended Topology depends on NNM for an inventory of devices, ARP cache information, and so on.

First, set up your discovery filter to assure that your NNM database contains only the nodes you are interested in. These nodes will be “exported” to Extended Topology for its discovery. If your database is jammed with uninteresting nodes, you could prematurely exhaust your Extended Topology license.

It is best to let NNM's discovery to run to completion. Otherwise, your first Extended Topology discovery will be seeded with an incomplete inventory of your network devices.

Check your results in NNM to ensure that your filtering is correct, and that you have SNMP access to the devices you want to manage.

Make sure you have your SNMP configuration up to date. Extended Topology gets its community strings from NNM.

SNMP Configuration

SNMP Configuration

- Configure community names for Extended Topology to use through **`Options:SNMP Configuration`** menu item.
- To use changes, restart Extended Topology discovery.

U5089S ©2004 Hewlett-Packard Development Company, L.P. 3

Configuring Extended Topology SNMP Access through NNM

Use the `Options:Configuration-->SNMP Configuration` menu to change device community string information in NNM. Extended Topology applies SNMP community string configuration changes during the next discovery cycle.

To apply new SNMP community string changes immediately and run a new Extended Topology discovery, initiate a full discovery through Home Base or run `etrestart.ovpl`.

Starting Extended Topology Analysis

- Run **`setupExtTopo.ovpl`**
  - Checks the system kernel parameters
  - Do you want to discover information about specific protocols?
  - Set your initial Extended Topology configuration (tomcat) user and password
  - Determines the number of nodes NNM is managing
  - In larger environments, offers to automatically partition NNM's IPv4 devices into smaller discovery zones
  - Begins first discovery if possible

U5089S ©2004 Hewlett-Packard Development Company, L.P. 4

NNM Extended Topology is not enabled, turned on, at installation because you need to complete NNM discovery first.

After you install NNM AE, you must run the following script to enable Extended Topology:

- Windows: `%OV_BIN%\setupExtTopo.ovpl`
- UNIX: `$OV_BIN/setupExtTopo.ovpl`

For more information about the `setupExtTopo.ovpl` command, see the *setupExtTopo.ovpl* manpage.

This command initializes Extended Topology as follows:

- It determines the protocols your environment supports and decides what information it needs to discover.
- It checks the system kernel parameters and tells you if you need to make any system modifications.
- It asks you whether you want to discover information about certain protocols.
- It asks you to set up your initial Extended Topology configuration password. It is recommended that you do not use the `root` or `Administrator` password, as that could grant excessive authority to the person who is merely responsible for Extended Topology configuration.

- It determines the number of nodes NNM is managing.
- When possible, Extended Topology immediately proceeds with its first discovery without recommending zone configuration.

  1. It seeds Extended Topology with data from NNM.

     — IP address and hostname information about your network

     — NNM's ARP cache information from your devices

     — NNM's SNMP community string information. For more information about NNM and Extended Topology community names, see the *ovsnmp.conf_db* and *setupExtTopo.ovpl* man pages.

  2. It starts the first Extended Topology discovery process. You see no resulting data in any views until discovery has completed. If a previous discovery has been done, the previous database remains intact until the next discovery completes. There is a brief period of inaccessibility while the database is replaced.

- In larger environments, it can divide the IPv4 devices discovered by NNM into smaller discovery *zones*.

  Zone-based discovery consumes fewer computer resources, resulting in potentially *much* faster network discovery. Zones can be thought of as “islands of connectivity”, which are discovered independently and later brought together through their edge connections.

NOTE Extended Topology uses the information it receives from NNM to calculate the number of nodes it may discover. Extended Topology notifies you when the number of discovered nodes exceeds the number of nodes it is licensed for. Extended Topology may discover additional interfaces after receiving information from NNM, however it does not count these interfaces as discovered nodes for licensing purposes.

Selecting Protocols

You can enable or disable the sets of agents to run based upon what protocols you wish to have discovery for.

With the introduction of IPv6 support in Extended Topology, users wishing to conduct an IPv6 discovery can do so in isolation from the usual IPv4 discovery. This feature allows you, when it is time to enable Extended Topology, to specify what type of discovery to perform, allowing you to include or exclude v6 discovery. By using auto-detection of IPv6 support on your platform, this feature also solves the problem of allowing IPv6 agents to be run on a non-IPv6 platform. This feature also addresses a problem that will grow with the introduction of support for more and more protocols. A customer may not wish to waste time or gather irrelevant data by running discovery agents for a large number of different protocols, when all they may care about are two or three of them. This feature allows customers to customize their discovery to the protocols that they are concerned about. This feature also solves the problem of discovery starting upon setup when the network is too large for a single zone to handle.

The `setupExtTopo.ovpl` script detects if your platform supports IPv6. If so, it gives you a choice to discover IPv4 only or IPv4+IPv6. Otherwise, IPv4 is the only choice, and you are not prompted. If IPv6 is chosen, you are informed that IPv6 configuration (seed file) needs to be done.

You can also enable HSRP with a simple yes / no prompt, without any checks required.

`setupExtTopo.ovpl` prompts for your setup preferences each time you run it. For example, if you say “yes” to HSRP one time, but later run setup and say “no,” HSRP will no longer be enabled. The choices determine which subset of functionality (agents, etc.) gets enabled.

Automatic Zone Partitioning

Automatic zone configuration automatically and intelligently partitions your network into zones, allowing low-hassle or no-hassle zone configuration for large scalability. The algorithms driving the partitioning use a number of factors to produce zones that:

- are non-isolated, well connected, respectful of routed connections between Layer 2 domains.
- are respectful of L2 switch fabric, and connections within a fabric, meaning that nodes in a common L2 domain will, in so far as it is possible, not be split.
- are respectful of dynamically calculated zone size constraints, based on a number of factors such as system memory, number of managed nodes and their interfaces, etc. Zone size is in terms of number of managed nodes and their interfaces.
- minimize unnecessary zone overlap, and hence, duplicate SNMP queries.
- are well distributed and of a performance-conscious size.
- minimizes chance of splitting logical groupings, such as VLANs. (Discovery handles split VLANs, allowing VLANs to span zones, but it is a good idea to avoid that if possible.)
- are consolidated when this would improve performance.

Automatic zone configuration leverages NNM topology data, as Extended Topology data is not available when setting up zones. (In fact, you probably do not have a good Extended Topology discovery when zones are being set up.)

Automatic Zone Configuration

Automatic Zone Configuration

- **`setupExtTopo.ovpl`** runs the automatic zone utility.
- Verify your zones
  1. Start Home Base.
  2. Select Discovery Status.
  3. Click [**`Extended Topology Configuration`**].
  4. Select Discovery Zones tab.
  5. Click [**`Test All Zones`**].
- Initiate discovery from GUI or run **`etrestart.ovpl`**.
- If new nodes are added, rezone from the configuration GUI.

©2004 Hewlett-Packard Development Company, L.P.

When you run `setupExtTopo.ovpl`, Extended Topology determines the need for using discovery zones, and can automatically configure these zones for larger environments. If you choose to have Extended Topology configure these zones for you, make sure you follow the displayed instructions carefully.

If setup detects that zones would be needed in your environment, it asks whether to run the automatic zone script. You are warned that old zones will be overwritten, and told to view the zones in the Extended Topology config GUI.

In large environments, autozoning can take some time. If you do not wish to do this during setup, you can say “no” and configure zones automatically later from the GUI.

Since zone partitioning is not guaranteed to be completed by the time setup is complete, to avoid a race condition, `setupExtTopo.ovpl` does not start discovery in this case. You first check your zones, and then use `etrestart.ovpl` to start a discovery. This also allows you to see any possible warnings produced by the automatic zoning utility, which would recommend minor edits to the zones produced.

NOTE Use the automatic zone configuration feature only after completing a successful NNM discovery.

Use the following procedure to have Extended Topology automatically configure and test your zones prior to running your first discovery:

1. Open the Configure Extended Topology GUI using the NNM `Options`: `Extended Topology Configuration` menu or select the `Discovery Status` tab from `Home Base`, then select [`Extended Topology Configuration`].
2. Select the `Discovery Zones` tab.
3. If you already ran the `setupExtTopo.ovpl` script and asked Extended Topology to automatically configure zones for you, do not reconfigure your zones at this time. However, test your zones prior to starting a discovery or manually altering the zones. If you want Extended Topology to automatically configure zones for you, select [`Configure Zones Automatically`].
4. Click [`Test All Zones`] to test all of the zones, display any warnings, and view any suggested remedies. If necessary, manually reconfigure the new zones to resolve these warning messages.
5. Be sure to select `[Apply]` to activate any manual changes.

Once these zones are successfully configured, select [`Initiate Full Discovery Now`] or run the `etrestart.ovpl` script to start the discovery:

- Windows: `%OV_BIN%\etrestart.ovpl`
- UNIX: `$OV_BIN/etrestart.ovpl`

If you request that Extended Topology configure zones automatically at a later time, you will overwrite any existing zone configuration information.

Verifying and Modifying Your Zone Configuration

You can refine or redefine your zones via the Extended Topology configuration GUI, if you choose, though there should be little or no requirement to do so. Auto-zoning provides a solid starting point.

You can use autozoning at any time to recalculate zones (if you wish to change old zones to represent a change in the network for example). Autozone will look at the current network and determine the new zones. You are provided with feedback while the partitioning is in progress, as well as any feedback you need to know from the partitioning itself. Once the partitioning is complete, the new zones are displayed and you can make any edits if you desire.

If the following changes occur after you configure your zones, you may need to manually adjust your new zones:

- NNM discovers new nodes.
- You manage existing nodes that were previously unmanaged.

Extended Topology could place these nodes in the default zone or in an existing zone if the IP address matches one of the zone's subnet wildcards. Use the following techniques to decide if Extended Topology placed nodes into incorrect zones:

- Check the output for nodes that incorrectly appear in the default zone by using the following procedure:

  1. `autozone.ovpl` creates zones that are output to `$OV_CONF/nnmet/etconfig.xml` (`%OV_CONF%\nnmet\etconfig.xml` on Windows). Since it is a complete set, the old zones are overwritten. You may save the `etconfig.xml` file to a backup location prior to automatic zoning if you wish to be able to return to the previous values.

2. Open the Configure Extended Topology GUI using the NNM `Options`: `Extended Topology Configuration` menu or select the `Discovery Status` tab from `Home Base`, then select `[Extended Topology Configuration]`.
3. Select the `Discovery Zones` tab.
4. Select `[Test All Zones]` and review the displayed information.

- Check your topology for nodes that are missing connections.

After you move devices between or among zones, you should initiate a full Extended Topology discovery. If you add new devices to or delete devices from a single zone, you can save time by initiating an Extended Topology discovery on that single zone.

Manually Starting Extended Topology Discovery

Manually Starting Extended Topology Discovery

•Running **`ovstart`** may launch a new Extended Topology discovery if you configure it

•To launch an Extended Topology discovery at any time

- From Home Base, click [**`Initiate Full Discovery Now`**]
- Run **`etrestart.ovpl`**
- Does not affect NNM

©2004 Hewlett-Packard Development Company, L.P.

- Go to Home Base and complete the following procedure.
  1. Select the `Discovery Status` tab.
  2. Select [`Extended Topology Configuration`].
  3. Select [`Initiate Full Discovery Now`].
- For changes you made to a specific zone, complete the following procedure:
  1. Go to Home Base.
  2. Select the `Discovery Status` tab.
  3. Select [`Extended Topology Configuration`].
  4. Select the [`Discovery Zones`] tab.
  5. Select the zone you modified.
  6. Select [`Discover Zone`].
- Run the `etrestart.ovpl` script. This affects only Extended Topology; NNM continues to function normally. Any existing Extended Topology data remains available until it is replaced at the end of the new discovery.

For more information about the `etrestart.ovpl` command, see the *etrestart.ovpl* manpage. For more information about Extended Topology configuration see the HP OpenView web online help.

- Whenever you shutdown NNM background processes (via `ovstop`), you also shut down Extended Topology processes. When you later restart NNM background processes (via `ovstart`) you also restart Extended Topology processes (and launch a new Extended Topology discovery if configured).

To stop a running Extended Topology discovery, `ovstop ovet_disco`.

Incremental Discovery

•Discover a zone by itself without discovering the whole topology

•Run a full discovery to completion first.

•Use incremental discovery if add or delete nodes in a zone
•Use full rediscovery if move node(s) from zone to zone
•Use full rediscovery if remove a whole zone

•Start zone discovery from Home Base or **`etrestart.ovpl`**

•Updates the connectivity, containment, and redundancy models of a zone

You can discover a zone by itself without discovering the whole topology. Zone discovery is triggered manually from Home Base. All nodes, their contained interfaces and all related objects (VLAN, HSRP, InterfaceContainer) in the zone are deleted and recreated, including multi-zone nodes.

- Reduces the network traffic in the managed environment by rediscovering a single zone rather than the whole network.
- Allows you to manually request to have a single zone rediscovered, where a zone can be OAD, default, any zone or IPV6.

Discovering Devices in a Single Zone

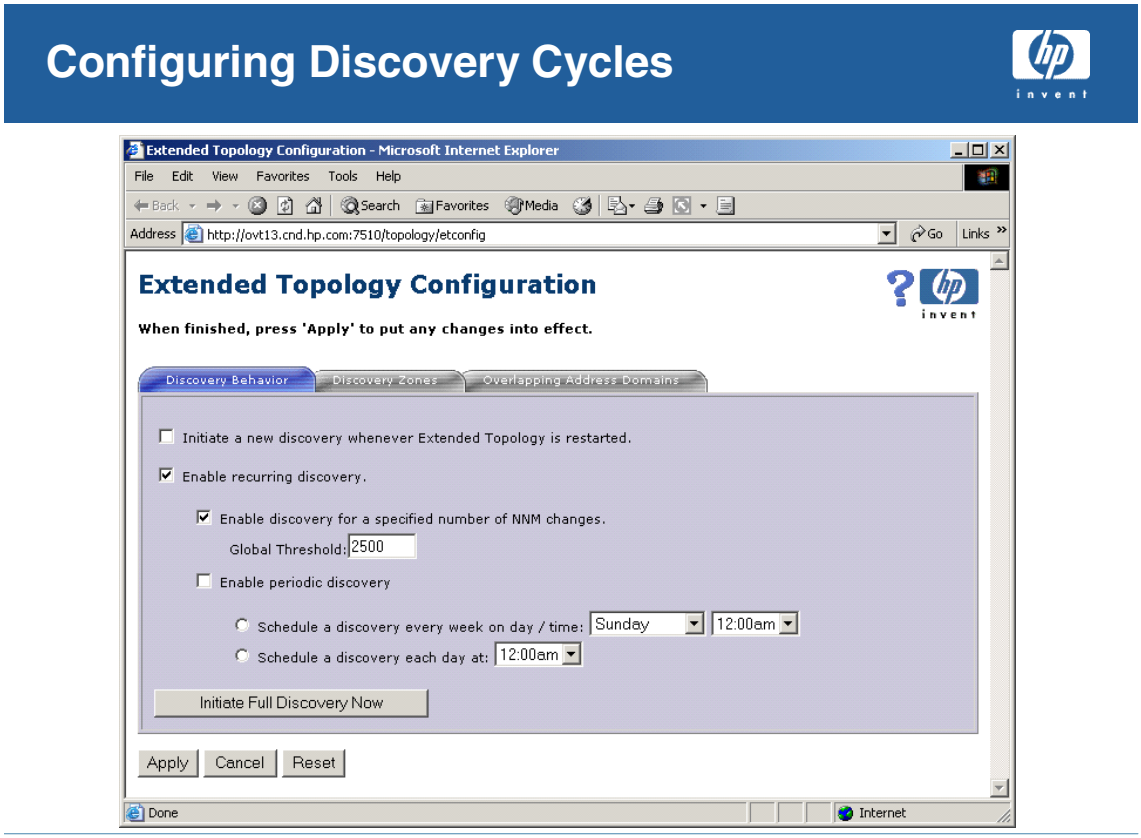
To do this, make sure there are no Extended Topology discoveries in progress, then follow these instructions:

1. From `Home Base`, select the `Discovery Status` tab.
2. Click [`Extended Topology Configuration`].
3. Select the `Discovery Zones` tab. If you configured Overlapping Address Domains, you can also select the `Overlapping Address Domain` tab and select a zone from that view.

4. Select the zone you want to discover.
5. Click [Discover Zone] to initiate a discovery of the selected zone.

After you initiate your discovery, you can monitor the status of your discovery from Home Base.

Configuring Discovery Cycles



In smaller environments, Extended Topology begins discovering network information after you run the `setupExtTopo.ovpl` script. Extended Topology defaults to running a discovery once a default number of NNM topology changes occur.

To modify Extended Topology discovery options, use the NNM menu `Options:Extended Topology Configuration` or select the `Discovery Status` tab from `Home Base`, then select the `[Extended Topology Configuration]` button. You have several configuration options. You can configure Extended Topology discovery behavior if you select the `Discovery Behavior` tab. This allows you to do the following:

- Initiate a new discovery every time Extended Topology is restarted.

NOTE If you configure Extended Topology to initiate a new discovery every time Extended Topology is restarted, then every time you run an `ovstop/ovstart`, it initiates a new Extended Topology discovery.

Use caution when using the `ovstop` and `ovstart` commands, as they restart all NNM and Extended Topology processes. For more information about the `ovstart` command, see the *ovstart* reference page in NNM's online help (or the UNIX manpage).

- Enable or disable Extended Topology recurring discovery.

- Begin a new discovery after NNM detects a number of topology changes. The number of topology changes includes layer 3 discovery information from NNM.
- Schedule a new discovery daily or weekly.

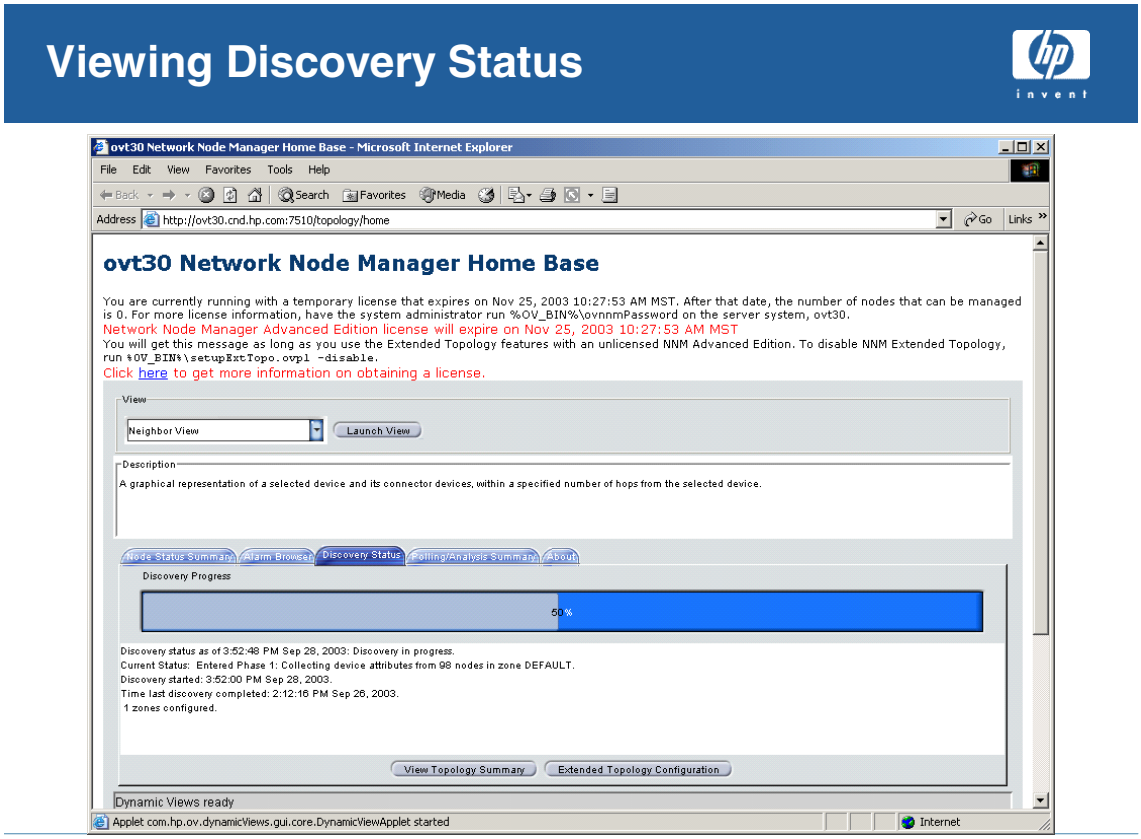
NOTE Extended Topology IPv4 discovery can be configured to monitor the number of changes reported by `netmon` and trigger an Extended Topology IPv4 discovery when a threshold is reached. This does not apply to IPv6. Periodic discovery configuration is crucial to IPv6 accuracy.

- Immediately begin a new discovery by clicking [`Initiate Full Discovery Now`].
- You can initiate discovery of a single zone from the Discovery Zones tab.

Make sure you select [`Apply`] to save your changes.

IMPORTANT When you "Apply" your changes, they are written to the Extended Topology configuration. However, any changes do not take effect until you click [`Initiate Full Discovery Now`] or run (as root or Administrator) `etrestart.ovpl`. That command commits your discovery changes in a secure way, and immediately starts a new discovery.

Viewing Discovery Status



You must wait for Extended Topology to complete an initial discovery before using Extended Topology views. To monitor Extended Topology discovery status, use the following procedure:

1. Point your browser to `Home Base`.
2. Select the `Discovery Status` tab.

To get the same information from the command line, use:

```
ovstatus -v ovet_disco
```

The “Additional Information” field in the resulting output explains which phase discovery is currently in. The message “Awaiting next discovery cycle.” means that the discovery is done. (Any other message indicates that discovery is not done.)

Checking Process Status

You can display the process statuses by executing `ovstatus -c`.

Many Extended Topology processes only run during discovery. After Extended Topology completes its discovery, these processes automatically exit. The following process status output shows the output of the `ovstatus -c` command for those processes:

```
ovet_processname -  NOT_RUNNING    Exited and awaiting next discovery. Exit (0).
```

If you restart Extended Topology processes and you do not have Extended Topology configured to run a new discovery every time you restart it, the ovstatus -c command output looks as follows:

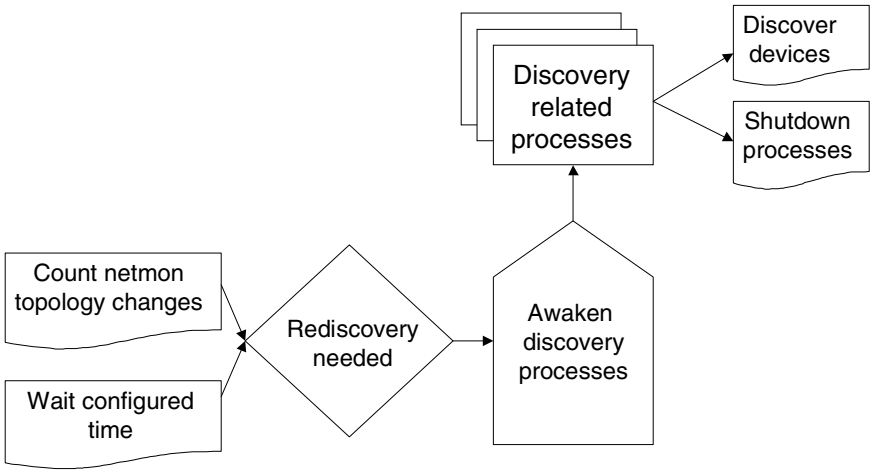
```
ovet_processname 5621  RUNNING        Discovery Completed: date and time of last discovery.
```

The RUNNING portion of the message indicates a state of readiness for discovery processes. A new discovery does not occur until

- Extended Topology meets the discovery configuration parameters you set in the Extended Topology Configuration GUI,
- you select [Initiate Full Discovery Now], or
- you execute etrestart.ovpl.

Sleeping Processes

To enhance scalability and reduce unnecessary resource (memory, etc.) consumption, Extended Topology discovery and related processes bring themselves down after they complete. ovet_disco itself shuts down temporarily so as to reclaim memory, and is then brought back up to stand-by and monitor for the need for rediscovery. Most other discovery-related processes (agents, helpers, etc.) are shut down and brought back only when needed by ovet_disco. Upon ovet_disco detecting the need to rediscover, ovet_bridge is notified and starts the necessary processes at that time.



Lab: Extended Topology Discovery

•In this lab you will

- Enable Extended Topology with **`setupExtTopo.ovpl`**
- Monitor Discover progress from Home Base
- Examine Discovery configuration from Home Base
- Trigger a rediscovery and observe the progress of the key Extended Topology daemons
- Install the Extended Topology demo database
- Explore topology views

©2004 Hewlett-Packard Development Company, L.P.

This lab will introduce you to some of the features in the Extended Topology product. In this lab you will set up Extended Topology, load a static database and explore some of the different views that are part of the Extended Topology capabilities.

Enabling Extended Topology

1. Check the current list of processes.

2. While logged in as root, enable Extended Topology.

3. Start the Extended Topology Discovery Status monitoring interface.

4. Monitor discovery status.

5. Check the current list of processes.

6. Check for new information about the topology.

Review Questions

1. From the Home Base Discovery Status tab, select `[Extended Topology Configuration]`.
2. What options are available for controlling the frequency of discovery?

3. If Extended Topology has already been enabled, how would you go about launching another discovery process?

Installing the Extended Topology Demo

To see additional Extended Topology features, install the Extended Topology demo:

The demo is not complete, but can be used to show all the new views plus launch these views directly from alarms in the browser. This demo should not be installed on a system which needs its NNM or Extended Topology environment to be restored after installing the demo.

HP-UX Installation

1. Due to various types of caching, it is best to eliminate browser temporary files and java caches

after demo installation, prior to bringing up the home base. Run through the demo once to cache as much as possible before bringing in the customer. Have the home base already up.

a. rm -rf ~/.netscape/cache

b. rm –rf ~/.jpi_cache

2. `cp DemoNNM75.tar.gz /tmp`
3. `gunzip /tmp/DemoNNM70.tar.gz`
4. Make a directory to hold the demo files. You must use precisely this directory name.

   a. `tar -xvf /tmp/DemoNNM75.tar`

5. Execute `/opt/OV/contrib/DemoNNM75/bin/setupDemo.ovpl` to run the install script.

   a. Verify all requested information.

   b. Say yes to all protocols and services questions.

To show views, choose connector devices from ovw or dynamic views and launch views. You can also choose events in the alarm browser and launch views directly. There are events for HSRP and OSPF in the All Alarms Browser that can be used to launch those views.

Windows Installation

1. Due to various types of caching, it is best to eliminate browser temporary files and java caches after demo installation, prior to bringing up the home base. Run through the demo once to cache as much as possible before bringing in the customer. Have the home base already up.

   a. Run the Control Panel (Start->[Settings]->[Control Panel])

   b. Double-click “Internet Options”

   c. Select [General] Tab

   d. In the “Temporary Internet Files” section, Select [Delete Files…] button

   e. Select [OK], then [OK] again to get back to the control panel

   f. Double-click “Java Plug-In”

   g. Select [Cache] Tab

   h. Select [Clear] button, respond [Yes] when asked to clear

   i. Close Java Plug-in window and the Control Panel

2. Locate the `DemoNNM75.exe` file.
3. Unzip the demo by double-clicking the executable. Accept the default target directory. Verify that it installs directly under the `\Program Files\HP OpenView\contrib\DemoNNM75` directory (does not create another Demo* directory under it).
4. Double-click `\Program Files\HP OpenView\contrib\DemoNNM75\bin\setupDemo.ovpl` to run the install script.

   a. Verify all requested information.

   b. Accept the default directory.

   c. In the WinZip self-extractor window, be sure to click [`Unzip`], then click [`Close`].

   d. Say yes to all protocols and services questions.

e. Ignore the error about replacing an SNMP entry.

To show views, choose connector devices from ovw or dynamic views and launch views. You can also choose events in the alarm browser and launch views directly. There are events for HSRP and OSPF in the All Alarms Browser that can be used to launch those views.

Exploring Extended Topology Views

1. Exploring some of the Extended Topology Views
   a. Start Home Base.
   b. What Views are available to you?

   c. Launch an Internet View. Select `Tools:Topology Summary`.

      What information is present here?

   d. Close the Topology Summary window and the Internet View.
2. From Home Base, select OSPF View.
   a. How many areas are connected to Area 0.0.0.0?

   b. Click on the All Areas tab. How many areas are defined here?

   c. Expand Area Name 0.0.0.0 in the table. How many area border routers are there?

   d. Go back to the Graph tab and double-click Area 0.0.0.90. Compare this to the listing for the same area in the All Areas table.

3. From Home Base, start the VLAN View. How many devices participate in the WIRELESS VLAN (which is not in an Overlapping Address Domain)? What happens when you double-click on one of the devices?

4. Neighbor Views
   a. From the VLAN view select (single-click) the 10.96.30.2 device in the WIRELESS VLAN.
   b. From the same browser window, select `Tools:Views->Neighbor View`.
   c. By default, how many hops are viewed and are end nodes displayed?

   d. Change the number of hops to 3 and check the box for Include End Nodes. Then click [`Refresh`].

      Note the changes to the display.
5. Right-click in the background of the Neighbor View and select `Highlight VLAN->WIRELESS`. What happens?

Remove the Demo Topology

To facilitate your experience in the rest of the course modules, please remove the demo topology and allow your system to rediscover the classroom.

1. From the demo directory, run the unsetup script.
   - UNIX:
     a. `cd $OV_CONTRIB/DemoNNM70/bin`
     b. `unsetupDemo.ovpl`
     c. Accept all defaults.
     d. `cd $OV_TMP` and run the cleanup script that has been created there.
   - Windows:
     a. In Windows file explorer, browse to *`install_dir`*`\contrib\DemoNNM75\bin`.
     b. Double-click `unsetupDemo.ovpl`.
     c. Accept all defaults.

d. In Windows file explorer, browse to *install_dir*\tmp and run the cleanup script that has been created there.

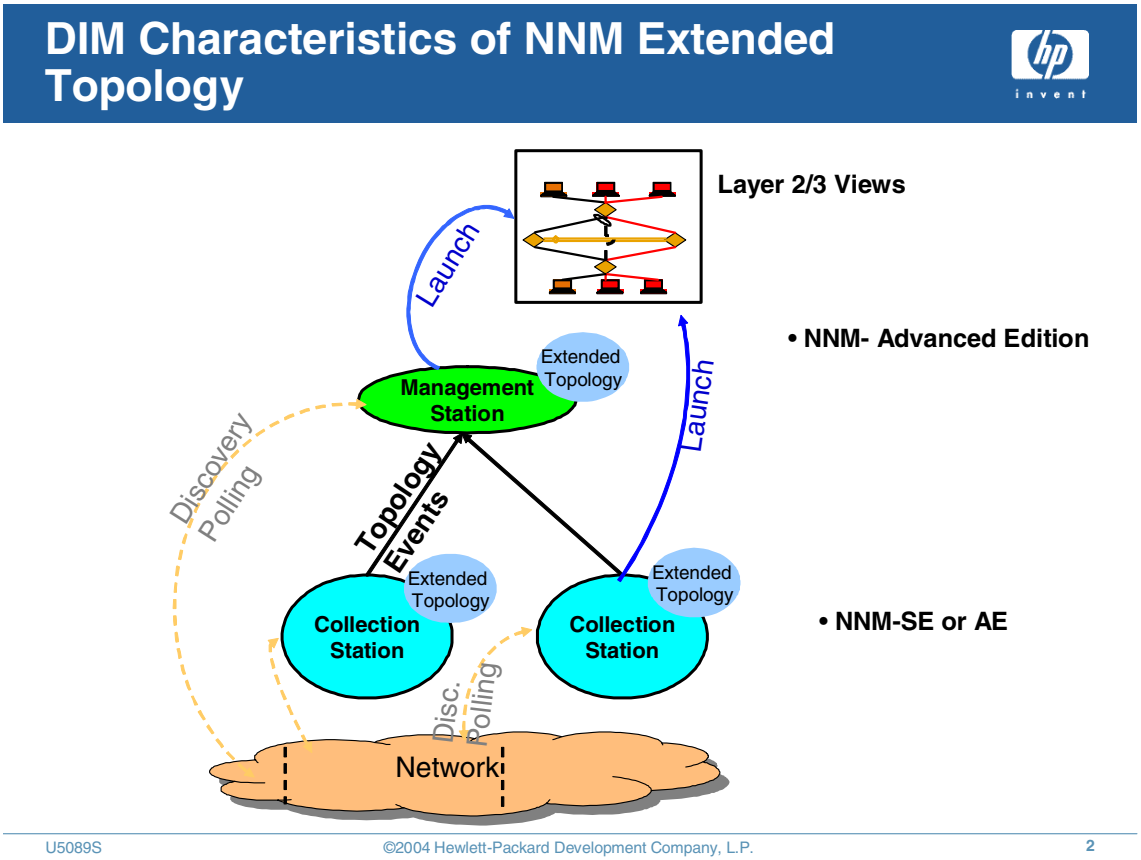
4 Distributing Extended Topology

Module Objectives

Distributing Extended Topology

Version C.00
U5089S Module 4 Slides

DIM Characteristics of Extended Topology

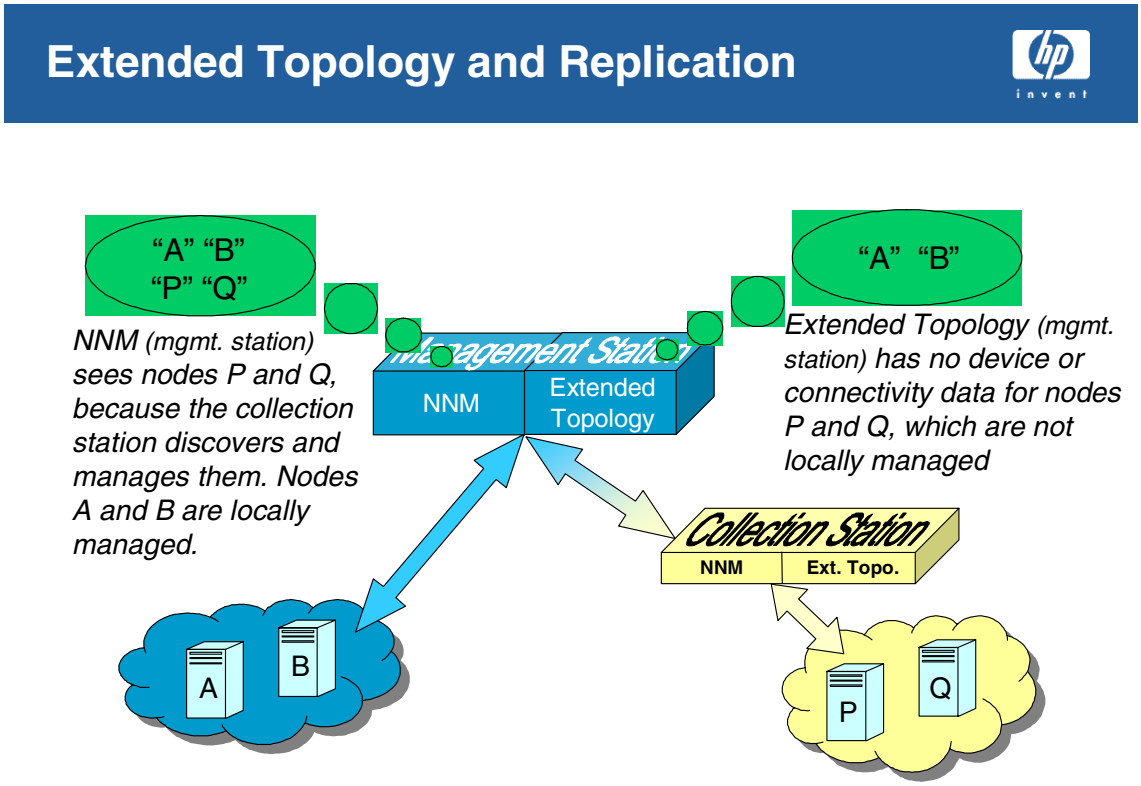


Each management station and collection station requires an appropriate NNM license. Only NNM Advanced Edition can act as a Management Station.

All stations must run the same version of NNM and NNM Extended Topology.

On a Windows platform, you can use your browser to see views from a UNIX host.

Extended Topology and Replication



NNM Extended Topology discovers objects that are *local, primary and in managed state*, in the local NNM database. This is valid for any NNM collection or management station.

Extended Topology does not use replication (`ovrepld`). Nodes that are managed by an NNM collection station are unknown to the local Extended Topology. (This model avoids redundant polling in the present architecture.)

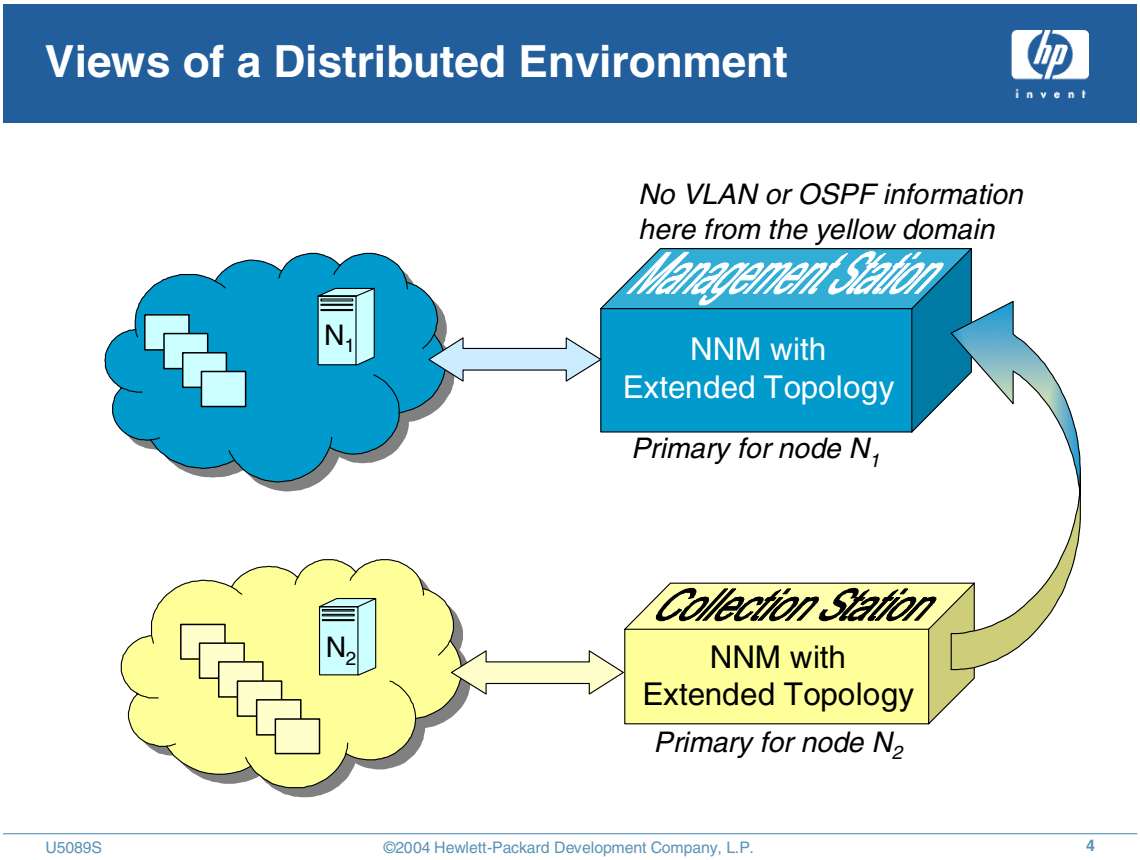
In this diagram, the management station makes use of the collection station in a distributed NNM environment. NNM at the management station can see and integrate nodes in the collection station domain, but Extended Topology at the management station can not.

Extended Topology derives certain information from local objects (such as VLAN and layer-2 connectivity data). It can not derive such information from objects that are managed by a collection station.

TIP ECS may show different behavior of ConnectorDown on the Collection Station and Management Station. On the management station, you may have less accurate analysis because the Extended Topology information is not propagated.

Suggested alternative: Use OVO on the management station, not replicated topology. Pull the raw events and correlate them. You would not get the integrated topologies from multiple collection stations, but you would get all the benefits of OVO.

Views in a DIM Environment



If you are working on a management station without Extended Topology information about a node (for example, a secondary object), point your browser to the collection station owning this object.

First, we'll see how the Neighbor, Path, and Node Views work in a distributed NNM environment.

In this diagram, the NNM topology at the management station includes and shows node N2 from the collection station domain. This is because the collection station monitors the domain and sends the topology to the management station. But the management station does not have Extended Topology data about node N2.

Suppose you select node N2 on an NNM map at the management station, and request a Neighbor View from the NNM `Tools:View` menu. The result is the best Neighbor View that the management station's NNM can create for node N2 *without Extended Topology data*.

To see a view which includes Extended Topology data, browse to the collection station's Home Base (`http://collection_station:7510`) for the local (collection station) Neighbor View of node N2. This lets you get the "high-level" view (from the management station), or drill-down to the "local view" (at the collection station).

This same model holds true for Path View and Node View as well.

To determine the collection station responsible for a device, use the comand `ovtopodump -SC` *`devicename`*. Secondary collection stations are indicated with an ampersand (&) and the primary collection station has no indicator.

Extended Topology-based Views

Next, we'll cover the views that come from the Extended Topology database (VLAN, OSPF, HSRP, etc.) in regard to a distributed environment. We'll talk about VLANs first. The most basic point is that *VLAN information is not communicated across the distributed environment.*

As before, in this diagram the NNM topology at the management station includes and shows switch S2 from the collection station domain. This is because the collection station monitors the domain and sends the topology to the management station. But again, the management station does not have Extended Topology data about switch S2.

Suppose you request the VLANs view from the management station. Because it has no Extended Topology data from the collection station domain (meaning no VLAN data about switch S2), the VLANs view available from the management station *includes nothing about switch S2*.

If you need to know the VLAN information about switch S2, you have to open the VLANs view *from the collection station* Home Base (that is, `http://collection_station:7510`) or via the `ovw` menus at the collection station.

The same model holds true for OSPF, Overlapping Address Domain, HSRP, and IPv6 views also. That is, OSPF information is not communicated across the distributed environment, and must be viewed from each NNM installation independently.

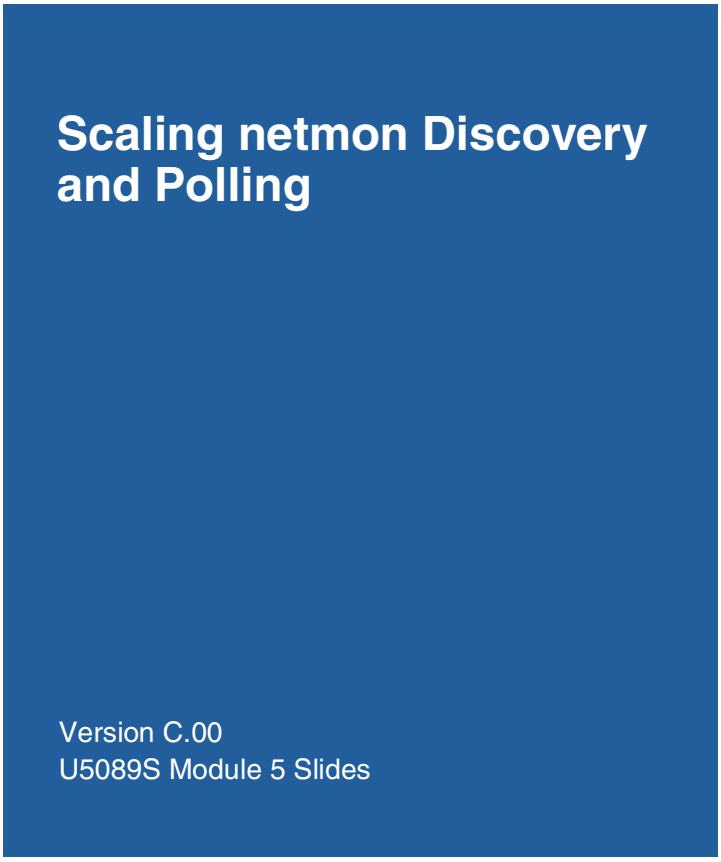
Lab Exercises

Lab Exercises

©2004 Hewlett-Packard Development Company, L.P.

5 Scaling netmon Discovery and Polling

Module Objectives



At the completion of this module the student will be able to:

- Automatically limit the interfaces managed by `netmon`.
- Improve NNM performance by limiting forward and reverse DNS lookups throughout NNM.

Scalability of netmon Status Polling

Scalability of netmon Status Polling

- NNM's status polling is dependent on the number of managed interfaces.
- Excluding unnecessary interfaces based on their properties will allow NNM to poll a much larger number of devices.
- Being selective about devices and interfaces in netmon discovery allows more Extended Topology management.

©2004 Hewlett-Packard Development Company, L.P.

In extremely large environments, you'll want to be very selective about the devices that are discovered and polled by NNM in order to achieve your scalability goals. NNM provides several tools to help you automate the selectivity.

Controlling the Set of Managed Interfaces

Controlling the Set of Managed Interfaces

- In access networks
- In networks with backup dialup links
- On high port density switches

- By default, NNM manages all interfaces
- Interfaces can be unmanaged manually using **xnmtopoconf**.

```
xnmtopoconf -unmanage MS_name filter_name
```

U5089S ©2004 Hewlett-Packard Development Company, L.P. 3

Managing Objects Such as Nodes

A managed object is one that is actively being polled by NNM to determine its status and configuration. An unmanaged object is one that exists in the databases and within maps, but is not being polled by NNM. NNM gives you the choice of managing or not managing objects, depending on your information needs and network resources. When an object is managed, NNM can obtain any information that you specify about that object (as long as that object's protocol is supported).

The more objects you manage, the more memory and disk space is required on the management station. In addition to the information that you specify, the management station will need processing power for the routine status and configuration polling and the event monitoring of each managed object.

If an object is critical to the network function, then you should manage it. If the object is not critical to the functioning of your network, you might choose to unmanage it; which means that NNM will not actively monitor the object. When an object is unmanaged, less memory and disk space and less processing time is required on the management station. However, you only get minimal information about that object on your map: its placement on the network and its IP/IPX address (static placement). NNM can still receive traps and post alarms in the Alarm Browser for

unmanaged objects.

```
xnmtopoconf -manageobj | unmanageobj  station objlist
```

```
Manage or unmanage the objects in objlist associated with station, where station is the name of
the local management station. Objects may only be managed or unmanaged on the local management
station. The objlist may be a filter.
```

Unmanaging Interfaces

In addition to unmanaging entire nodes, you may want to unmanage selected interfaces on devices. These may be interfaces which are not connected, whose status changes frequently, or for which you are not responsible.

Unmanaging interfaces reduces CPU and bandwidth consumption.

Unmanaged interfaces do not contribute to the status calculation of the node.

Automatically Unmanaging Interfaces

Automatically Unmanaging Interfaces

- Batch auto-unmanage tool to automatically stop monitoring interfaces that do not need to be managed, based on user-defined criteria.
- Run periodically (using cron or Scheduler) to keep the set of managed interfaces focused.
- Lower total cost of ownership (TCO):
  - Larger networks managed by single station
  - Lower network bandwidth used for management tasks
  - Don't need to unmanage by hand
  - Less noise

©2004 Hewlett-Packard Development Company, L.P.

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

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

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

Old Status	Passes Filter	Fails Filter
Managed	Unmanaged	Managed
Unmanaged	Unmanaged	Managed, status Unknown

You cannot limit NNM discovery with the `ovautoifmgr` command. For information about limiting NNM discovery, use discovery filters.

After all of the rules have been processed, `ovautoifmgr` will terminate. This is not a continuously-running process. It requires that `ovtopmd` be running in order to do its work.

`ovautoifmgr` was designed to be run periodically by the `cron` facility on UNIX or Windows scheduler. See the examples in the *ovautoifmgr* man page (reference page on Windows).

Configuring Interface Unmanagement

Configuring Interface Unmanagement

- Three filter expressions:
  - Device of interest : Filter based on properties of the device (SNMP cabable, isBridge, …)
  - Interfaces of interest on device (type, bound IP address)
  - Peer device (connected to interface)
    - Peer device is only considered for interfaces that are directly connected (i.e., not part of shared media segment with multiple devices).
- Configure in **ovautoifmgr.conf**

"Dev_Filt" "Interface_Filt" [|| | && "Peer_Filt"[0]]

- Run the command **ovautoifmgr**.
  - Log your changes using the –v options to document with interfaces were changed.

 ©2004 Hewlett-Packard Development Company, L.P.

1. Create a filter in the `filters` file.
2. Create or edit the configuration file to set up your rules.

   UNIX: `$OV_CONF/$LANG/ovautoifmgr.conf`

   Windows: `%OV_CONF%\C\ovautoifmgr.conf`

   Each rule is defined by two or three filters.

   ```
   qnFilter ifFilter [ conj enFilter [ 0 ] ]
   ```

`qnFilter`	This is the Qualifying Node Filter. Only nodes that pass this filter will have their interfaces examined according to the remainder of this rule.
`ifFilter`	This is the Interface Filter. Each interface on the qualifying node will be examined by this filter to determine whether the interface should be in the managed or unmanaged state. If the interface passes the filter, it will be a candidate for unmanaging.
`enFilter`	This is the End Node Filter. If there is only one node connected to the candidate interface in the topology database, then this filter is applied to that connected node. If there is more than one node connected to the candidate interface, then the filter will be considered to have failed.
`conj`	This is a conjunction which must be either `||` or && and is only used when

there is an enFilter specified. If conj is ||, then the interface will be a candidate for unmanaging if either ifFilter or enFilter (or both) pass. If conj is &&, then the interface will only be a candidate for unmanaging if both ifFilter and enFilter pass.

0 — This is the zero flag. It can only be used when an enFilter is specified. When the zero flag is specified, then a candidate interface that shows no connected end node will be treated as if there was one end node connected which passes enFilter.

ovautoifmgr will apply each rule in the order in which it is found in ovautoifmgr.conf. Each managed node in the topology database is examined to see if it passes the *qnFilter* for the rule. If it does, that node's interfaces are examined according to the Interface Qualification portion of that rule.

Each interface passing the qualifying criteria is a candidate for being unmanaged. If the interface has a status of UNMANAGED, nothing changes. Otherwise, its status is changed to UNMANAGED.

Each interface that does not pass the qualifying criteria is a candidate for being managed. If the interface has a status of UNMANAGED, it is changed to a status of UNKNOWN. Otherwise, nothing changes. Being in a managed state allows netmon to monitor that interface's operational status and generate events.

Once a node passes the *qnFilter* it is no longer eligible for any subsequent rules in the configuration file. Thus, if a node passes more than one *qnFilter*, only the first one in the file applies to that node. Also, all unmanaged nodes are ignored by ovautoifmgr.

If there is a syntactical error found within one of the rules of ovautoifmgr.conf, or if a rule specifies a filter that can not be found, then ovautoifmgr will output an error message (and log it in the log file) and processing continues with the next valid rule. The faulty rule is completely ignored, and any nodes that would have passed its *qnFilter* will be eligible to be processed by later rules.

The *ifFilter* is treated somewhat differently from the other filters in the rule. Filters are normally limited to a subset of ovwdb fields as documented in *HP OpenView - A Guide to Scalability and Distribution for Network Node Manager*. However, the *ifFilter* allows any ovwdb field to be specified.

3. Run ovautoifmgr to set interfaces to be unmanaged.
4. Verify the results by examining the nodes' interface status using ovtopodump -v or through the GUI.

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

Filter Examples

Filter Examples

```
Sets {
        serverNodes "a list of important servers" {"server1.company.com",
"server2.company.com" }
}
Filters {
        Switches "Any bridge or switch" { isBridge }
        Repeaters "Any multi-port repeater" { isHub }
        Routers "Any Router" { isRouter }
        ServersSet "Any designated Server node"
              { "IP Hostname" in serverNodes }
        NonIPInterface "Any interface not supporting an IP address"
              { isInterface && !isIP }
}
FilterExpressions {
        UnimportantNodes "nodes not a network device or server"
              { !Routers && !Repeaters && !Switches && !ServersSet }
}
```

Unmanage switch interfaces connected to unimportant nodes or unconnected:

```
Switches NonIPInterface && UnimportantNodes 0
```

Which nodes — *Which Interfaces* — *Which end nodes* — *Unconnected*

©2004 Hewlett-Packard Development Company, L.P.

Additional examples can be found in the *ovautoifmgr.conf (4)* man page (reference page on Windows).

```
Sets {
        serverNodes "a list of important servers"
              {"server1.company.com",
               "server2.company.com" }
}
Filters {
        Switches "Any bridge or switch" { isBridge }
        Repeaters "Any multi-port repeater" { isHub }
        Routers "Any Router" { isRouter }
        ServersSet "Any designated Server node"
              { "IP Hostname" in serverNodes }
        NonIPInterface "Any interface not supporting an IP address"
              { isInterface && !isIP }
        DialupISDN "Interface supporting individual ISDN dialup connections"
              { isInterface && "SNMP ifType" == "Basic ISDN" }
        ChicagoSwitches "all switches in the Chicago office"
              { isBridge && "IP Hostname" ~ "^chisw" }
        endNodePorts "switch ports only used for end nodes in Chicago office"
              { ( !isIP ) &&
                "SNMP ifName" != "1/1" &&
                "SNMP ifName" != "1/2" &&
                "SNMP ifName" != "2/1" &&
                "SNMP ifName" != "2/2" }
```

```
}
FilterExpressions {
        UnimportantNodes "nodes not a network device or server"
              { !Routers && !Repeaters && !Switches && !ServersSet }
}
```

The following rule will cause `ovautoifmgr` to examine all switch devices and unmanage any interfaces which are connected to unimportant nodes or not connected at all. In this case, an unimportant node is considered to be a node that is not a connector (switch, repeater, router, etc.) and is not listed in a list of important server machines. Also notice that any interface on the switch that has an IP address will remain managed.

```
Switches NonIPInterface && UnimportantNodes 0
```

A slight variation of this rule will cause switch interfaces that do not appear to be connected to remain managed, while all other interfaces will be treated the same as above.

```
Switches NonIPInterface && UnimportantNodes
```

Handling DNS Problems

Handling DNS Problems

- DNS lookups are blocking, and can cause significant slowdown.

Solutions:

- HP recommends running a caching-only secondary DNS server on the NNM station.
- NNM has tools to help you understand DNS problems.
- You can reduce slowdowns due to DNS lookup failures (reverse lookups) by populating the **`ipNoLookup.conf`** file.
- NNM maintains a no-lookup cache that is used to prevent forward name lookups for things that are not IP hostnames.

©2004 Hewlett-Packard Development Company, L.P.

Recommendations for tracking and improving DNS performance are available in the NNM whitepapers directory, *DNSImprovements.pdf*.

After a node is discovered and its configuration gathered via SNMP, it must be given its hostname. The hostname is derived by an inverse address lookup of an interface via the local host file or the Domain Name System (DNS) in the following order: lowest numbered IP address on the node, or SysName via SNMP. If NNM cannot retrieve a name via the `hosts` file or the DNS it will use the SNMP `system.SysName`. This brings us to the importance of the accuracy and proper configuration of the `hosts` file, and DNS. They must both resolve to the same fully qualified hostname. If a valid hostname cannot be found using any of these methods, NNM will use the IP address as the hostname.

Refer to the `DNS and OpenView` whitepaper in the NNM whitepapers directory for information about configuring DNS and hosts files.

Testing DNS Resolution

- **resolveNames.ovpl**
- Whether the lookup succeeds or fails
- Can send output to a file for use by other tools

©2004 Hewlett-Packard Development Company, L.P.

`resolveNames.ovpl` extracts hostnames from the output of either `ovtopodump` (default) or `ovdumpevents` ("-e" option) and attempts to resolve the hostname to an IP address using the `gethost` executable.

The output of the script is a list of names that could not be resolved to an IP address using the system IP name service. This script only differentiates between names that were successfully resolved to IP addresses and names that were not resolved to an IP address. The script does not consider the amount of time required to resolve the name to an IP address.

The `resolveNames.ovpl` tool is found in:

UNIX: `/opt/OV/support`

Windows: *`install_dir`*`\support`

The following is the usage statement for `resolveNames.ovpl`

```
Usage: resolveNames.ovpl [-e] [-f <file>] [-i] [-n]
[-o ux10|ux11|sol|nt] [-v] [-?]
-e : get names from ovdumpevents (default is ovtopodump)
-f : copy output to file
-i : ignore IP objects (not valid with -e option)
-n : check network names (not valid with -e option)
```

```
-o : specify the os version
-v : verbose output
-? : print this message
```

Restricting Forward Lookups

Restricting Forward Lookups

- NNM maintains a cache of names that should not be resolved by the IP name service using forward lookups:
  - Segment names
  - Names derived from link layer addresses
  - Hostnames based on IPX addresses (Windows)

- You can control the entries in the no lookup cache using the **snmpnolookupconf** command:

```
snmpnolookupconf [-add|-load|-disable] <parameter>
```

U5089S ©2004 Hewlett-Packard Development Company, L.P. 9

There are some nodes whose name should not, or cannot, be resolved to an IP address using the system IP name service. NNM maintains an internal cache to determine if a hostname should be resolved. During the discovery process, `netmon` populates the No Lookup cache with names that are known to not be resolvable to an IP address using only the above criteria. netmon does not add names to the No Lookup cache because a name could not be resolved to an IP address. In addition, you may add entries to the cache.

You may enable DNS tracing using the directions in the *DNS Improvements* whitepaper to determine which lookups are failing (and causing delays in NNM) in your environment. You may also use the tool `resolveNames.ovpl` to check the devices in the topology database or event database for their resolution.

SYNOPSIS

```
snmpnolookupconf  -a[dd] [-fullS[egName]] hostname

snmpnolookupconf  -clearC[ache]

snmpnolookupconf  -d[isable] [-fullS[egName]] hostname

snmpnolookupconf  -dumpC[ache]

snmpnolookupconf  -l[oad] [-v[erbose]] [-fullS[egName]] filename

snmpnolookupconf  -t[est] hostname
```

`snmpnolookupconf` maintains the SNMP No Lookup cache. The SNMP No Lookup cache

determines which hostnames the NNM processes are not allowed to resolve to IP addresses. The NNM processes use the system IP name server try to resolve hostnames to IP addresses. The SNMP No Lookup cache stores the hostname of objects that cannot be resolved to an IP address.

Before NNM processes attempt to resolve a hostname to an IP address they first consult the SNMP No Lookup cache to determine if this lookup should be permitted. If the lookup is not permitted then the NNM process continues as if the lookup failed. By default all entries in the No Lookup cache are enabled (meaning that a DNS lookup on this hostname is not permitted). Hostnames in the SNMP No Lookup cache can be disabled (meaning that a DNS lookup is permitted on this hostname). If an entry is disabled in the SNMP No Lookup Cache it can only be re-enabled by the user. NNM processes are not allowed to overwrite a disabled entry and thus re-enable the entry.

By default when the name of segment is added to the SNMP No Lookup cache it is truncated to remove the trailing segment ID. For example, if 192.168.Segment1 was added to the SNMP No Lookup cache it would be truncated and the entry in the cache would be 192.168.Segment. The truncated segment name matches all entries of the form 192.168.Segmentxxxxx, where x is a number between 0 and 9. Segment names are truncated to minimize the size of SNMP No Lookup cache.

Entries in the SNMP No Lookup cache are case sensitive. Entries remain in the cache permanently.

Parameters

`snmpnolookupconf` supports the following options:

NOTE The options to `snmpnolookupconf` may be abbreviated as shown in the SYNOPSIS section. For example: `-add` can be abbreviated `-a`.

`-add [-fullSegName] hostname` Add hostname to the SNMP No Lookup cache. This entry will be enabled (meaning a DNS lookup is not permitted on the hostname). If the entry already exists and it is disabled, the entry will be enabled. If the -fullSegName option is specified and hostname is the name of a segment then the segment name will not be truncated to remove the segment ID.

`-clearCache` Delete all entries from the SNMP No Lookup Cache.

`-disable [-fullSegName] hostname` Disable hostname in SNMP No Lookup Cache. If the entry does not exist in the SNMP No Lookup Cache then the entry is added and then disabled. When an entry is disabled NNM processes are permitted to attempt to resolve the hostname to an IP address using the system IP name server. If the -fullSegName option is specified and hostname is the name of a segment then the segment name will not be truncated to remove the segment ID.

`-dumpCache` Display the contents of the SNMP No Lookup Cache. One entry is displayed per line. If the output of this option is captured it can be used to reload the SNMP No Lookup Cache using the -load option.

`-load [-verbose] [-fullSegName] filename` Load the SNMP No Lookup Cache with contents of the file specified by filename. The file specified by filename should contain one entry per line. By default each entry is enabled in the SNMP No Lookup Cache unless the entry is preceded by <DISABLED>. If the `-fullSegName` option is specified then segment names will not be truncated to remove the segment ID before adding the segment name to the cache.

`-test hostname` Test hostname to see if NNM processes are permitted to resolve hostname to an IP address using the system IP name server.

Restricting NNM's Reverse Name Lookups

Restricting NNM's Reverse Name Lookups

- **`ipNoLookup.conf`** file located in

  UNIX: $OV_CONF

  Windows: %OV_CONF%

- Designate a set of IP addresses or ranges that should NOT be looked up for resolution to host names (such as using DNS PTR records) by NNM.
  - If a part of your managed address space does not have DNS entries, you should create this file to stop NNM from attempting to resolve such addresses.
  - Each line contains an IP address range specification such as:
    - 10.1.*.*
    - 192.168.123.100-200
- Stop and restart NNM processes

©2004 Hewlett-Packard Development Company, L.P.

The `ipNoLookup.conf` file is used by all NNM processes to determine if an IP address should be resolved to a hostname using the system IP name server. NNM processes will attempt to match an IP address against each entry in this file before attempting to resolve an IP address to a hostname. If a match is found then the process will not attempt to resolve the IP address to a hostname using the system IP name server.

Lines in the file contain one IP address or IP wildcard per line (see the *netmon.noDiscover* reference page for rules regarding IP wildcarding). Each entry must be on a single line. Comments are denoted by a number sign (#), and cause the remainder of the line to be ignored. Blank lines are allowed.

Use the `ipNoLookup.conf` file when you determine that a specific IP address (or range of IP addresses) cannot be resolved to a hostname using the system's IP name server.

Note that the `ipNoLookup.conf` file must created by an administrator. It does not exist by default.

If the `ipNoLookup.conf` file is modified while NNM processes are running these processes must be stopped and restarted for the changes to take affect.

The following is an example of a `ipNoLookup.conf` file.

```
# A single IP Address

192.168.1.100
```

```
# An IP Wildcard
10.*.*.*
# An IP Wildcard Range
192.168.1.101-255
```

Lab Exercises

Lab Exercises

- Identify proper use of tools
- Review current name resolution

©2004 Hewlett-Packard Development Company, L.P.

1. How could you use `xnmtopoconf` to unmanage all end nodes currently discovered?

2. How could you use `xnmtopoconf` to continuously unmanage all end nodes that get discovered?

3. How could you use `ovautoifmgr` to continuously unmanage all end nodes that get discovered?

4. Which tool would you use to unmanage all switch ports connected to end nodes? Why?

5. Run `resolveNames.ovpl` to see whether any systems in your current topology do not resolve.

6. Review the list of names not currently looked up with the command `snmpnolookupconf -dumpCache`.

6 Controlling Extended Topology Discovery

Module Objectives

At the completion of this module, you will be able to:

- Configure Extended Topology discovery zones.
- Limit the devices passed from NNM topology to Extended Topology based on IP address or filters.
- Modify zone configurations.
- Discover a single zone.
- Determine which nodes did not respond to SNMP during Extended Topology discovery.
- Review Extended Topology data using `ovet_topodump.ovpl`.

Limiting Extended Topology Discovery

Limiting Discovery with the Extended Topology Discovery Exclusion List

- Extended Topology Discovery can be limited by contents of the **`bridge.noDiscover`** file located in
  - UNIX: **`$OV_CONF/nnmet/`**
  - Windows: **`%OV_CONF%/nnmet/`**

- To create the **`bridge.noDiscover`** file:
  1. As root, create the file **`bridge.noDiscover`**
  2. Add IP addresses, wildcards, or filters you want excluded by Extended Topology discovery. Enter one per line.
  3. Run a new Extended Topology discovery.

©2004 Hewlett-Packard Development Company, L.P.

You can exclude devices from Extended Topology discovery by creating an **Extended Topology Discovery Exclusion List**.

Creating the Extended Topology Discovery Exclusion List

Extended Topology limits the breadth of discovery according to the contents of the following file:

- Windows: `%OV_CONF%\nnmet\bridge.noDiscover`
- UNIX: `$OV_CONF/nnmet/bridge.noDiscover`

You enter NNM filter names, IP addresses and wildcards into the `bridge.noDiscover` file that you want the Extended Topology discovery process to ignore.

Here are a few examples of valid `bridge.noDiscover` file entries:

- **`10.2.112.86`** # Exclude this IP address from discovery.
- **`*.*.*.*`** # Exclude all nodes from discovery.
- **`10.2.*.*`** # Exclude all IP addresses from 10.2.0.0 to 10.2.255.255.
- **`10.2.0-2.*`** # Excludes all nodes from 10.2.0.0 to 10.2.2.255.

- **`Routers`** #Excludes all nodes matching the NNM filter *Routers*.

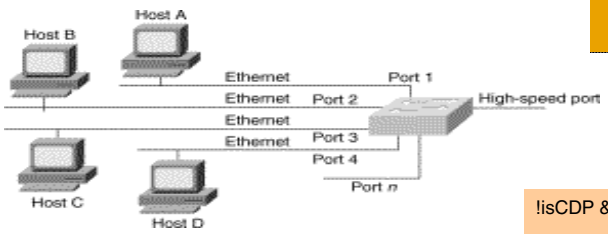
Do the following to create the `bridge.noDiscover` file:

1. As `Administrator` or `root`, create the `bridge.noDiscover` file.
2. Add NNM filter names, IP addresses or wildcards you want excluded by Extended Topology discovery. Enter one NNM filter name, IP address or wildcard per line.
3. You can explicitly test what nodes get excluded using the filter using the [`Test Zone Configuration`] button in the Extended Topology Configuration dialog (and then looking at the results in `$OV_TMP/etzonetest.out`) or by running `ovtopodump -f <`*`filter-name`*`>` for each of the specified filters.
4. Run a new Extended Topology discovery.

bridge.noDiscover Filtering

bridge.noDiscover Filtering

•Use filters from NNM **`filters`** file



("IP Hostname" ~ ".*switch.*\.foo\.net")

Target Device Filter

!isCDP && "IP Address" !~ 10.2.20.1-254

Peer Device Filter (for interface)

!isCDP && !isSNMP && "IP Address" !~ 10.2.20.1-254

Peer Device Filter (for interface)

sysObjectId	Scope based on type of system: .1.3.6.1.4.1.9.*
IP Hostname	Naming convention (switch.location.corp.net)
Capabilities	isCDP, (isBridge && !isGateway)
Interface Properties	Based on ifName, ifDescr, ifType, ifAlias "SNMP ifDescr" ~ "VLAN.*"; "SNMP ifType" = "ATM"
Peer Node Properties	Capability(!isCDP, !isSNMP), IP address range (server farm), SNMP agent

©2004 Hewlett-Packard Development Company, L.P.

If you have more nodes in your `netmon` discovery than you want to manage with Extended Topology, you may limit the nodes that are passed to Extended Topology discovery

using NNM filter definitions. `ovet_bridge` allows for an NNM filter name to be specified in `bridge.noDiscover`. This allows you to specify whole classes of devices (e.g., vendor XYZ's products, all end nodes, etc.) in a much easier fashion.

Note that examining filters does increase the CPU and time required for Extended Topology discovery.

Zone Discovery

Extended Topology discovers router connectivity information only if routers support CDP. If your routers don't support CDP, it doesn't matter how you divide them up, as Extended Topology cannot obtain connectivity information from routers not supporting CDP.

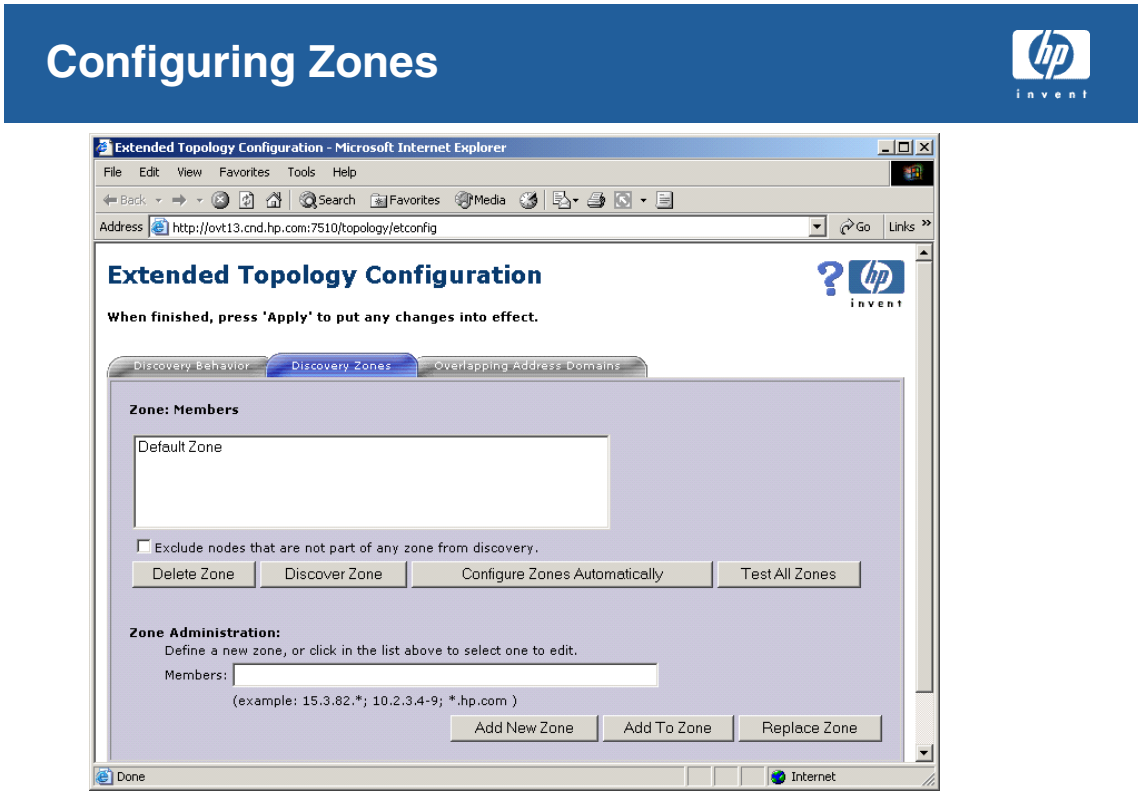
Zone-based discovery for Extended Topology consumes fewer computer resources, resulting in potentially much faster network discovery. Zones can be thought of as "islands of connectivity," groupings of IP subnets, which are discovered independently and later brought together through their edge connections. A good strategy for defining zones is to categorize your network devices by geography, such as by city, state, or building.

NOTE When defining your zones, do not separate switches that are connected together within a subnet. Keeping connected switches in the same zone allows Extended Topology to discover switch meshes and switch configurations with multiple VLANs. Keep switches that participate in the same VLAN together. If you separate switches, disconnected topologies may result.

Routers are expected to participate in multiple zones. After zones are discovered, "multi-zone" nodes are stitched together. For example, if a router has IP interfaces 15.2.112.1 and 15.2.120.1, and the zone configuration defines the 15.2.112 and 15.2.120 subnets as being in different zones, `ovet_disco` interacts with the device during the zone defined for the 15.2.112 subnet, creating connections from the router to devices (e.g., switches) inside the zone. The node and interface entities for the router are sent to modelling at the completion of discovery for the zone. When the

zone defined for the 15.2.120 subnet is discovered, a different set of connections are created for the interfaces on the router attached to devices in that zone. At the complete of this second zone, only the connected interfaces of the router in this zone are updated in modelling. Extended Topology communicates with a device again for each zone in which it participates.

Configuring Zones



 ©2004 Hewlett-Packard Development Company, L.P.

Manually Configuring Zones

With automatic zone configuration, Extended Topology configures zones for you. However, in certain circumstances you may want to manually adjust your zone configuration to try and reduce Extended Topology's discovery time or to address warnings given by [`Test All Zones`].

NOTE When defining your zones, do not separate switches that are directly connected together within a subnet.

To manually organize your devices into zones, use the following procedure:

1. Open the Configure Extended Topology GUI using the NNM `Options`: `Extended Topology Configuration` menu or select the `Discovery Status` tab from `Home Base`, then select [`Extended Topology Configuration`].
2. Select the `Discovery Zones` tab.
3. Organize your devices into zones. You may need to calibrate your zones as outlined in this procedure.
   - Organize zones with fewer nodes when these nodes contain many interfaces. An example

of this would be a network that contains a high quantity of switches housing many ports.

- Organize zones with more nodes when these nodes contain fewer interfaces. An example of this would be a network that contains a low quantity of switches housing few ports.

4. You can limit Extended Topology discovery to only those devices you specify in zones. To do this, select the check box that excludes nodes that are not included in any of the zones you configure.

NOTE When the Extended Topology discovery process begins, NNM transfers its node information to Extended Topology. Extended Topology includes these nodes in its discovery process. If you assign nodes to discovery zones, there may be a subset of the nodes transferred from NNM that aren't included in any zone. Extended Topology automatically assigns these remaining nodes to a default zone. Select this check box to stop Extended Topology from discovering these nodes.

You can also limit your discovery with the `bridge.noDiscover` file. Devices specified there are not discovered regardless of how you configure your zones.

5. In the `Zone:Administration` text box you can specify any hostname that your DNS server can resolve to an IP address, or directly specify any node's IP address. Separate entries with a semicolon. You can also use the following wildcard symbols:

   - Asterisk: Use an asterisk to represent any number of characters up to the next period:

     `*.corp.com` matches `pc.corp.com` or `ws.corp.com`.

     `pc.*.com` matches `pc.corp.com` or `pc.location.com`.

     `*.*` matches `corp.com` or `pc.com`, but not `pc.corp.com`.

     The * in 10.*.1.3 matches any number.

   - Question mark: Use to match a single character:

     `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`.

     `pc.???.com` matches `pc.abc.com`, `pc.bcd.com`, or `pc.cde.com`, but does not match `pc.ab.com` or `pc.abcd.com`.

   - Brackets: Use to match single characters, characters within a range, or characters not within a range:

     `[bcf]an.corp.com` matches `ban.corp.com`, `can.corp.com`, or `fan.corp.com`, but does not match `dan.corp.com` or `lan.corp.com`.

     `[b-d]an.corp.com` matches `ban.corp.com`, `can.corp.com`, or `dan.corp.com`, but does not match fan.corp.com, an.corp.com, or clan.copr.com.

     `[!d-z]an.corp.com` restricts the selection to `aan.corp.com`, `ban.corp.com`, or `can.corp.com`.

   - Dash: Specify a range of IP addresses.

     10.2.1-3.1 represents 10.2.1.1, 10.2.2.1, or 10.2.3.1

6. Select `[Add New Zone]` to move each new zone into the `Current Zones` area of the `Extended Topology Configuration` screen. Select a zone, then select [`Add to Zone`] to add more addresses to a specific zone. You can also use `[Delete]` and `[Replace]` to help you manage your zones.

7. Select [Test All Zones] to test your zones. This test will check your zone configuration and may recommend configuration changes.
8. Select [Apply] to save your changes.
9. Once these zones are successfully configured, select [Initiate Full Discovery Now] or execute, as Administrator or root,

   etrestart.ovpl

10. You can check the amount of time Extended Topology spends discovering your zones by running the ovstatus -v ovet_disco command. If the discovery time of any of your zones is abnormally long when compared to that of other zones, do one or both of the following:

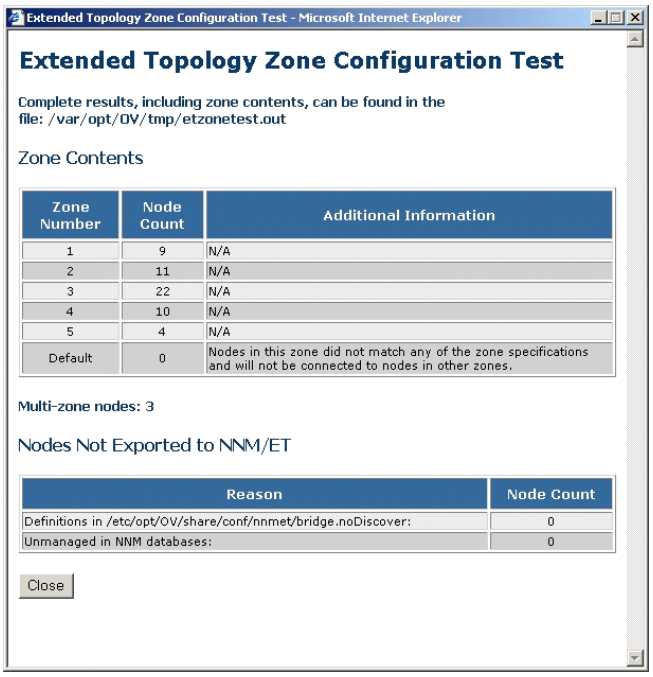
    - Add a new zone and move some of the devices from the problem zone into the new zone.
    - Move some of the devices from the problem zone into one or more of the existing zones that are being discovered faster.

    Once these zones are successfully configured, select [Initiate Full Discovery Now] or execute, as Administrator or root, the etrestart.ovpl script to start a new Extended Topology discovery.

NOTE Once Extended Topology completes a discovery with the new zone configuration, you should check the discovery results to make sure your zones are configured correctly.

Zone Configuration Test

Zone Configuration Test

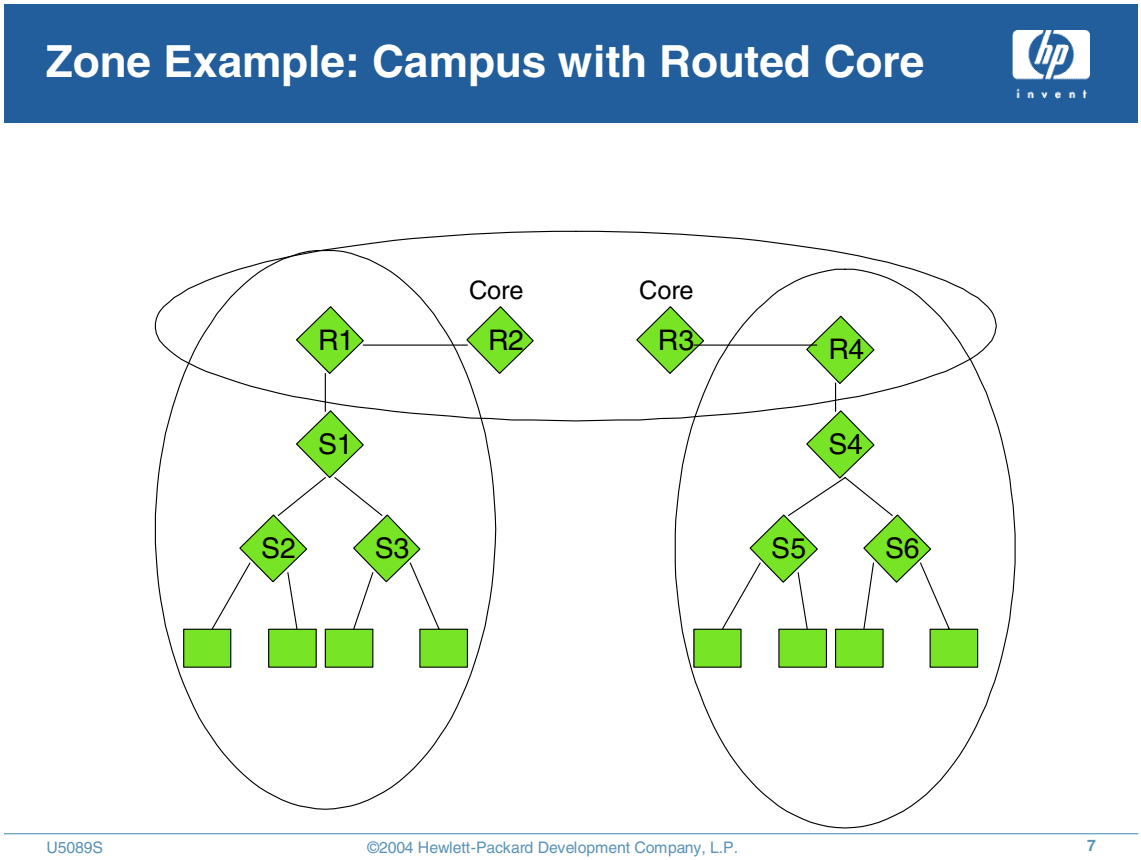


U5089S ©2004 Hewlett-Packard Development Company, L.P. 6

The [`Test Zone Configuration`] button triggers a script that uses the zone configuration to analyze NNM topology. It warns you of non-recommended zone configurations, such as zones exceeding available memory capacity, etc., as well as producing a breakdown of the network by zone for you to view in a separate file (`/var/opt/OV/tmp/etzonetest.out`), if desired.

This number is determined at the time that `setupExtTopo.ovpl` is run, based on physical memory, and can be 150, 200, 350, or 550.

Zone Example with Routed Core



Possibility 1: Core Devices are Routers

- The network contains core devices that are all routers.
- Campus buildings/sites are mostly switched, with router interfaces placed between switch blocks or VLANs.

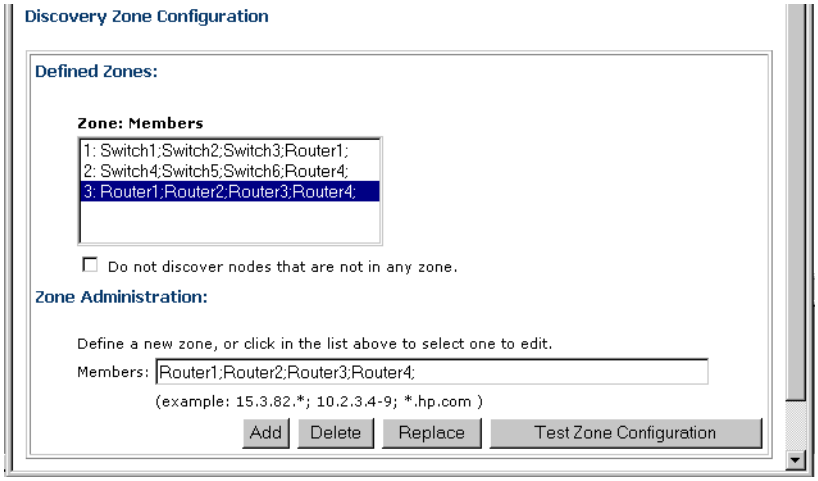
Suggested Zone Configuration

- To configure discovery for the switch-to-switch and switch-to-router connectivity, do the following:

  Put all of the access switches, distribution switches/routers, and core router(s) that service the building into zone 1 using the name wildcard method for defining a zone. Using this method, you can create a zone for every building in your campus, provided it meets the above criteria.
- To configure discovery for the router-to-router connectivity, do the following:

  Configure the core routers/switches into one zone by themselves. Splitting the core

routers/switches into multiple zones may increase management traffic from bordering routers to the core routers for each zone you add the core routers to. You should split large core routers into a few zones to speed up discovery. Doing this will speed up discovery at the cost of higher management traffic. It is a design choice that you need to make at install/setup time.



If you don't need to view router connectivity within the Extended Topology neighbor view, configure the core routers in as few zones as possible.

If the network size in each building is too small, combine the networks of two or more buildings to get a bigger zone (up to the maximum recommended by `setupExtTopo.ovpl` for your memory size). Add the core routers for each building into that zone.

Using this information as a guideline, adjust your zone size higher or lower than the recommended number of nodes depending on the following parameters:

— The port densities of the devices you plan to discover.

— Extended Topology system memory size.

— Extended Topology system CPU size.

— The number of VLANs your network contains.

— Other resource settings.

Remember to use the [`Test Zone Configuration`] button to test your zones.

Extended Topology Support Tools

Extended Topology Support Tools

- During discovery
  - **`dumpDiscoStatus.ovpl`** - summary of current discovery state for each agent
  - **`dumpAgentProgress.ovpl`** - progress of specified agent
- After discovery
  - **`Topology Summary`** – compare NNM and Extended Topology information
  - **`Doesn't respond to SNMP hyperlink`** - list nodes which did not respond to SNMP requests
  - **`ovet_topodump.ovpl`** – node and interface information for normal and overlapping domain nodes

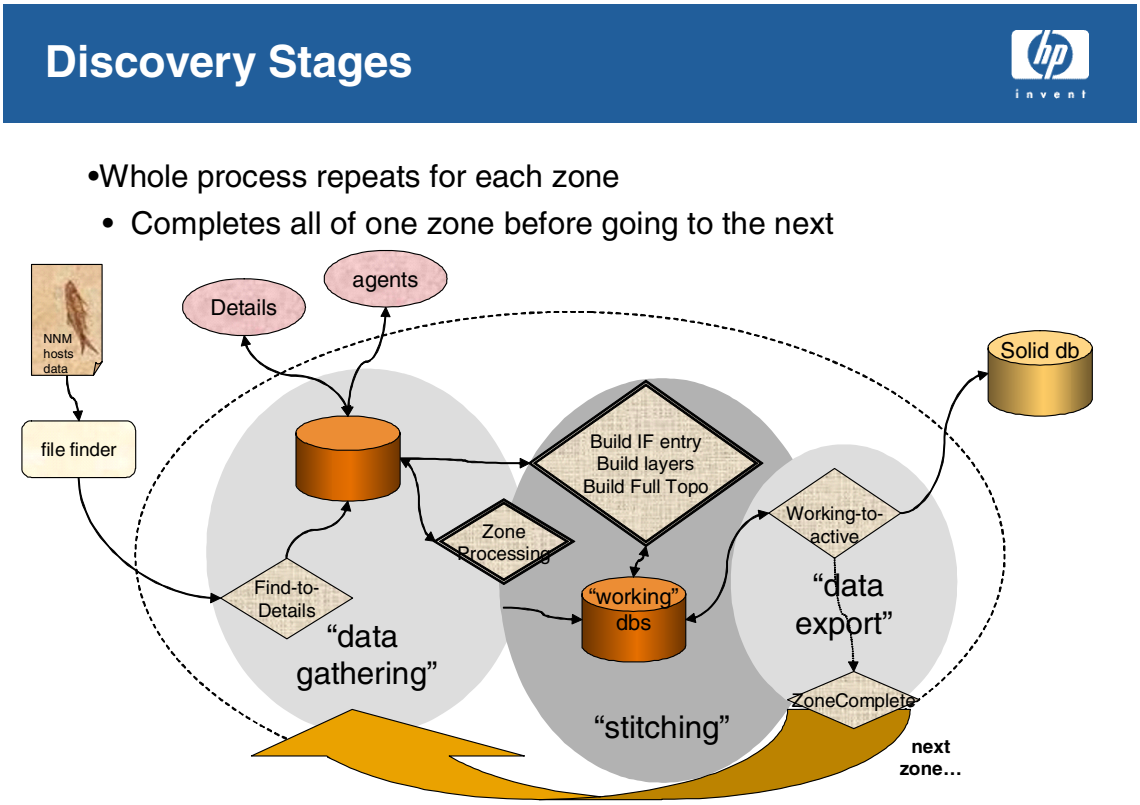
U5089S ©2004 Hewlett-Packard Development Company, L.P. 8

Extended Topology provides tools to enable you to inspect the Extended Topology database. The intended use of the tools is for trouble shooting problems related to topology.

You can run these tools from `/opt/OV/support/NM` on UNIX, *`install_dir`*`\support\NM` on Windows.

To see a summary of the nodes which didn't respond to SNMP queries, use the hyperlink on the Topology Summary page for "Doesn't respond to SNMP."

Discovery Stages



Extended Topology discovery proceeds in five stages of 0 through final.

- Phase 0: the file finder feeds `hosts.nnm` and `IPv6Seed.conf` to `ovet_disco`.
- Phase 1: discovery agents collect protocol, interface and VLAN information.
- Phase 2: is unused in Extended Topology.
- Phase 3: switch discovery agents collect Forwarding DB table data.
- Final phase: stitchers run to process agent-collected data.

After completion, Extended Topology restarts `ovet_disco` in stand-by mode and waits for the next discovery cycle.

Phase 0

During Phase 0, the file finder gets the `hosts.nnm` and `IPv6Seed.conf` files to insert the individual nodes into `ovet_disco`'s internal table.

Each node that is an IPv4 node is dispatched to the **Details agent**. This unique agent runs prior to the Zone Processing stitcher. The Details agent does a quick SNMP query to the node to retrieve its system information, especially its sys_objectID. The Details agent puts the data from

each node (and whether it responded to the SNMP query) into the details.returns table within `ovet_disco`.

The Details agent completes prior to Phase 1. It receives all nodes, with no filtering based on sys_objectID.

Phase 1

After the Details agent has processed the entire list of nodes, `ovet_disco` calls the **Zone Processor**. The Zone Processor reviews the number of configured zones and gets the nodes from the first zone.

For each node in that zone which responded to the Details SNMP queries, the Zone Processor dispatches it to all the discovery agents other than Details. (Nodes which did not respond get a dummy entry in the database at the end of the discovery process.) As the Zone Processor places the node in the work table for each agent, `ovet_disco` validates the node's sys_objectID against the list of OIDs appropriate for this agent configured in the `.agnt` file. Inappropriate nodes are dropped from the work table and not dispatched to the agent.

The **agents** gather SNMP data from the nodes in their lists. During Phase 1, they gather interface and VLAN data, as well as protocol information such as EDP and CDP. Some agents, such as the IFDetails, CDP, EDP, ILMI, and HSRP agents complete all their work during Phase 1.

The various **Switch agents** also begin their work during Phase 1 and collect interface and VLAN information during Phase 1.

Phase 3

When Phase 1 officially ends and Phase 3 officially begins, the various **Switch agents** gather additional information from the Forwarding Data Base table on each switch. This allows them to pair a discovered interface with its remote neighbors.

Any switch that is not assigned to a vendor-specific agent (based on sysObjectID) goes to the StandardSwitch agent. A node which is assigned to a vendor-specific agent, but fails that agent's filtering may be reassigned to the StandardSwitch agent.

Final Phase (Phase 4)

After the agents have collected all the data from the nodes, `ovet_disco` transitions from data collection into data processing mode.

First the data is combined to build an interface entity. From there the stitchers build the layers, such as ATM, CDP, and IP). Finally these are combined into the Full Topology.

Zone-Independent Processing

The phases are repeated for each zone configured. Once all zones have been discovered and stitched, the VLAN, HSRP, and Connection data from across zones are stitched together.

dumpDiscoStatus.ovpl

dumpDiscoStatus.ovpl

- Use during discovery
- Lists each agent and it's current operational state
- Expect to see them all transition up to 4
- Watch for ones stuck on 3
  - Determine which agent is taking all the time
  - Get the name of the agent for **`dumpAgentProgress.ovpl`**

- These tools allow you to see the progress of discovery with more granularity than the GUI.

©2004 Hewlett-Packard Development Company, L.P.

You can run the support tool `dumpDiscoStatus.ovpl` during Phase 3 of discovery:

```
ovet_oql ( Command Line OQL )
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.

ovet_auth has authenticated your session.
Executing query:
select * from disco.status;
{
m_DiscoveryMode=0;
m_Phase=-1;
m_BlackoutState=1;
m_CycleCount=0;
}
( 1 record(s) : Transaction complete )
ovet_oql ( Command Line OQL )
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.
```

```
ovet_auth has authenticated your session.
Executing query:
select * from agents.status where m_State <> 0;
.......
{
m_Name='CDP'; <--------------------------Get the Agent Name!
m_State=4;
m_LastRecordTime=1046903784;
m_NumConnects=1;
}
{
m_Name='CiscoSwitchSnmp'; <--------------------------Get the Agent Name!
m_State=4;
m_LastRecordTime=1046903780;
m_NumConnects=1;
}
{
m_Name='Details'; <--------------------------Get the Agent Name!
m_State=4;
m_LastRecordTime=1046903776;
m_NumConnects=1;
}
...<cut>...
{
m_Name='InterfaceDetails'; <--------------------------Get the Agent Name!
m_State=4;
m_LastRecordTime=1046903791;
m_NumConnects=1;
}
{
m_Name='StandardSwitch'; <--------------------------Get the Agent Name!
m_State=4;
m_LastRecordTime=1046903915;
m_NumConnects=1;
}
( 7 record(s) : Transaction complete )
```

dumpAgentProgress.ovpl

dumpAgentProgress.ovpl

- Despatch shows all the devices that will run through that agent
- Number of records shows how many must finish
- See results of the queries that finished
- See the record the agent is currently working on
- The message m_fail indicates that the device does not support the right MIB.

©2004 Hewlett-Packard Development Company, L.P.

The support tool `dumpAgentProgress.ovpl StandardSwitch`:

```
ovet_oql ( Command Line OQL )
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.
ovet_auth has authenticated your session.
Executing query:
select * from StandardSwitch.despatch;
{
m_UniqueAddress='15.2.113.152';
m_Name='hpcndsn.cnd.hp.com';
m_ObjectId='1.3.6.1.4.1.11.2.3.2.5';
}
{
m_UniqueAddress='15.2.113.65';
m_Name='ratbert.cnd.hp.com';
m_ObjectId='1.3.6.1.4.1.11.2.3.2.5';
}
```

```
{
m_UniqueAddress='15.2.113.9';
m_Name='tthp17.cnd.hp.com';
m_ObjectId='1.3.6.1.4.1.11.2.3.2.5';
}
...<cut>...
( 10 record(s) : Transaction complete )
```

Viewing Extended Topology Data

Viewing Extended Topology Data

- Supported means to display Node, Interface and Address information for the Extended Topology database

- Syntax:

```
ovet_topodump.ovpl [-node|-nodeif|-info] [ IP Address
[<OADId>] | NodeName ]
```

- Example:

```
ovet_topodump.ovpl -nodeif 15.2.32.81

NAME                    STATUS     COMMADDR       PRVADDR    OADID
mcrouter81.cnd.hp.com   Normal     15.2.32.81     -          0
Fa0/0                   Normal     -              -          -
ssss                    Normal     -              -          -
-                       Rspd       15.2.32.81     -          0
```

 ©2004 Hewlett-Packard Development Company, L.P.

`ovet_topodump.ovpl` provides a supported means for you to display node and interface information of data in the Extended Topology database. NNM has `ovtopodump` but the tool does not query Extended Topology data. In an OAD environment, OAD nodes are only stored in the Extended Topology database and not in the NNM database.

You can query node and interface topology information by:

- IP Address and Overlapping Address Domain (OAD) name
- Dump all nodes in a OAD domain
- Dump all nodes in the topology

`ovet_topodump.ovpl` supports these parameters:

`-node [OADId]` Dump all the node information. If Overlapping Address Domain Id is given, only dump the nodes in the OAD.

`-nodeif [OADId]` Dump all the nodes and their interface information. If Overlapping Address Domain Id is given, only dump the nodes in the OAD.

`-node <Name|IP> [OADId]`: Dump the node and its interfaces which has the given Name or IP. If Private IP is used, OADId needs to be provided. If no OADId is supplied, the IP is assumed to be a public IP.

No wildcard is allowed on either the address or node name. However for node name, a short name

is accepted (cisco1 rather than cisco1.cnd.hp.com). If more than one node is match by the node name, only the first match is returned.

The following data is shown for a node:

- Node Name
- IPAddress – Public, Private, OAD (integer)
- Overall Status

For an interface:

- IF Alias or IF Name or IF Description (in the listed order)
- IPAddress – Public, Private, OAD (integer)
- Overall Status

If no parameters are given, the topology summary is displayed.

NAME: The hostname of the Node. For interface (indented from the Node Name), the name is picked from ifAlias, ifName, ifDescription or boardno/portno. For address, this is a '-'.

STATUS: For Node and Interface, this is the overall status field of the object. For address, this is the ping state.

COMMADDR: The communication (public) address. For node, this is the management address. For interface, the list of addresses (if > 1) come after the interface.

PRVADDR: The private address. For node, this is the management address. For interface, the list of addresses (if > 1) come after the interface. If private address is the same public address, it is shown as '-'.

OADID: The Overlapping Address Domain ID. An integer or "-" if not applicable.

Deployment Tips: Pre-Discovery

- Understand your network before you start.
  - What are all of the complexities of your network?
  - Does your network contain a lot of VLANs?
  - Does your network contain a lot of redundant paths?
  - Does your network contain HSRP routers?
- Test DNS before you start.
- Test with a small NNM discovery.
- Filter out unsupported devices.
- Enable CDP or EDP when possible.

Extended Topology Deployment Tips

OpenView engineers compiled some simple, yet important Extended Topology deployment tips. For better understanding, read through the Extended Topology manual prior to reviewing the following information. It is recommended that you read through this entire list of deployment tips prior to running your first Extended Topology discovery.

Prediscovery Tips

Below is a list of activities that you may want to complete prior to running your first Extended Topology discovery.

- It is much easier to discover devices and connectivity on a healthy network. The support directory contains tools to help you understand the health of your NNM installation. Scripts such as `checkDNS.ovpl` and `resolveNames.ovpl` can be extremely helpful. You can find support tools in the following (support) directory:
  - Windows: *`install_dir`*`\support`

— UNIX: `$OV_MAIN_PATH/support`

- You should research whether Extended Topology can discover information about the devices on your network. You can do this by viewing the device list or the supported MIBs for a family of devices at the following URL:
  `http://www.hp.com/managementsoftware/products/nnmet/support/device_support.html`.

  If your network contains unsupported devices, you may want to run your first Extended Topology discovery without these nodes. You can use the `bridge.noDiscover` file to stop Extended Topology from discovering device information about these nodes. See the *bridge.NoDiscover* reference page in NNM's online help (or the UNIX manpage) for more information.

NOTE If you choose to run your discovery and include these nodes, these unsupported devices may cause problems.

- You should not allow Extended Topology to discover device information about managed hubs. You should enter any managed hubs in the `bridge.noDiscover` file. You may use an NNM filter to do this. See the *ovtopodump* reference page in NNM's online help (or the UNIX manpage) for more information.
- It is a good idea to start your first Extended Topology using only a small quantity (10-20 nodes) of directly connected nodes. It is best to target networking nodes such as switches and routers. You could include a pair of routers that support HSRP too. You should know their topology so you can validate the results. Another advantage of a small test run is that you don't need to worry about zone definitions.
- If you want to avoid using complex Extended Topology filters, you can use the following approach to create a small NNM database containing only a few nodes:

  1. Use the `loadhosts` tool to build a small database. See the *loadhosts* reference page in NNM's online help (or the UNIX manpage) for more information.
  2. After loading nodes with the `loadhosts` tool, run an `nmdemandpoll` *`nodename`* on each node to speed up NNM's configuration poll. See the *nmdemandpoll* reference page in NNM's online help (or the UNIX manpage) for more information.
  3. After you complete loading these nodes, run, as root or Administrator, the `setupExtTopo.ovpl` script to configure Extended Topology and start your discovery. See the *setupExtTopo.ovpl* reference page in NNM's online help (or the UNIX manpage) for more information.

- Before running your first Extended Topology discovery, you need to understand your network topology. You should understand the following characteristics of you network.

  — What are all of the complexities of your network?

  — Does your network contain a lot of VLANs?

  — Does your network contain a lot of redundant paths?

  — Does your network contain HSRP routers?

  You'll be more confident in the Extended Topology discovery results if you can compare them to the actual topology.

- There are several protocols that can help Extended Topology discovery layer 2 more accurately. For example, if the Cisco devices on your network support CDP and the Extreme Network devices on your network support EDP, then your layer 2 connectivity should be much more accurate.

- Networks that have low traffic levels are much more difficult to discover. This is due to switches in this low traffic area having empty Forwarding Database (FDB) Tables.

  To remedy this, run your Extended Topology discovery when the network is active. If you must run your Extended Topology discovery during low traffic times, telnet onto the switches shortly before Extended Topology discovery and ping the adjacent switches. This will fill the FDB tables. If your switches support CDP or EDP, this may not be necessary.

- If you run OSPF in your network, and you have purchased and installed a license for the Advanced Routing SPI, then you need to run an OSPF discovery separately from a normal Extended Topology discovery. It is important to understand that Extended Topology only discovers OSPF nodes that are currently managed by NNM.

  You will probably want to run an OSPF discovery against your large network, as running this discovery with only a small test network will probably not be successful. A nice debug approach is to run `ospfdis.ovpl dbg=1`.

Deployment Tips: During Discovery

Below is a list of activities that you should complete in order to successfully run your first Extended Topology discovery.

After you have used the relevant information in the Prediscovery Tips section, you are ready to run your first Extended Topology discovery. If you have not enabled Extended Topology, run, as `root` or `Administrator`, the `setupExtTopo.ovpl` script. Answer *yes* to questions about enabling the appropriate agents. After this script completes, an Extended Topology discovery begins.

If you enabled Extended Topology earlier, then, as `root` or `Administrator`, run the `etrestart.ovpl` script to run a new Extended Topology discovery. See the *etrestart.ovpl* reference page in NNM's online help (or the UNIX manpage) for more information.

TIP	You can use the `-verbose` option with the `etrestart.ovpl` script to show process status messages.

To make sure that the entries in your `bridge.noDiscover` file are working correctly, monitor the following file in the couple minutes immediately preceding discovery:

- Windows: `%OV_DB\nnmet\hosts.nnm`
- UNIX: `$OV_DB/nnmet/hosts.nnm`

This file contains a list of all the nodes from which Extended Topology will discover information. This file is updated at the beginning of each Extended Topology discovery.

As an example of how to monitor the `hosts.nnm` file, suppose that you expected your filters to target an Extended Topology discovery of only ten nodes. If the `hosts.nnm` file contains more than ten nodes, then something is wrong with your filters and should be corrected.

It is a good idea to monitor the Discovery progress by launching the Home Base page from a browser. To do this, open the Extended Topology Configuration GUI using the NNM `Options`: `Extended Topology Configuration` menu or select the `Discovery Status` tab from `Home Base`. If you are having problems getting Home Base to come up, try the following:

- Make sure you have the right JPI installed.
- Launch Home Base from another computer. If this works, look for differences in your browser settings.
- Proxy Servers can cause some problems for Home Base. Try running without the Proxy Server and see if that helps.
- Try clearing the cache on your browser and clearing the cache in your JPI (done via the control panel in Windows).
- Another common problem you could encounter is an improperly configured domain name resolution (DNS) server. The browser must be able to resolve the DNS name of the Extended Topology server. This problem usually manifests itself with Extended Topology displaying a blue box without loading the Dynamic Views applet. To test DNS, from the Extended Topology system, run the following script:
  - Windows: *`install_dir`*`\support\NM\ovgethostbyname.ovpl`
  - UNIX: `$OV_MAIN_PATH/support/NM/ovgethostbyname.ovpl`

    This script should return a fully qualified host name (like foo.hp.com). If it returns a short name, you should change your DNS server configuration or your hosts file on the Extended Topology server to remedy this problem. After your name resolution is working properly, you need to, as `root` or `Administrator`, re-run the `setupExtTopo.ovpl` script.

Deployment Tips: Post-Discovery

- From the Home Base Discovery Status tab, click [Topology Summary].
  - Verify layer 2 connections were discovered.
  - Check for nodes not responding to SNMP using the “Doesn’t respond to SNMP” hyperlink.
- Launch other views to verify results.

©2004 Hewlett-Packard Development Company, L.P.

Below is a list of activities that you will want to complete after running your first Extended Topology discovery.

After discovery completes, go to Home Base and select the `Discovery Status` tab. From there, select [`View Topology Summary`] and review the results.

1. Review the number of Layer 2 and VLAN connections. Were any found during the discovery? If not, then something went wrong during the discovery. Extended Topology needs SNMP access to devices in order to complete an accurate discovery. You should check for nodes that didn’t respond to SNMP by using the hyperlink on the Topology Summary page for "Doesn’t respond to SNMP."
2. You should fix any SNMP access problems using NNM’s SNMP Configuration user interface (UI). You can access the SNMP Configuration UI using NNM’s `Options:SNMP Configuration` menu. Extended Topology will apply any changes you make to its next discovery.

Another useful tool to examine the results of the discovery is listed below:

- Windows: *`install_dir`*`\support\NM\ovet_topoobjcount.ovpl -al`
- UNIX: `$OV_MAIN_PATH/support/NM/ovet_topoobjcount.ovpl -al`

Use the following techniques to launch some of the other views to see how they look.

- From Home Base, try running a neighbor view for one of your switches.
- If you are happy with your views and want to try a full discovery, now is a good time to have Extended Topology discover your entire network.
- If you would rather take a more conservative second step when setting up a more extensive Extended Topology discovery, consider creating a *medium size* discovery with a few hundred nodes.To do this, you would remove specific switches and routers from the `bridge.noDiscover` file or you could modify some of your zone definitions. Go through the same analysis previously described to validate your discovered devices as you did in prior steps. Make sure you don't leave out critical network devices and end up with connectivity gaps.

Database Support for Extended Topology Data

- Allow Extended Topology database to be stored in RDBMS other than Solid.
- Both NNM Data Warehouse and Extended Topology tables must run on the same RDBMS, either both on Oracle or both on Solid or both on MS SQL Server.

 ©2004 Hewlett-Packard Development Company, L.P.

NNM supports Oracle or MS SQL Server RDBMS for the data store used by the Extended Topology database. Many users want the choice of RDBMS products to use with OpenView.

Both the NNM Data Warehouse and the Extended Topology tables are created at the same time. The setup scripts and steps documented for the Data Warehouse setup also create the instances and tables needed for Extended Topology.

Both the Data Warehouse and the Extended Topology tables must run on the same RDBMS, either both on Solid or both on Oracle or both on MS SQL Server. They use the same configuration file to find out where to connect through ODBC.

XPL Logging

- The XPL Logging feature gives us:
  - The ability to send all Log messages to a single shared log.
  - Provide a consistent way of producing these log messages.
  - The functionality to allow for multiple severity levels of logging.
- Logging is ALWAYS ON

 ©2004 Hewlett-Packard Development Company, L.P.

The feature generates the ability to send all Log messages to a single shared log. It also gives us an OpenView consistent way of producing these log messages. It also allows for multiple severity levels of logging. If the trace server is not running on the management system, all log messages are lost. No log file is created.

The logging implementation is able to:

- Serialize the message list in its raw form, to a file.
- Record a message in a platform-specific general logging location.
- Generate an event using `opcmsg` technology, meaning the logged event may be written to one or more locations.

XPL logging is currently available to all NNM and Extended Topology processes. Over time, more and more processes will switch to using this logging.

The Windows implementation records a message in the Windows Event Log. The message is formatted into a particular language BEFORE it is written to the Windows Event Log. On UNIX, this is written to a product log file as a multibyte text file using the current locale information to determine the particular multibyte encoding. As a text file, it is viewable using any text-based tool that can handle the multibyte encoding used.

View the log in `$OV_PRIV_LOG/system.txt` (UNIX) or `%OV_PRIV_LOG%\system.txt` (Windows).

Lab Exercises

Lab Exercises

- Set up MIMIC environment
- Discover a simulated environment
- Use monitoring tools repeatedly to watch discovery progress
- Split the environment into zones and rediscover
- Simulate a problem in the environment and rediscover

©2004 Hewlett-Packard Development Company, L.P.

Review Questions

1. What is the purpose of Zones of discovery?

2. Are there guidelines for defining Zones?

Preparing for Extended Topology Labs

Setting Up Extended Topology MIMIC Labs

Objectives:

Successfully setup the student workstation in preparation for execution of the Extended Topology Labs.

At the completion of this module you will be able to:

- Successfully execute and complete the labs.

Student Workstation System Setup Procedure:

NOTE UNIX: This procedure assumes that everything is executed as User root in a ksh environment and that the `/opt/OV/bin/ov.envvars.sh` file has been sourced in each working window.

Windows: This procedure assumes that `%OV_BIN%\ov.envvars.bat` has been executed in a cmd window.

CAUTION The procedure described for the Student's workstation assumes that NNM has been installed AND the Extended Topology component has been setup using the `$OV_BIN/setupExtTopo.ovpl` script.

During the `setupExtTopo.ovpl` script, answer YES to all questions. When you are prompted for the Administrator password (for tomcat), enter a username of `ov` and a password of `ov`. Allow discovery to complete before continuing.

1. Install the lab files. They must go in exactly the right directory. Execute

   UNIX:

   ```
   gunzip NNM7labs.tar.gz

   tar -xvpf NNM7labs.tar
   ```

   Windows: Install the `NNM7labs.zip.exe` self extracting file. Ensure that the files extract into the `%OV_CONTRIB%\NNM7labs` directory.

2. Initialize the labs using the IP address of the MIMIC server that your instructor provides.

   ```
   cd $OV_CONTRIB/NNM7labs
   initialsetup.ovpl ip_addr_MIMICSERVER
   ```

3. You will want to be able to save images of discoveries for comparison with broken discoveries. Enable `File:Save` and `File:Load` from dynamic views.

   The following labs require you to save dynamic views for later comparison. These options are not enabled by default in the product. To enable these menu options, do the following:

   a. Close all browsers and java consoles.

   b. Change to the directory `$OV_WWW_REG/dynamicViews/$LANG/`.

   c. Save a copy of the current `dynamicViews.xml` file as #dynamicViews.xml.

   d. Use an editor to edit the file `dynamicViews.xml`.

   e. Search for the string "Save/Load: unsupported".

f. Delete both lines that have the string “Save/Load: unsupported”. One starts with `<!--` and the other ends with `-->`. You must correctly delete both lines or the XML will have mismatched brackets.

g. Save the new `dynamicViews.xml` file.

h. Clear the browser and java caches:

- UNIX:

  ```
  rm –rf ~/.netscape/cache
  rm –rf ~/.jpi_cache
  ```

- Windows: Select Start->Settings->Control Panel.

  Double-click Internet Options.

  Under Temporary Internet Files, press the Delete Files button, click OK.

  In the control panel, double-click Java Plug-in.

  Under the Cache tab, press the Clear button.

  Close all windows.

You may have to disable proxying for your web browser as proxies also cache. See your instructor for information on this.

4. Your system is ready to begin the labs.

Troubleshooting Deployment

Lab D: Full Discovery, Setting the Stage

Objectives:

- Set the stage for the following labs: discover the correct topology.
- Gain additional experience using the support tools against additional agents not used before.

At the completion of this module you will be able to:

- Run several Extended Topology discovery process monitoring tools and understand their results.
- Execute the database dump tools and understand the results.

Assumptions:

- `$OV_CONTRIB/NNM7labs/initialsetup.ovpl` has been executed.

Directions

1. Change working directory to `$OV_CONTRIB/NNM7labs/Lab_deploy`:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_deploy
   ```

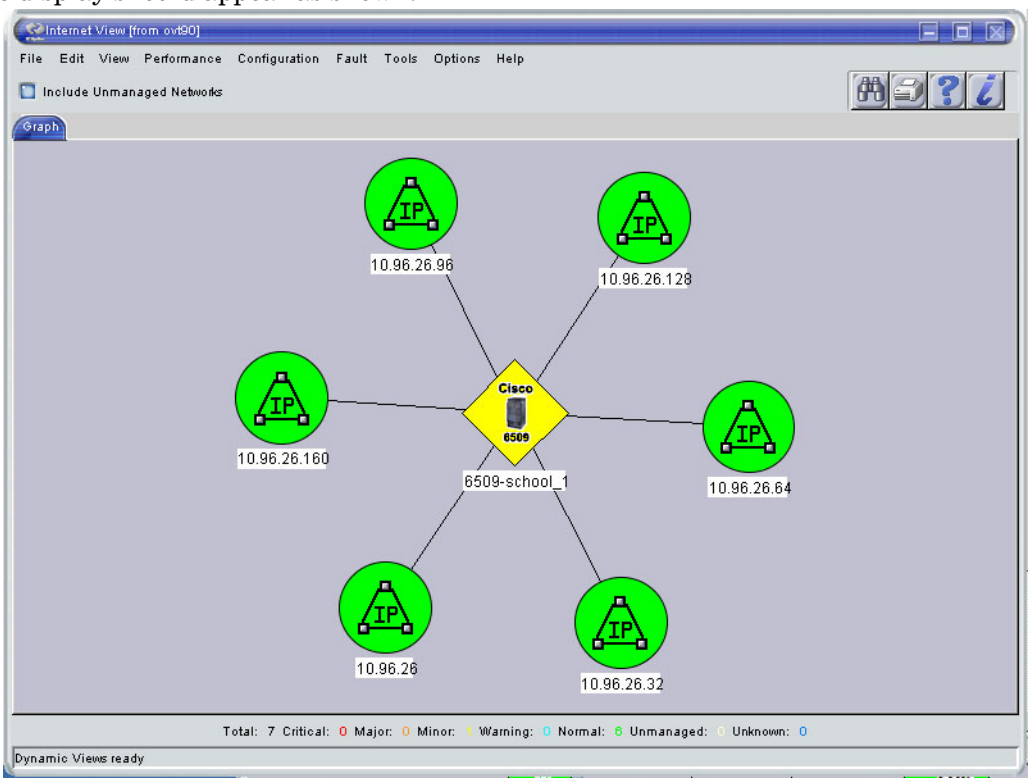
2. When the instructor has given you the go-ahead, execute the setup script:

   `setupLab_deploy.ovpl`

3. When this completes, open an Internet View from Home Base.

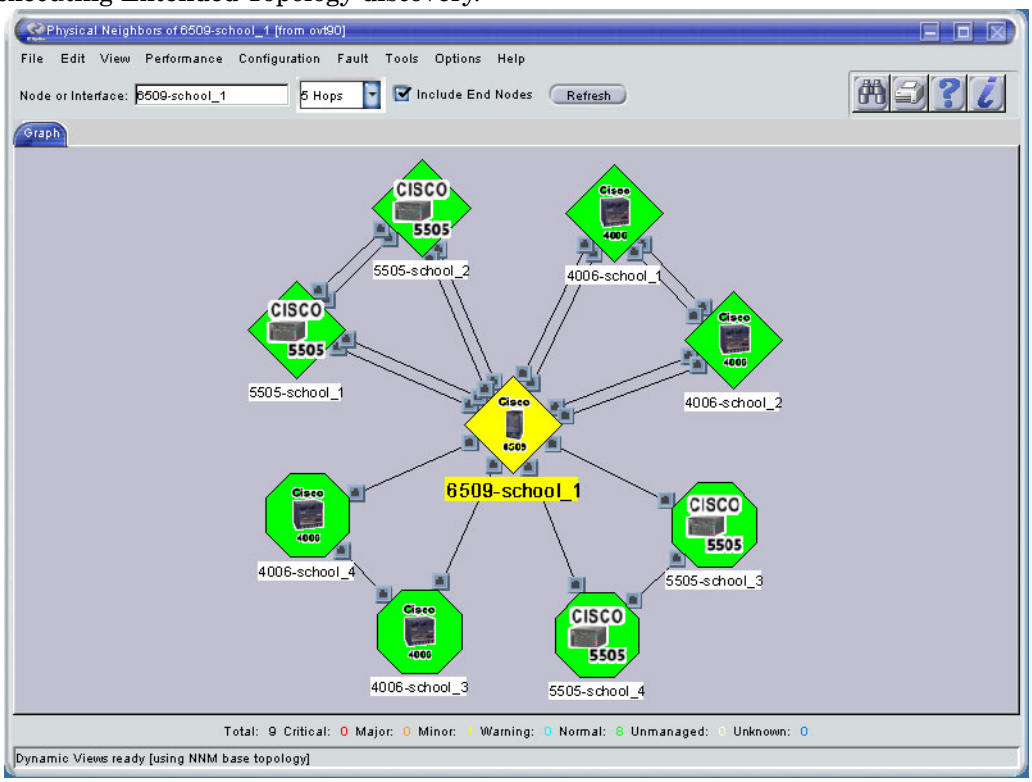
4. Verify proper and complete discovery of the simulated network. Locate any symbol that may be unmanaged or unknown. If unmanaged, select it, and then use the menu `Edit:Manage`. For any node which shows as a blank square, select the node and `Fault:Network Connectivity:Poll Node`.

5. The display should appear as shown.



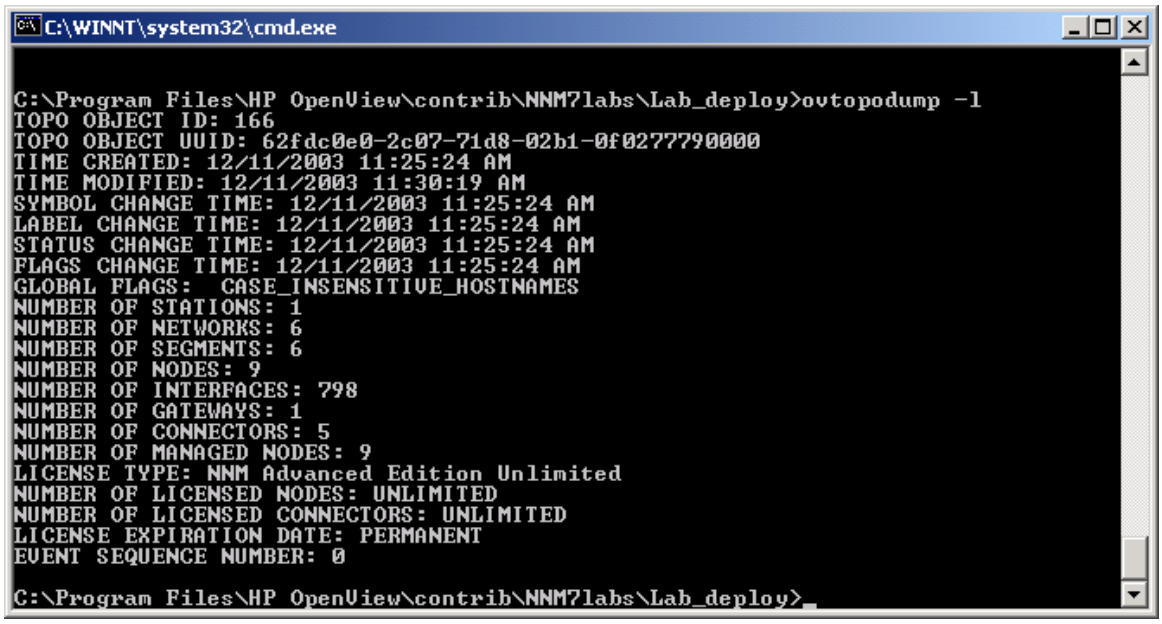
6. Execute a Neighbor view of the device 6509-school_1 with 3 hops and include end nodes, prior

to executing Extended Topology discovery.



7. Confirm correct topology has been discovered:

```
ovtopodump -l
```



8. When you have confirmed that all devices are managed and appear as shown in the above figures, commence the Extended Topology discovery from Home Base.

NOTE You will run several iterations of Extended Topology discovery throughout these labs. Each execution of the discovery will give you an opportunity to use the support tools to view progress of the discovery processes and agents. These

tools will be especially important in the labs that simulate discovery problems.

Be aware that the topology is quite small and the overall discovery process is brief. Hence the tools that dump status or progress and watch the log file(s) are only "active" a short period of time (during actual discovery). Run several discoveries in order to understand the full impact of these labs.

9. Once discovery has started (entered Phase 1 as indicated by `ovstatus -v ovet_disco`), in a separate window tail the `disco_log` file. Be aware that this file is renamed to `ovet_disco.log.old` when `ovet_disco` is restarted and when it completes.

   ```
   tail -f $OV_PRIV_LOG/ovet_disco.log
   ```

10. Execute:

    ```
    $OV_MAIN_PATH/support/NM/dumpDiscoStatus.ovpl
    ```

    to monitor the overall progress of discovery and to identify the state of each agent.

    m_State= 1 or 2 or 3 or 4. State 4 indicates completion.

    You are interested in m_State=3 (active) and 4 (completed).

    However, if you execute `dumpDiscoStatus.ovpl` too early, you will get stuck inside the query command as follows (the same occurs with dumpAgentProgress.ovpl):

    ```
    #> dumpDiscoStatus.ovpl
    ================================================Start
    ovet_oql ( Command Line OQL )
    Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.


    Executing query:
    select * from disco.status;
    ================================================END
    ```

    If you are stuck inside the query tool, you may get the prompt:

    ```
    |"tntdemo2:2.>"
    ```

    If so, press the Enter key and then type `quit;`.

    You will either see a listing of progress or exit the tool. If you exit, re-execute the tool.

A correct execution of the tool should result in a data dump similar to:

```
#> dumpDiscoStatus.ovpl
================================================Start
ovet_oql ( Command Line OQL )
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.

ovet_auth has authenticated your session.
Executing query:
select * from disco.status;
.
{
        m_DiscoveryMode=0;
        m_Phase=-1;
        m_BlackoutState=1;
        m_CycleCount=0;
}
( 1 record(s) : Transaction complete )
ovet_oql ( Command Line OQL )
```

```
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.

ovet_auth has authenticated your session.
Executing query:
select * from agents.status where m_State <> 0;
.......
{
        m_Name='CDP';
        m_State=4;
        m_LastRecordTime=1044731840;
        m_NumConnects=1;
}
{
        m_Name='Details';
        m_State=4;
        m_LastRecordTime=1044731839;
        m_NumConnects=1;
}
{
        m_Name='EDP';
        m_State=4;
        m_LastRecordTime=1044731843;
        m_NumConnects=1;
}
{
        m_Name='ExtremeSwitch';
        m_State=3;
        m_LastRecordTime=1044731965;
        m_NumConnects=1;
}
{
        m_Name='ILMI';
        m_State=4;
        m_LastRecordTime=1044731850;
        m_NumConnects=1;
}
{
        m_Name='InterfaceDetails';
        m_State=4;
        m_LastRecordTime=1044731859;
        m_NumConnects=1;
}
{
        m_Name='StandardSwitch';
        m_State=3;
        m_LastRecordTime=1044731964;
        m_NumConnects=1;
}
( 7 record(s) : Transaction complete )
#>
================================================END
```

While the m_State=3, the agent is not finished and the results are likely to be incomplete.

NOTE One of the more common problems with discovery is an agent appears to “hang” at m_State=3. By default, agents may take up to one hour before they time out. The training setup has reduced this to 10 minutes.

For a short period of time after the discovery has completed, you may be able to view the results of the agent when the m_State=4.

If you execute the `dumpDiscoStatus.ovpl` command and it returns the following output, you will not be able to view any agent details.

```
#> dumpDiscoStatus.ovpl
================================================Start
ovet_oql ( Command Line OQL )
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.

ovet_auth has authenticated your session.
Executing query:
select * from disco.status;
.
{
        m_DiscoveryMode=0;
        m_Phase=0;
        m_BlackoutState=0;
        m_CycleCount=0;
}
( 1 record(s) : Transaction complete )
ovet_oql ( Command Line OQL )
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.

ovet_auth has authenticated your session.
Executing query:
select * from agents.status where m_State <> 0;
.
( 0 record(s) : Transaction complete )
================================================End
```

While the m_State =3 (or 4) for an agent, you can run the `dumpAgentProgress.ovpl` tool to get agent details, for example:

```
#> dumpAgentProgress.ovpl ExtremeSwitch
================================================Start
Looking for agent "ExtremeSwitch" in Extended Topology database...

ovet_oql ( Command Line OQL )
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.

ovet_auth has authenticated your session.
Executing query:
select * from ExtremeSwitch.despatch;
...
{
        m_UniqueAddress='10.96.26.193';
        m_Name='black_diamond';
        m_ObjectId='1.3.6.1.4.1.1916.2.11';
}
{
        m_UniqueAddress='10.96.26.194';
        m_Name='3808-1';
        m_ObjectId='1.3.6.1.4.1.1916.2.17';
}
{
        m_UniqueAddress='10.96.26.226';
        m_Name='3808-2';
        m_ObjectId='1.3.6.1.4.1.1916.2.17';
}
( 3 record(s) : Transaction complete )
ovet_oql ( Command Line OQL )
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.

ovet_auth has authenticated your session.
Executing query:
select * from ExtremeSwitch.returns where m_LastRecord = 1;
.
( 0 record(s) : Transaction complete )
```

```
================================================END
```

The above results show that there are THREE entries to be processed, but ZERO have been processed.

The dump listing below shows the three devices (records) exist, and have completed processing.

```
#> dumpAgentProgress.ovpl ExtremeSwitch
================================================START
Looking for agent "ExtremeSwitch" in Extended Topology database...

ovet_oql ( Command Line OQL )
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.

ovet_auth has authenticated your session.
Executing query:
select * from ExtremeSwitch.despatch;
...
{
        m_UniqueAddress='10.96.26.193';
        m_Name='black_diamond';
        m_ObjectId='1.3.6.1.4.1.1916.2.11';
}
{
        m_UniqueAddress='10.96.26.194';
        m_Name='3808-1';
        m_ObjectId='1.3.6.1.4.1.1916.2.17';
}
{
        m_UniqueAddress='10.96.26.226';
        m_Name='3808-2';
        m_ObjectId='1.3.6.1.4.1.1916.2.17';
}
( 3 record(s) : Transaction complete )
ovet_oql ( Command Line OQL )
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.

ovet_auth has authenticated your session.
Executing query:
select * from ExtremeSwitch.returns where m_LastRecord = 1;
...
{
        m_UniqueAddress='10.96.26.193';
        m_Name='black_diamond';
        m_UpdAgent='ExtremeSwitch';
        m_Capabilities=['isLanSwitch'];
        m_LastRecord=1;
}
{
        m_UniqueAddress='10.96.26.226';
        m_Name='3808-2';
        m_UpdAgent='ExtremeSwitch';
        m_Capabilities=['isLanSwitch'];
        m_LastRecord=1;
}
{
        m_UniqueAddress='10.96.26.194';
        m_Name='3808-1';
        m_UpdAgent='ExtremeSwitch';
        m_Capabilities=['isLanSwitch'];
        m_LastRecord=1;
}
( 3 record(s) : Transaction complete )
```

```
================================================END
```

This is ONE EXAMPLE of using the `dumpDiscoStatus.ovpl` and `dumpAgentProgress.ovpl` tools.

11. Once discovery completes, execute `$OV_BIN/ovet_topodump.ovpl {-? for usage)` to dump a summary and detailed listing of the results of Extended Topology discovery. For the purposes of our lab direct the output to a file such as:

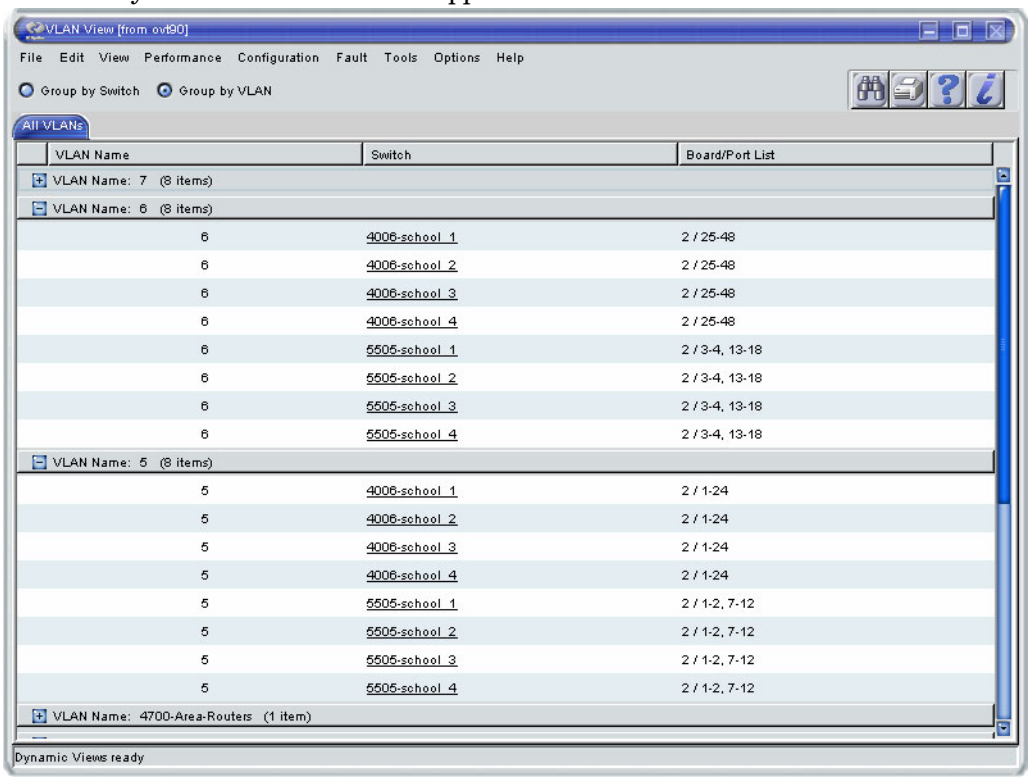
    ```
    ovet_topodump.ovpl -info > labdiscoFull-topo.txt
    ```

    Since you will execute additional discoveries under a variety of simulation scenarios ("broken discovery"), you will need to compare the results in subsequent labs.

12. Since the purpose of this lab is to discover the full topology in preparation for the following labs, use the VLAN view, Node View, and Neighbor Views as directed below to gain familiarity with this topology and the VLAN structure. Subsequent labs will simulate a variety of discovery issues, which will then require you to compare the "broken" discovery with the full discovery.

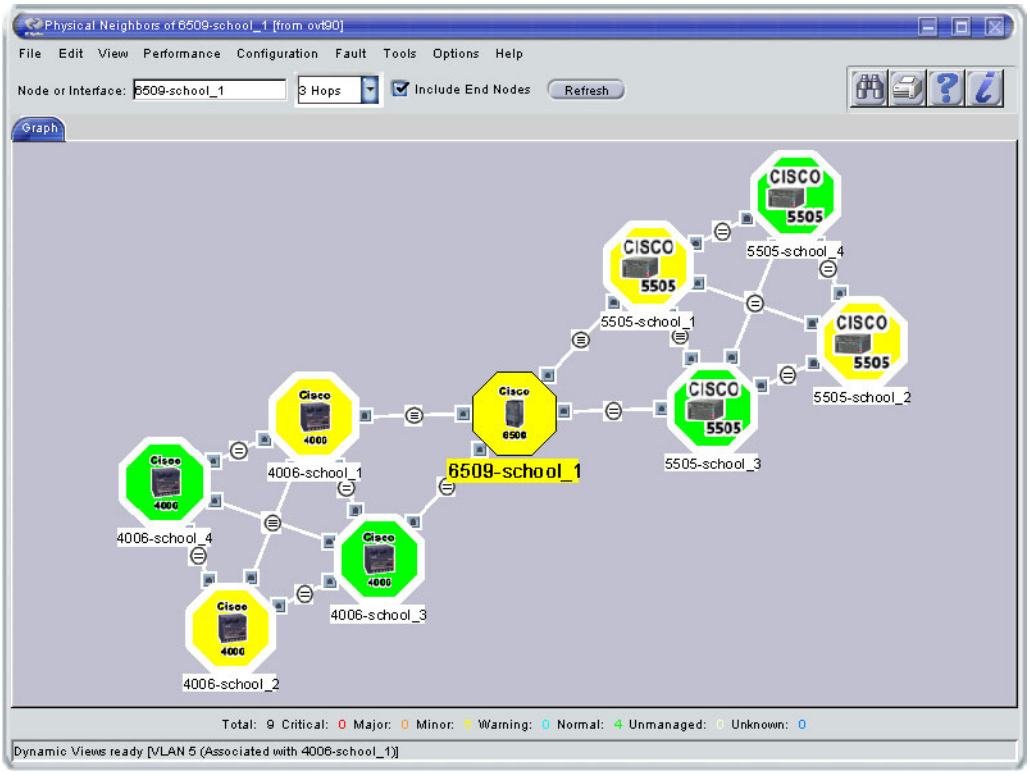
13. Once Extended Topology discovery completes, confirm correct and complete topology has been discovered, load the VLANs view from Home Base. Open the group for OAD Name: Non-overlapping default. You can get a better view by placing the mouse over the header VLAN Name, pressing the right mouse button, and selecting Ungroup.

    Confirm that your VLAN information appears as shown.



14. Now confirm the topology using the Neighbor View from 6509-school_1 using 5 hops and

Include End Nodes.



15. Within the neighbor view of 6509-school_1, use the Highlight VLAN feature and highlight VLAN 5 associated with 4006-school_1. Save the view using the `File:Save` option.

Once you have confirmed the topology discovered is similar to that documented above, inform the instructor you are ready to proceed with Lab E.

CAUTION Do not proceed to Lab E until instructed to do so. The MIMIC simulator must be configured for Lab E.

Lab E: Extended Topology SNMP Failure

CAUTION Successful execution of this lab depends on following the directions given by your instructor AT THE TIME those instructions are issued.

Do not proceed with this lab until the Instructor has said to continue. Changes to the MIMIC server are required.

Objectives:

- Understand the importance of SNMP access to network devices and the impact changes to SNMP access have on Extended Topology discovery and rediscovery.
- Gain additional experience using the support tools in an effort to understand Extended Topology discovery processes and agent behavior when snmp access and/or devices become unavailable.

At the completion of this module you will be able to:

- Properly diagnose when SNMP access failures (or device failures) adversely impact Extended Topology discovery.
- Run several Extended Topology discovery process monitoring tools and understand their results.
- Execute the database dump tools and understand the results.

Assumptions:

- Successful completion of Lab D.

Directions, Phase 1: Rediscovery

Using the base discovery you had for Lab D, wait for the instructor's go-ahead and rediscover the same environment.

1. Re-execute Extended Topology Discovery from Home Base or the command line.
2. Using similar procedures outlined in Lab D, use the tools to view the status/progress/output of the Extended Topology tools used before.
   a. Run several iterations of discovery,
   b. Run `dumpDiscoStatus.ovpl` several times during each discovery
   c. Run `dumpAgentProgress.ovpl` *{agentName}* for a variety of agents several times during a discovery
   d. `tail` the `ovet_disco.log` file
   e. Run `ovet_topodump.ovpl -nodeif`
3. When `ovstatus -v ovet_disco` reports discovery has STARTED (after the initial set of messages), run the tools in separate windows several times throughout discovery, for example:

   ```
   $OV_MAIN_PATH/support/NM/dumpDiscoStatus.ovpl

   $OV_MAIN_PATH/support/NM/dumpAgentProgress.ovpl details
   ```

   a. For example, in one terminal window run `dumpDIscoStatus.ovpl` to determine when the agent(s) are finished (reach m_State=4).
   b. In one or two other windows, execute different `dumpAgentProgress.ovpl` *{agentName}*

based upon the agents listed from the `dumpDiscoStatus.ovpl` command.

c. From the Topology Summary, select `Doesn't Respond to SNMP`. This shows which device(s) (if any) failed to respond to Extended Topology SNMP query and some suggested reasons for the failure.

4. Once discovery completes, execute `$OV_BIN/ovet_topodump.ovpl`.

   Prior to using the tool, `cd` to the directory you want to save the output into, for example:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_deploy/
   ```

   Determine what node(s) if any, is causing problems with discovery. Here are some hints:

   a. Determine which agent is causing the most delay in the discovery progress (`dumpDiscoStatus.ovpl`)

   b. Use the tools provided and pipe the output as shown in the EXAMPLE:

      `dumpAgentProgress.ovpl CDP| grep m_Unique | sort` (note, try different agents)

      Look for those IP address that do not have pairs. This indicates the agent has not completed the query to that device.

   c. Determine if any device failed an SNMP request. From the Topology Summary, select `Doesn't Respond to SNMP`.

5. Do a Node View, Show Nodes: All, Status >= Normal, IP Range: *.*.*.* What is different about this view?

   Answer:

   Nothing is different. It appears exactly as for Lab D.

6. Hold your mouse over 6509-school_1. What information is available in the mouseover?

   Answer:

   The same information as in Lab D. This information has been carried forward from the previous discovery. Unless you attempt a MIB query of 6509-school_1, you have no indication that something changed for this discovery cycle.

Directions, Phase 2: Initial Discovery

This time you will start with a clean database and observe the impact of a non-responsive device during initial discovery.

1. Change working directory to `$OV_CONTRIB/NNM7labs/Lab_deploy`:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_deploy
   ```

2. Re-setup the lab environment to start with a fresh database.

   ```
   setupLab_deploy.ovpl
   ```

3. Re-execute Extended Topology Discovery from Home Base or the command line.

4. Using similar procedures outlined in Lab D, use the tools to view the status/progress/output of the Extended Topology tools used before.

5. Once discovery completes, execute `$OV_BIN/ovet_topodump.ovpl`.

   Prior to using the tool, `cd` to the directory you want to save the output into, for example:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_deploy/
   ```

   Determine what node(s) if any, is causing problems with discovery. Here are some hints:

a. Determine which agent is causing the most delay in the discovery progress (`dumpDiscoStatus.ovpl`)

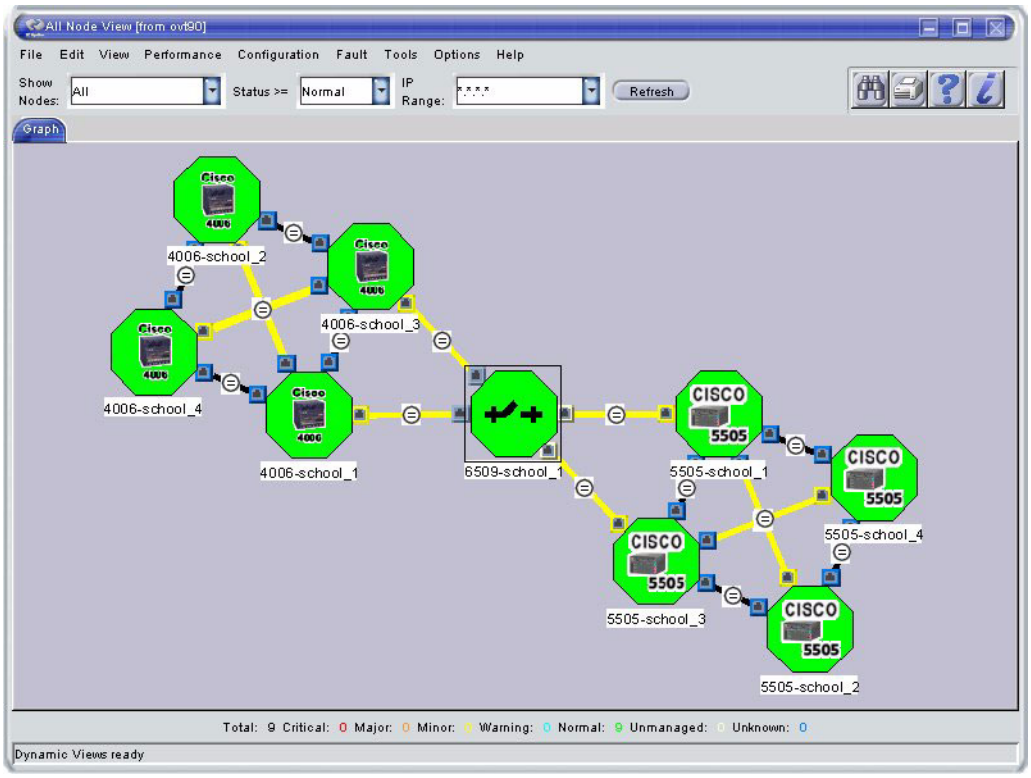
b. Use the tools provided and pipe the output as shown in the EXAMPLE:

`dumpAgentProgress.ovpl CDP| grep m_Unique | sort` (note, try different agents)

Look for those IP address that do not have pairs. This indicates the agent has not completed the query to that device.

c. Determine if any device failed an SNMP request. From the Topology Summary, select `Doesn't Respond to SNMP`.

6. Do a Node View, Show Nodes: All, Status >= Normal, IP Range: *.*.*.* What is different about this view?

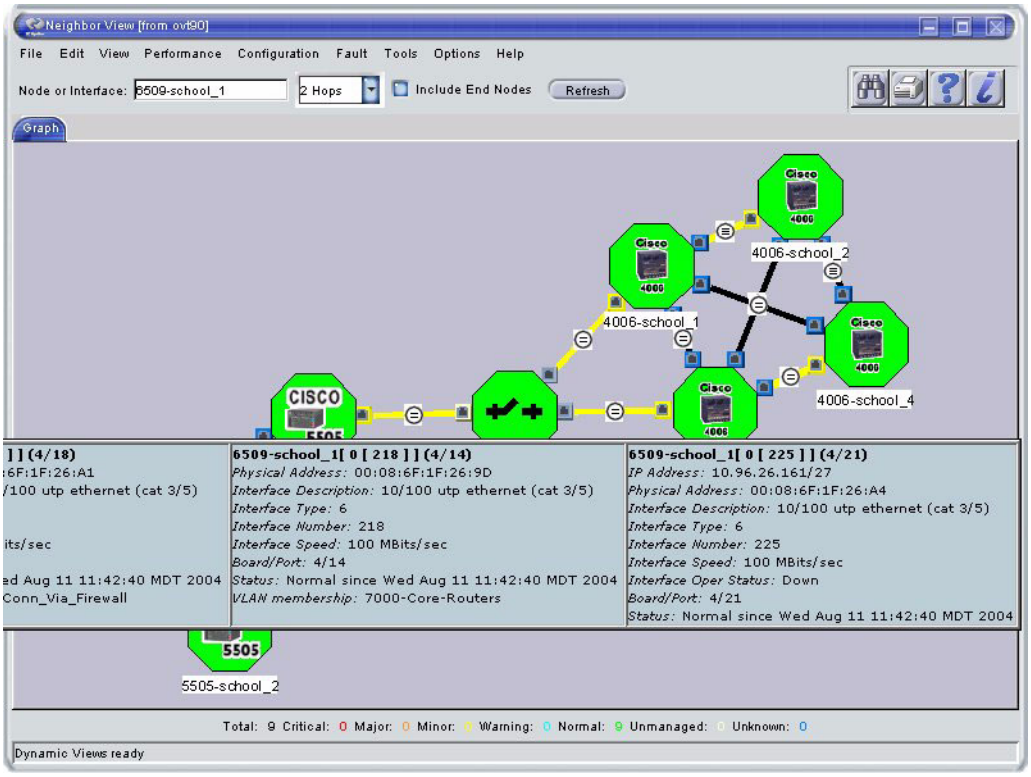


7. Hold your mouse over 6509-school_1 and its interfaces. What information is available?

Answer:

The mouseover indicates that the node was unresponsive during discovery. The additional

information is interpolated from the SNMP data gathered from the neighbors.



Lab E Extended Topology SNMP Failure Review Questions:

1. What is the impact to the Extended Topology discovery process when a device fails to respond to the initial Extended Topology SNMP access test?

   Hint: What were the differences between the Details agent and the other agents?

2. What was the affect to the views (Neighbor, Node, VLAN, etc.) as a result of incomplete access to all the devices discovered and managed by NNM?

3. How can you determine what devices fail to respond to an Extended Topology SNMP query?

   a. From the Topology Summary, select `Doesn't Respond to SNMP`. This shows which device(s) (if any) failed to respond to Extended Topology SNMP query and some suggested reasons for the failure.

a. from the Topology Summary, select `Doesn't Respond to SNMP`. This shows which device(s) (if any) failed to respond to Extended Topology SNMP query and some suggested reasons for the failure.

1. selecting `Doesn't Respond to SNMP`

a. from the Topology Summary, select `Doesn't Respond to SNMP`.

Lab G: Zone Discovery

CAUTION Do not proceed with this lab until the Instructor has said to continue.

Objectives:

- Understand correct behavior of Zone discovery with multi-zone devices.

At the completion of this module you will be able to:

- Implement correct Zone Discovery configurations.

Assumptions:

- Student has completed prior labs.

Directions

1. Execute the Extended Topology Configuration GUI from Home Base, Discovery Status tab.

WARNING **Do not change the configuration for anything in this browser window except as instructed below.**

2. Ensure that the Selection buttons are not checked for both
   "`Initiate a new discovery whenever Extended Topology is restarted`" and
   "`Enable recurring discover`".
3. Within the Discovery Zone Configuration box, position the cursor inside the text box besides "Members:"

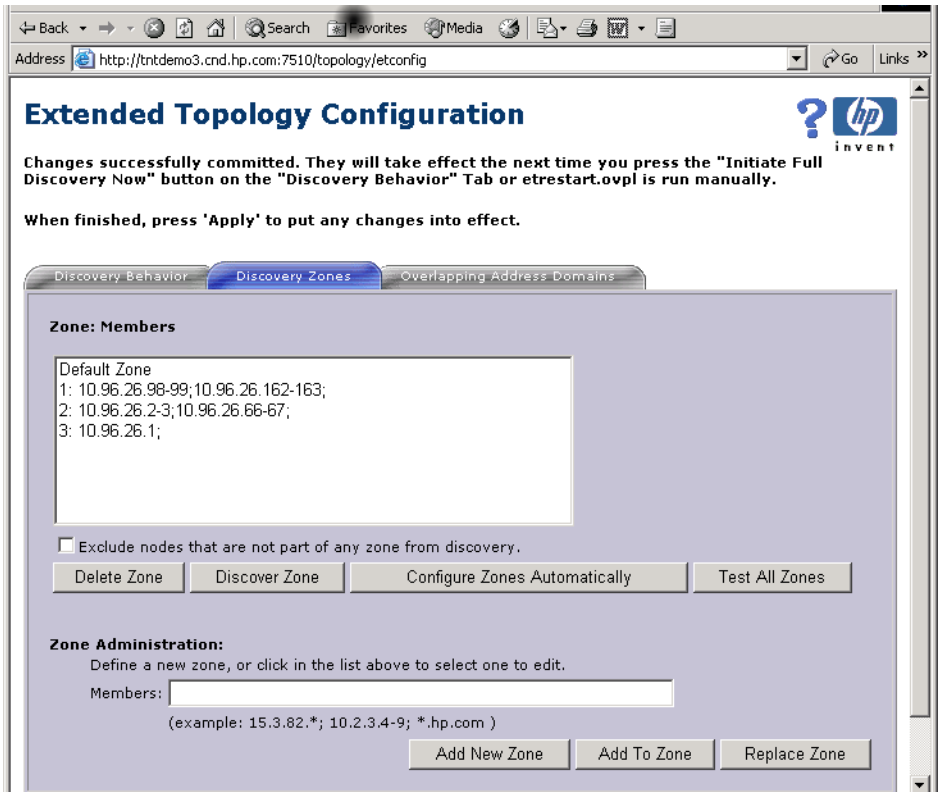
   Type in this box: `10.96.26.98-99;10.96.26.162-163` (Zone 1)
   then click [`Add New Zone`].

   Type in this box: `10.96.26.66-67;10.96.26.2-3;` (Zone 2)
   then click [`Add New Zone`].

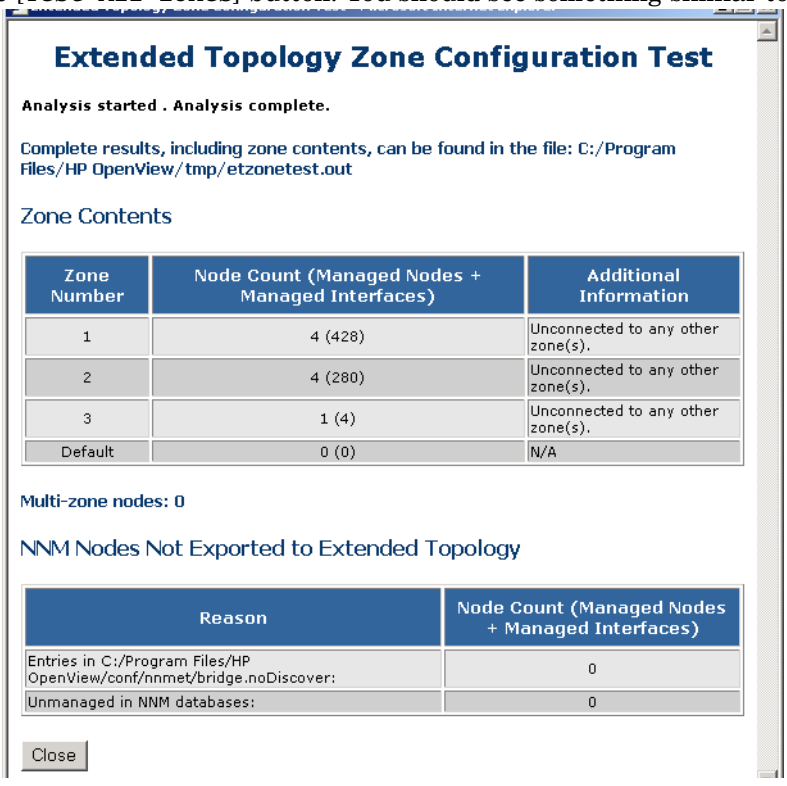
   Type in this box: `10.96.26.1;` (Zone 3)
   then click [`Add New Zone`].

   Click [`Apply`].

The browser window should be similar to this:



4. Now click the [`Test All Zones`] button. You should see something similar to:



Extended Topology Zone Configuration Test

Analysis started . Analysis complete.

Complete results, including zone contents, can be found in the file: C:/Program Files/HP OpenView/tmp/etzonetest.out

Zone Contents

Zone Number	Node Count (Managed Nodes + Managed Interfaces)	Additional Information
1	4 (428)	Unconnected to any other zone(s).
2	4 (280)	Unconnected to any other zone(s).
3	1 (4)	Unconnected to any other zone(s).
Default	0 (0)	N/A

Multi-zone nodes: 0

NNM Nodes Not Exported to Extended Topology

Reason	Node Count (Managed Nodes + Managed Interfaces)
Entries in C:/Program Files/HP OpenView/conf/nnmet/bridge.noDiscover:	0
Unmanaged in NNM databases:	0

Close

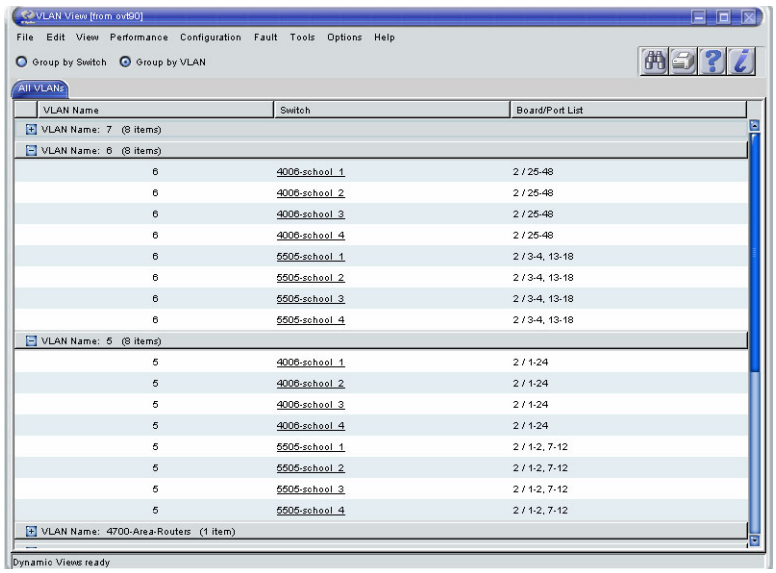
5. Click the "`Close`" button in the Zone Configuration Test window. Then click the "`Cancel`" button in the Extended Topology Configuration window.

NOTE If your system has discovered the local classroom network, you will see nodes in the "default" zone in addition to the nodes in the two zones we just configured and that are part of the Mimic Simulation Network. Recommendation: fix this before proceeding. Ensure New Node Discovery is OFF. Then re-run the `setupLab_deploy.ovpl` script.

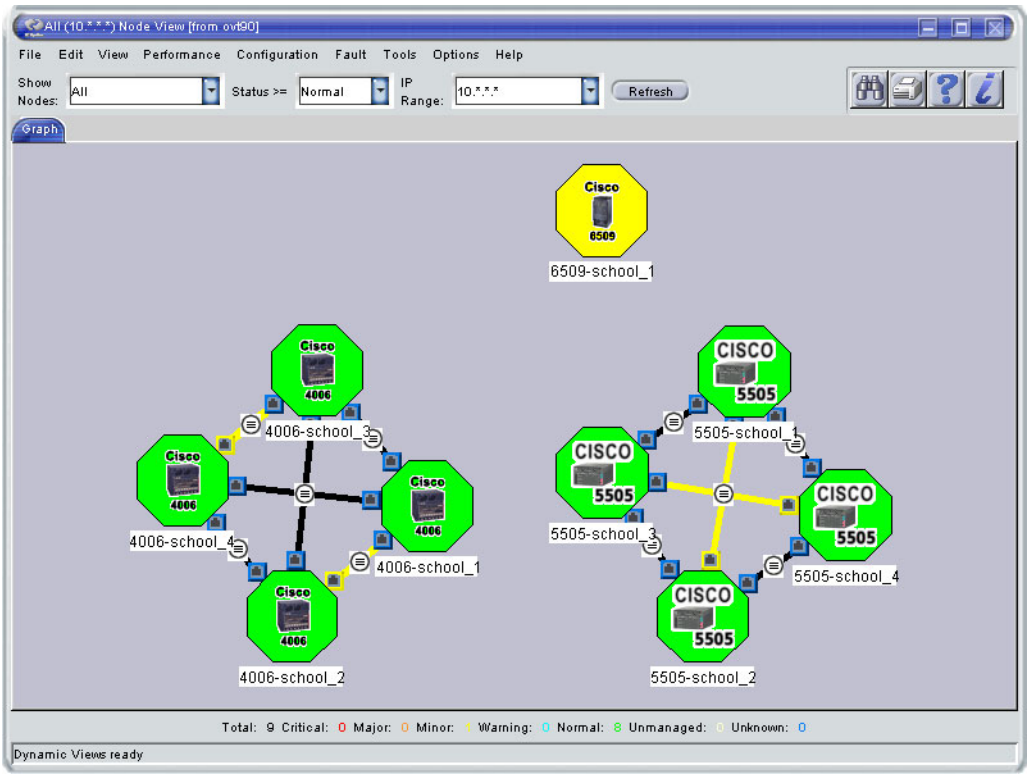
Repoll 6509-school_1 (`Fault:Network Connectivity->Poll Node`) or execute `$OV_CONTRIB/NNM7labs/Lab_deploy/bin/demandPoll3.ovpl`.

6. When the instructor has given you the go-ahead, initiate a full Extended Topology discovery.
7. Using similar procedures outline in previous labs use the tools to view the status/progress/output of the Extended Topology tools used before.
   a. Run `dumpDiscoStatus.ovpl` several times during each discovery
   b. Run `dumpAgentProgress.ovpl` *{agentName}* for a variety of agents several times during a discovery. Of particular interest will be the `CiscoSwitchSnmp` agent.
   c. `tail` the `ovet_disco.log` file
   d. At the completion of discovery run `ovet_topodump.ovpl`.
8. Confirm you have discovered the full topology similar to Lab D using the three zones. Execute a Node view as before.
9. Check the VLANs view to ensure full VLAN discovery WITHOUT overlap or duplicate contents.
10. Once discovery has completed, use the `ovet_topodump.ovpl` tool to isolate any difference.

    Hint: write the output to a file.
11. Use two windows side by side to compare Lab G and Lab D results as you did in earlier labs. Compare the Topology Summary page, and the results from the `ovet_topodump.ovpl`, etc.
12. Do a VLAN View. How does this differ from Lab E?



13. Do a Node View. How does it differ from Lab E?



14. Before proceeding to other labs, remove the zone configuration.

    a. Execute the Extended Topology Configuration GUI from Home Base, Discovery Status tab.

WARNING	**Do not change the configuration for anything in this browser window EXCEPT as instructed below**

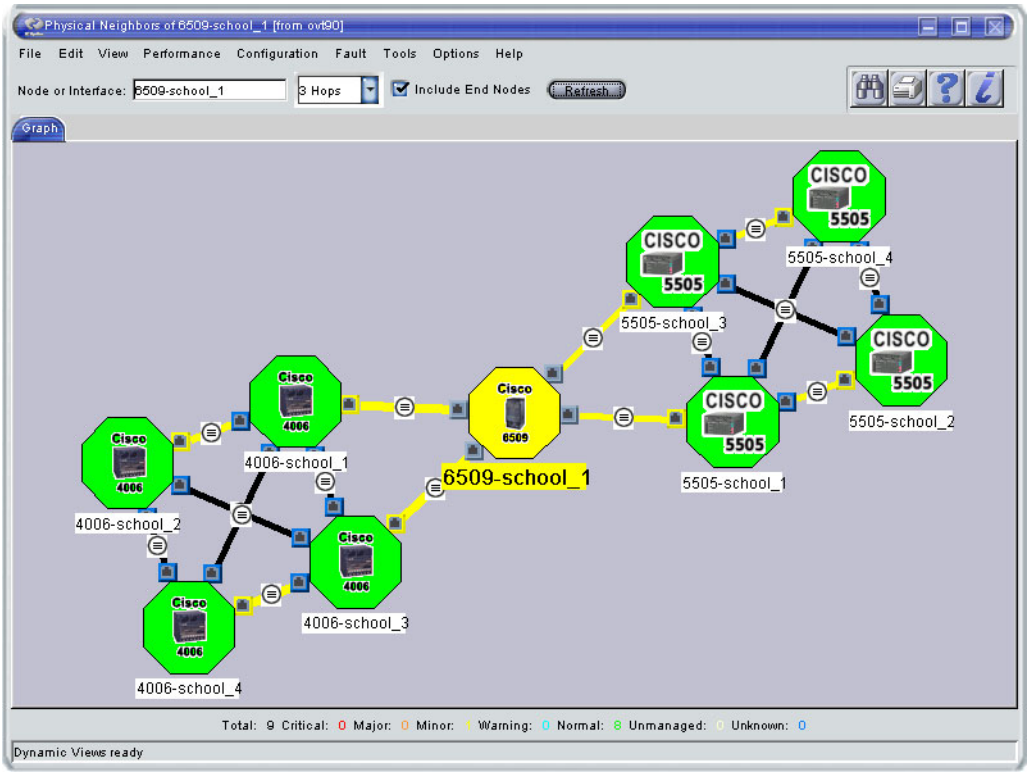
    b. Within the Discovery Zone Configuration box, and inside the `Zone: Members` list, select the first zone and then click [`Delete`].

    c. Select the remaining zones and click [`Delete`].

    d. Click [`Apply`].

    e. Confirm removal by reloading the browser window.

15. Configure the zones according to the overlap suggested in the module. Delete Zone 3 and add 10.96.26.1 to Zone 1 AND Zone 2.



Lab G Zone Discovery Review Questions:

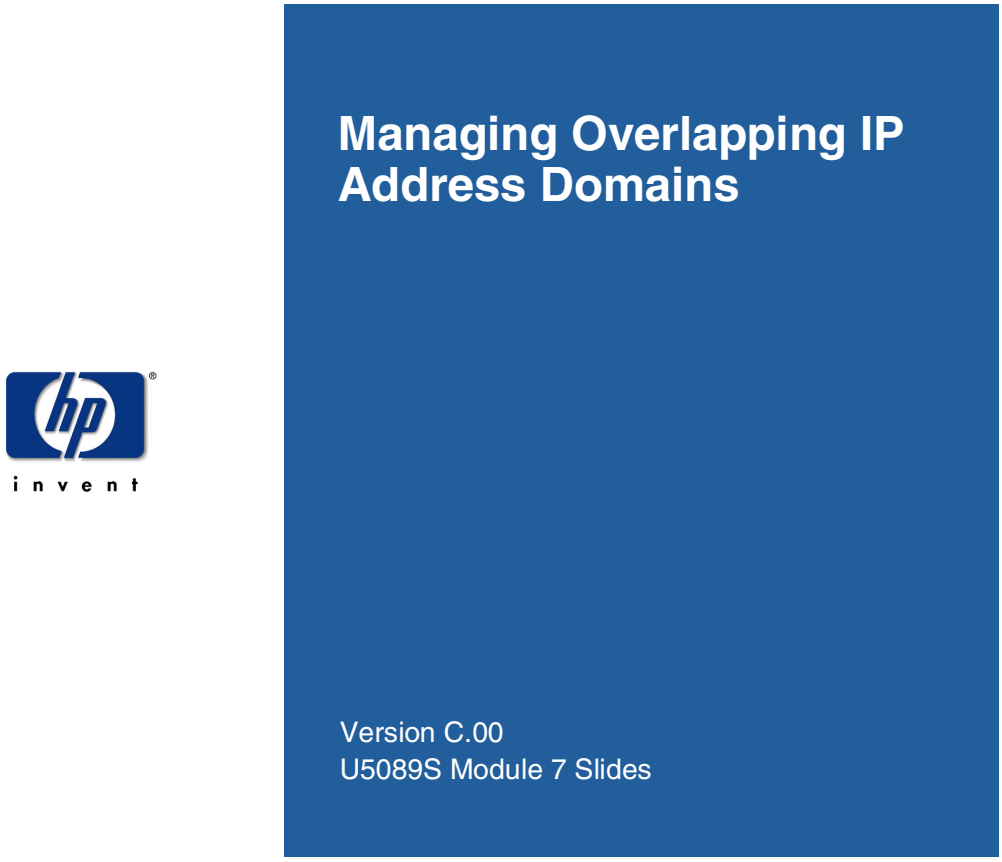
1. Describe the major differences between the zone configuration and the affect each had on the final results for topology?

Summary

1. Zone configuration requires insight and knowledge regarding the connectivity of the network.
2. Care must be taken in defining the specific devices that participate in a particular zone or zones.
3. The proper behavior for collecting and hence presenting information regarding Meshed, Switched VLANS eliminates duplication of data across zones.

7 Managing Overlapping IP Address Domains

Module Objectives



At the completion of this module the student will be able to:

- List the reasons for using private IP address domains.
- Describe how network address translation works.
- Identify various types of NAT.
- Describe how NNM manages NAT overlapping IP environments.
- List the required ports to open on a NAT device.
- Configure overlapping address domains for NNM Extended Topology discovery.
- Read NNM's Overlapping Address Domain views.
- Describe support of HSRP in OADs.
- Describe support of VLANs in OADs.
- Identify OADs in the Alarm Browser.

References

Using Extended Topology manual

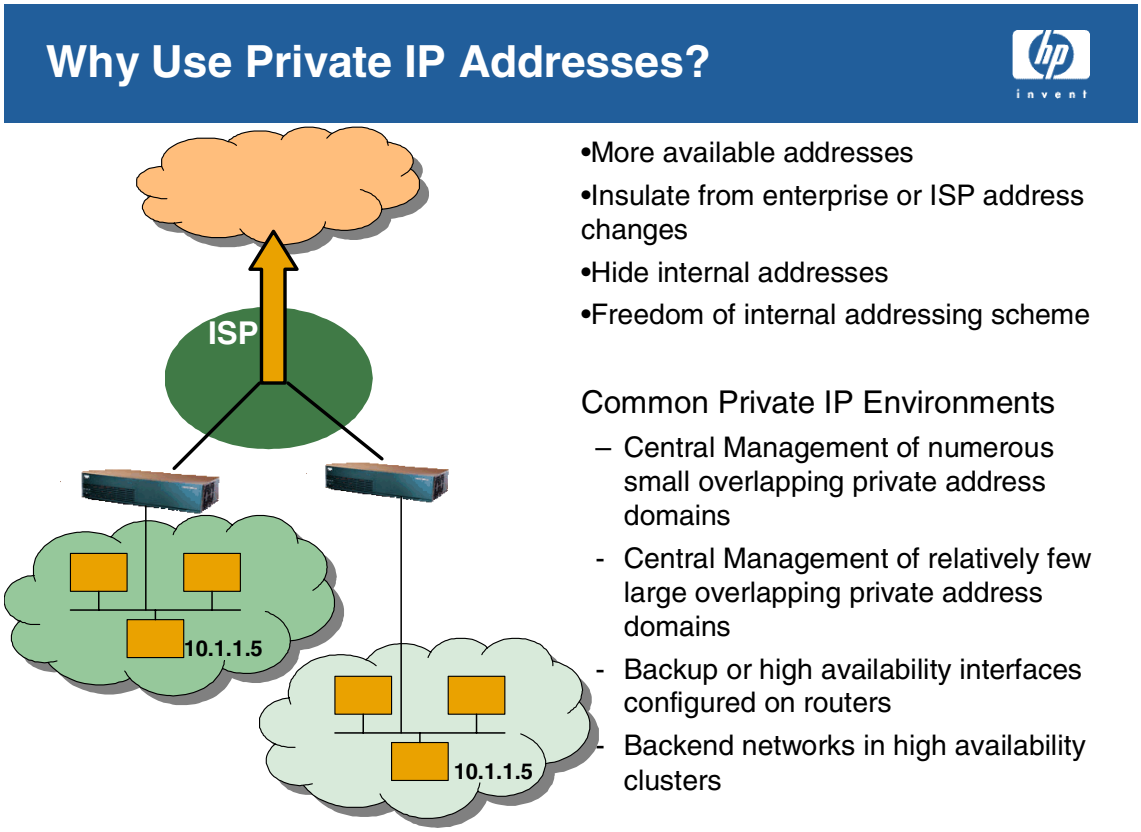
http://www.linktionary.com/n/nat.html

http://www.tcpipprimer.com/nat.cfm

http://www.ietf.org/rfc/rfc1918.txt

http://www.ietf.org/rfc/rfc2663.txt

Why Use Private IP Addresses?



The need to conserve IPv4 addresses led to the development and deployment of IP addresses in the private internet address space. To understand details about the private internet address space, see RFC1918.

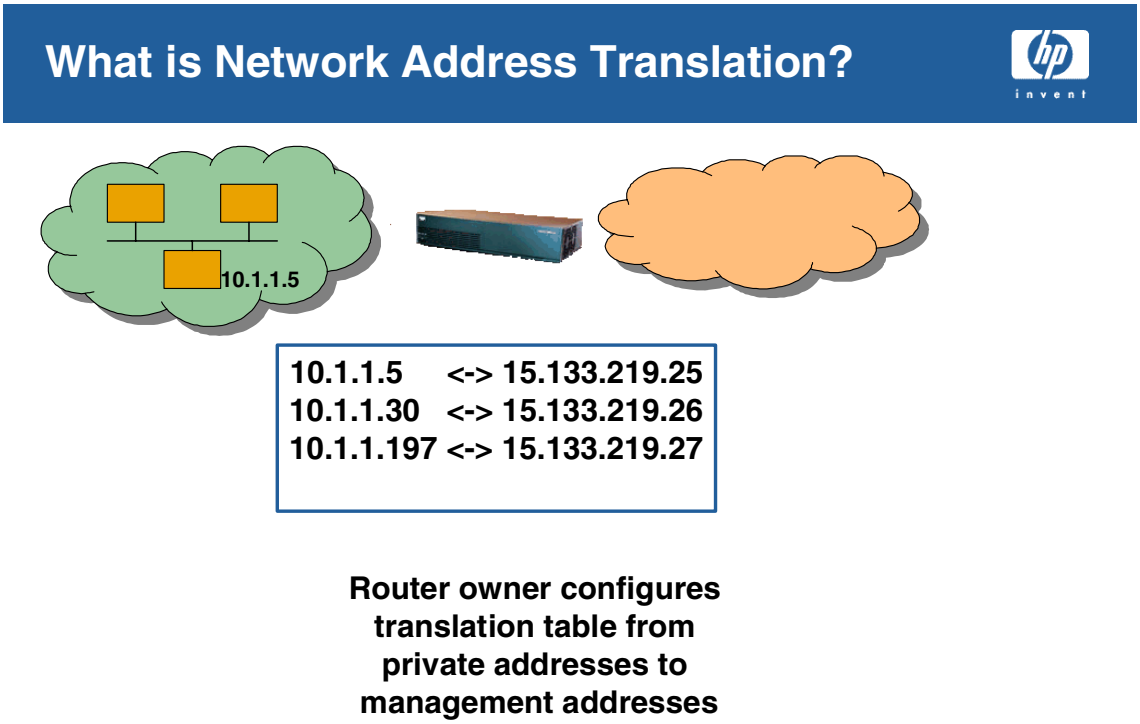
Such networks are commonly found in service provider environments, where they manage customers with overlapping IP addresses. Customers choose the RFC1918 address spaces for a variety of reasons, including

- perceived security
- isolation from renumbering when they change providers
- shortage of routable IPv4 addresses

The following table shows IANA-allocated, non-routable, IP address schemes (RFC 1918).

Address Class Range	Network Address Range
A	10.0.0.0 - 10.255.255.255
B	172.16.0.0 -172.31.255.255
C	192.168.0.0 - 192.168.255.255

What Is Network Address Translation (NAT)?



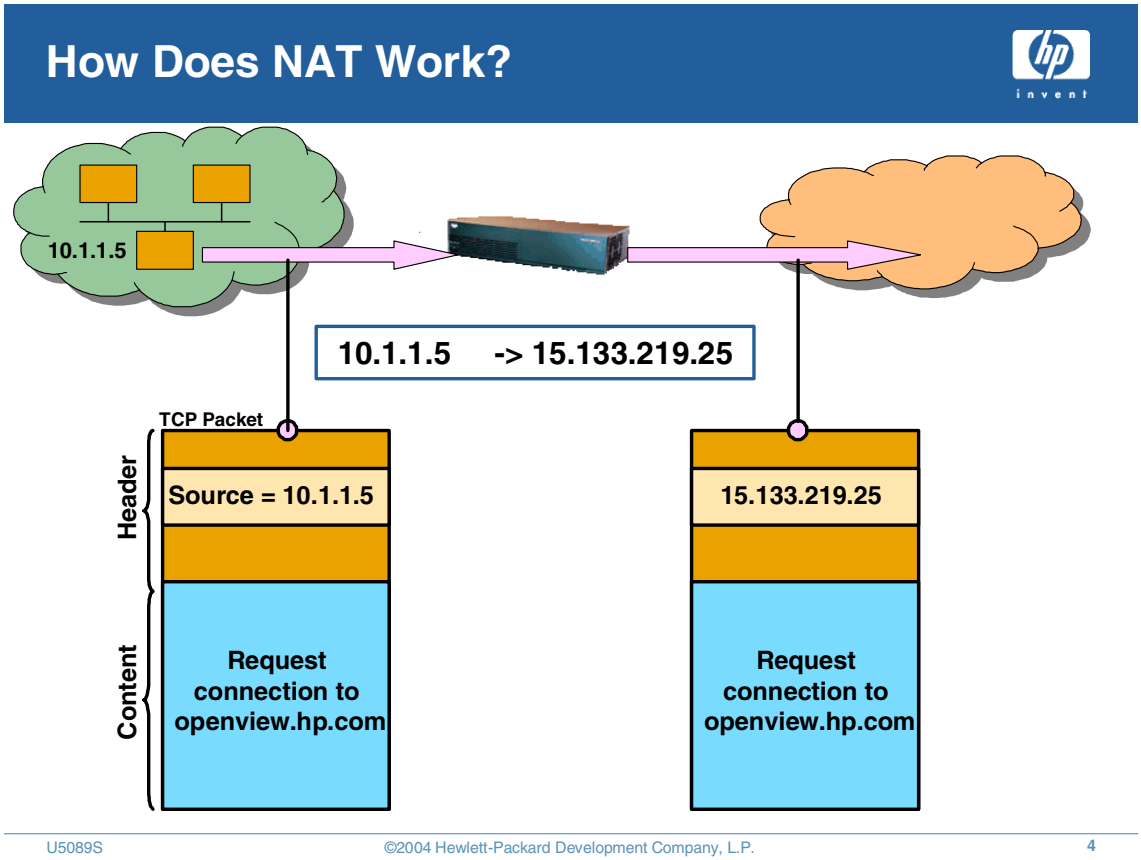
From http://www.ietf.org/rfc/rfc2663.txt

> *Network Address Translation is a method by which IP addresses are mapped from one realm to another, in an attempt to provide transparent routing to hosts. Traditionally, NAT devices are used to connect an isolated address realm with private unregistered addresses to an external realm with globally unique registered addresses.*
>
> *The need for IP Address translation arises when a network's internal IP addresses cannot be used outside the network either because they are invalid for use outside, or because the internal addressing must be kept private from the external network. Address translation allows... hosts in a private network to transparently communicate with destinations on an external network and vice versa.*

NAT hides the internal addresses by centralizing them to a single router. By allowing neighboring networks to use incompatible IP addressing plans, NAT allows multiple hosts on the inside network to simultaneously access remote networks using one or multiple IP addresses.

The NAT device is a router that transparently routes packets across address realms. NAT software is typically found in newer Internet routers and almost always used in firewalls and proxy servers. End nodes on each side of the NAT device are unaware of the address translation.

How Does NAT Work?

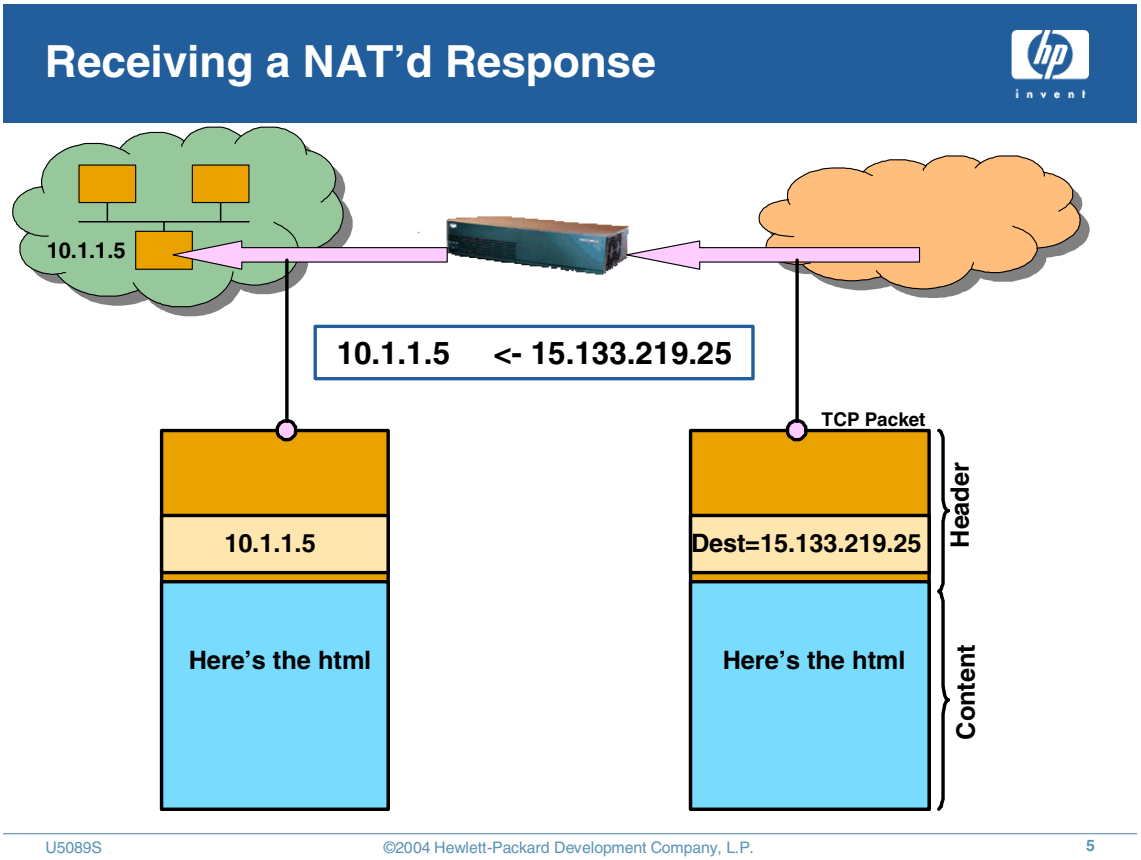


From http://www.ietf.org/rfc/rfc2663.txt

> *NAT devices attempt to provide a transparent routing solution to end hosts trying to communicate from disparate address realms. This is achieved by modifying end node addresses en-route and maintaining state for these updates so that datagrams pertaining to a session are routed to the right end-node in either realm. This solution only works when the applications do not use the IP addresses as part of the protocol itself. For example, identifying endpoints using DNS names rather than addresses makes applications less dependent of the private addresses that NAT chooses and avoids the need to also translate payload contents when NAT changes an IP address.*

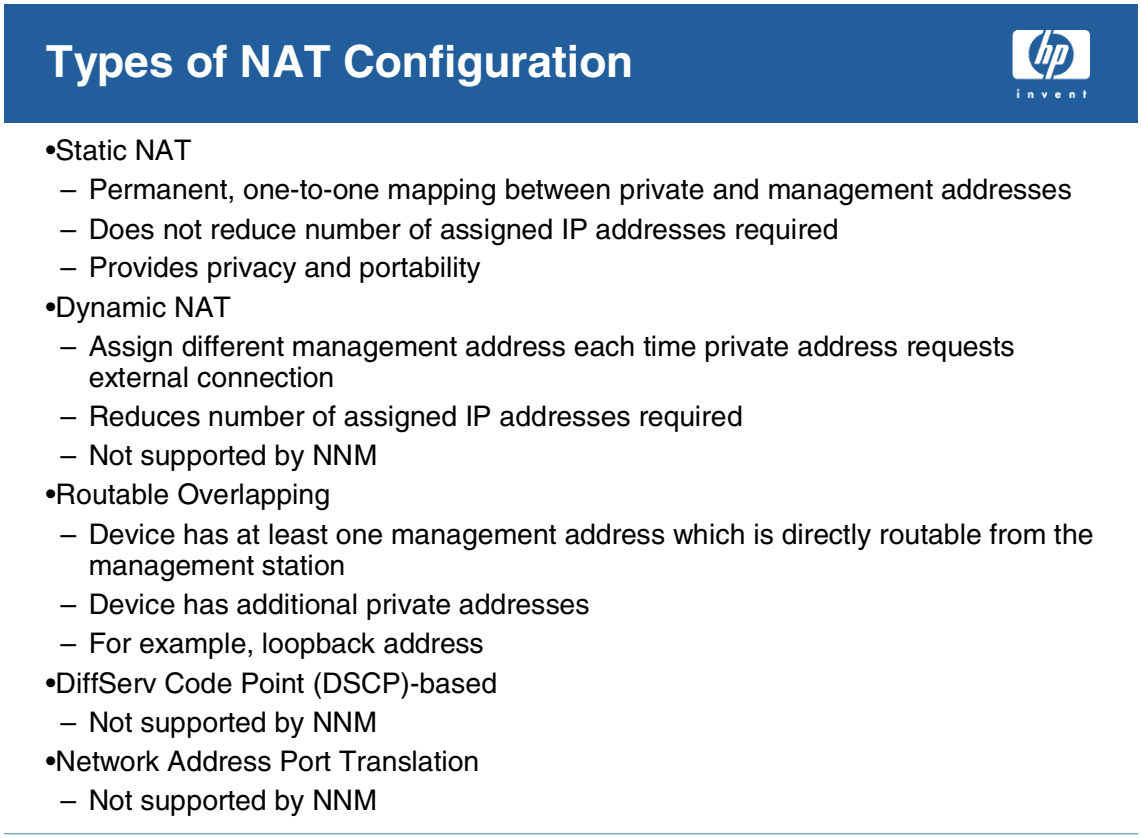
The NAT device goes into the packet header and replaces the private IP address with the translated address. It must also recalculate and replace the checksum(s) with the new content in the packet.

Receiving a NAT'd Response



For incoming packets, the NAT device replaces the management IP address with the private IP address used by the node in the local domain. It also recalculates and replaces the checksums.

Types of NAT Configuration



Routers may support several types of deployment of NAT configurations:

- **Static NAT**: in this model, managed devices have IPv4 address mapping set up in a way that each device is reachable via the mapped address from the management station. This is done by setting up NAT on the edge gateway connected to the managed network.

  Not all overlapping addresses need to have a mapping, but each device must have one address that is reachable using the NAT mapping. The routable addresses should be static for each device – that is, the same routable address can't be reused to reach different devices.

- **Dynamic NAT**: Each time a private address requires access to the external network, it is translated to an available management IP address from the pool, similar to DHCP. NNM does not manage devices that are only available using dynamic NAT.

- **Routable Overlapping**: Devices may have one routable address, without using NAT. Other addresses may be overlapping. A common convention is to configure the loopback address on network devices such that they are reachable from the management station.

- **DSCP-based**: in this model, policy based routers are deployed in the network so that devices with overlapping addresses can be reached from the management station by marking the diffserv code point (DSCP) on packets sent to them. DSCP is not currently supported by NNM.

- **Network Address Port Translation**:

  In Network Address Port Translation (sometimes called Port Address Translation), the

translating router maps many private addresses to a single public address by assigning each one a different port within the public address. NAPT is not supported.

NNM Management Features

- Intelligently handles configuration of address mapping for managed devices with overlapping IP addresses
- Discovers connectivity with a overlapping IP address network domain – includes L2 connectivity, VLANs
- Status monitoring via ICMP or SNMP for managed devices
- Fault analysis of failed devices
- Visual access to customer networks via dynamic views (Shows both the private IP addresses and management addresses)
- Enhanced event model to add a tag to every event identifying the unique private network being managed (ex. “Customer A”, “Hong Kong Sales” etc.)

 ©2004 Hewlett-Packard Development Company, L.P.

NNM Extended Topology component supports management of networks with overlapping IPv4 addresses. Extended Topology provides support for connectivity discovery, monitoring and event handling from networks with overlapping IP addresses. It identifies events by the unique device and domain from which they originate.

Service providers do not want a distributed solution, where a separate collection station would be deployed for each customer because of cost and maintenance issues. They want to manage networks with overlapping IPs from a single management station.

NNM allows operators to receive events from overlapping IP domains without being confused about the correct source and to navigate easily to views of that specific domain – from a single management station. This reduces MTTR by getting to the device quickly and understanding which customer is affected and prioritizing accordingly.

How Can NNM Manage Through NAT?

- Only for static NAT environments

- Given a list of management IP addresses
- Given all static mappings of those management IP addresses to private IPs
- NNM Extended Topology component queries each management IP device for its connectivity information
  - Router transparently changes packet headers to private IPs and back
  - Content returned (which interface connects to which address) is in private IP address space
  - NNM maps content using the configured mapping table
- Creates node and interface objects in the Extended Topology database
- netmon and NNM's topology database are not involved.

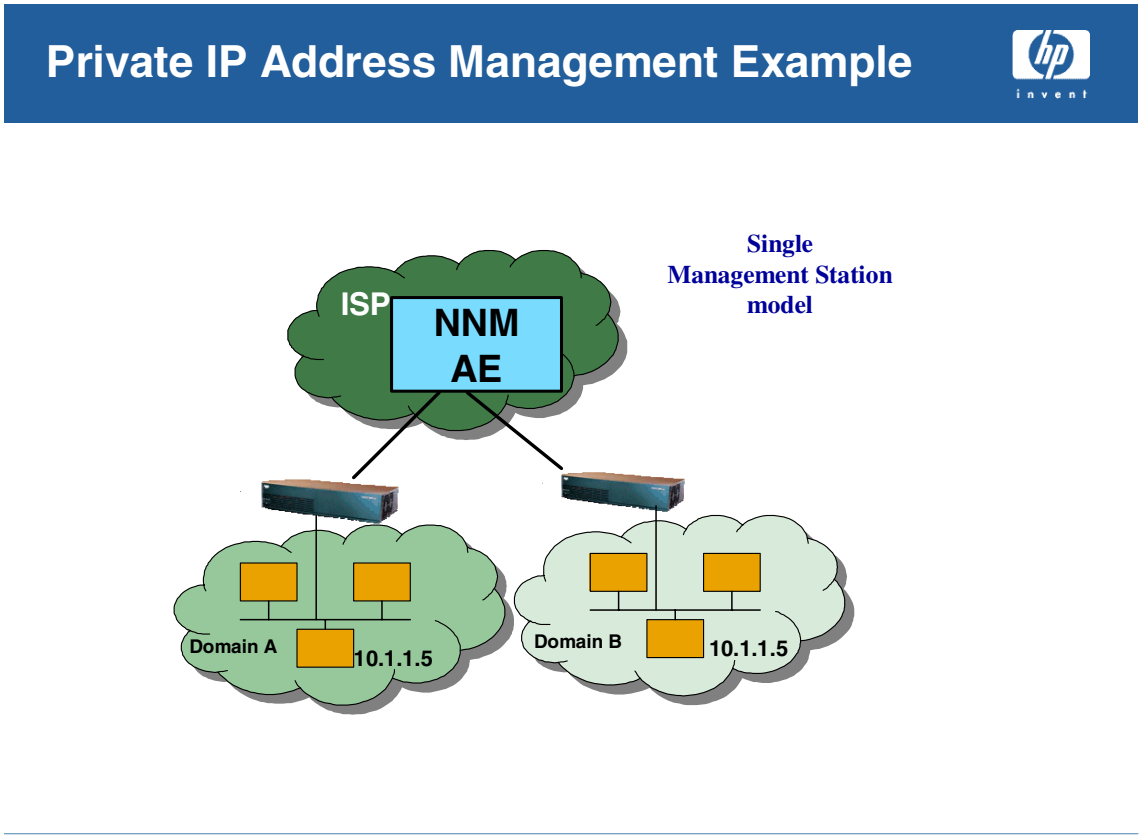
 ©2004 Hewlett-Packard Development Company, L.P.

NNM Configuration should use the management IP address to set community strings.

Understanding Overlapping Address Domain Status Information

The Advanced Problem Analyzer polls Overlapping Address Domain (OAD) addresses and reports device status information in the OAD view. It also displays alarms in the Alarm Browser. See the `trapd.conf` reference page in NNM's online help (or the UNIX manpage) for more information.

Private IP Address Management Example



1. Company ISP manages the networks for companies AAA and BBB, which have overlapping private address spaces.

   a. The router owner (either the ISP or the client company) sets up Static NAT tables on routers at the edge of their environments which provide unique IP addresses that can be used to communicate to the overlapping private addresses.

   b. Company ISP uses NNM AE to solve their problems. They configure their overlapping IP environment. They create the files `$OV_CONF/nnmet/dupip/AAA/dupip.conf` and `$OV_CONF/nnmet/dupip/BBB/dupip.conf` which tell Extended Topology what to call these environments and how to map addresses from private to public.

   c. Company ISP then gets a list of addresses to manage for Company AAA and BBB. They put a file called `dupip.seed` with these addresses into the same directories as the `dupip.conf` files. The administrator activates the changes through the Extended Topology configuration GUI, then starts the discovery of the environments.

      After discovery completes, Company XYZ can use dynamic views and alarm browsers to manage the two networks.

2. Company ISP now needs to manage company CCC, which again has overlapping IP addresses.

   a. The administrator at ISP creates the `dupip.conf` file for CCC, and adds the `dupip.seed` file with addresses to manage. This time, they go to the Extended Topology configuration

GUI and press the "`Refresh Configuration and Activate Changes`" button to start using this configuration.

b. The administrator then needs to find the new OAD for company CCC in the list of zones. They check the zone, and do an incremental discovery.

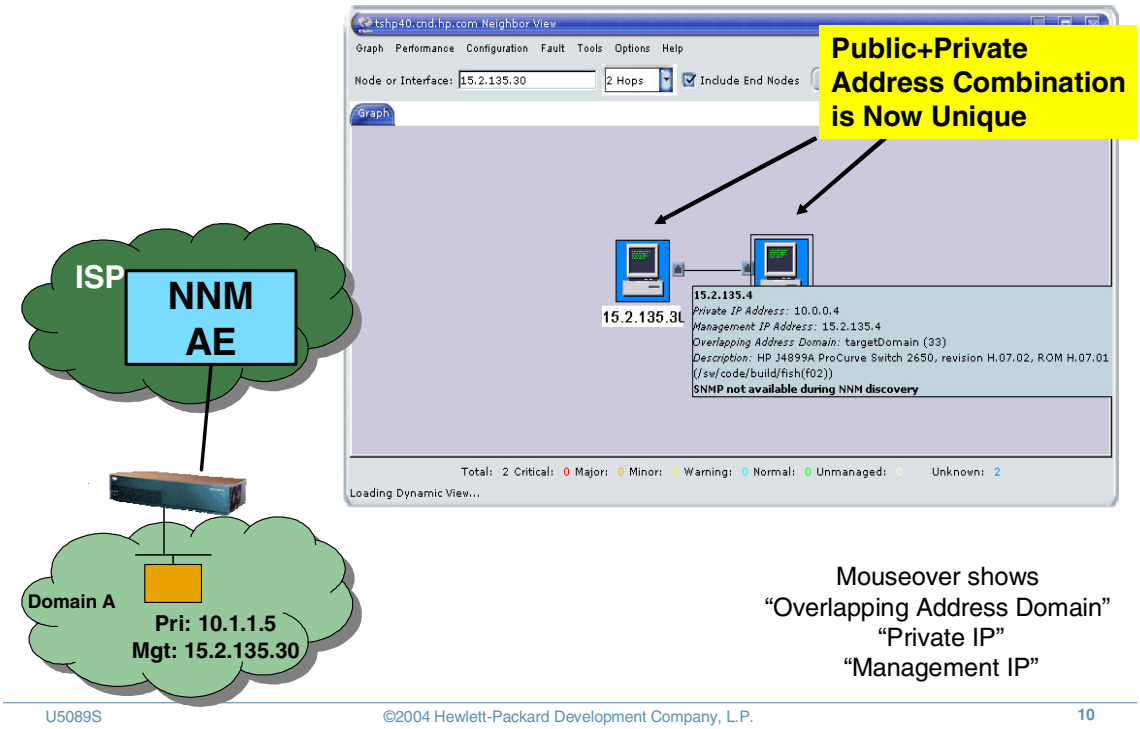
c. When discovery is complete, the dynamic views will show the new OAD and its managed devices.

3. Company ISP needs to manage another node for company BBB. They edit the seedfile for BBB and add the IP address, and then clicks "`Refresh Configuration and Activate Changes`" in the Extended Topology configuration GUI. The next step is to select the zone for BBB and do an incremental discovery.

NOTE If you delete or change an OAD ID, you must initiate a full discovery to recreate the data in the Extended Topology database with the modified IDs.

NNM Shows Overlapping Addresses

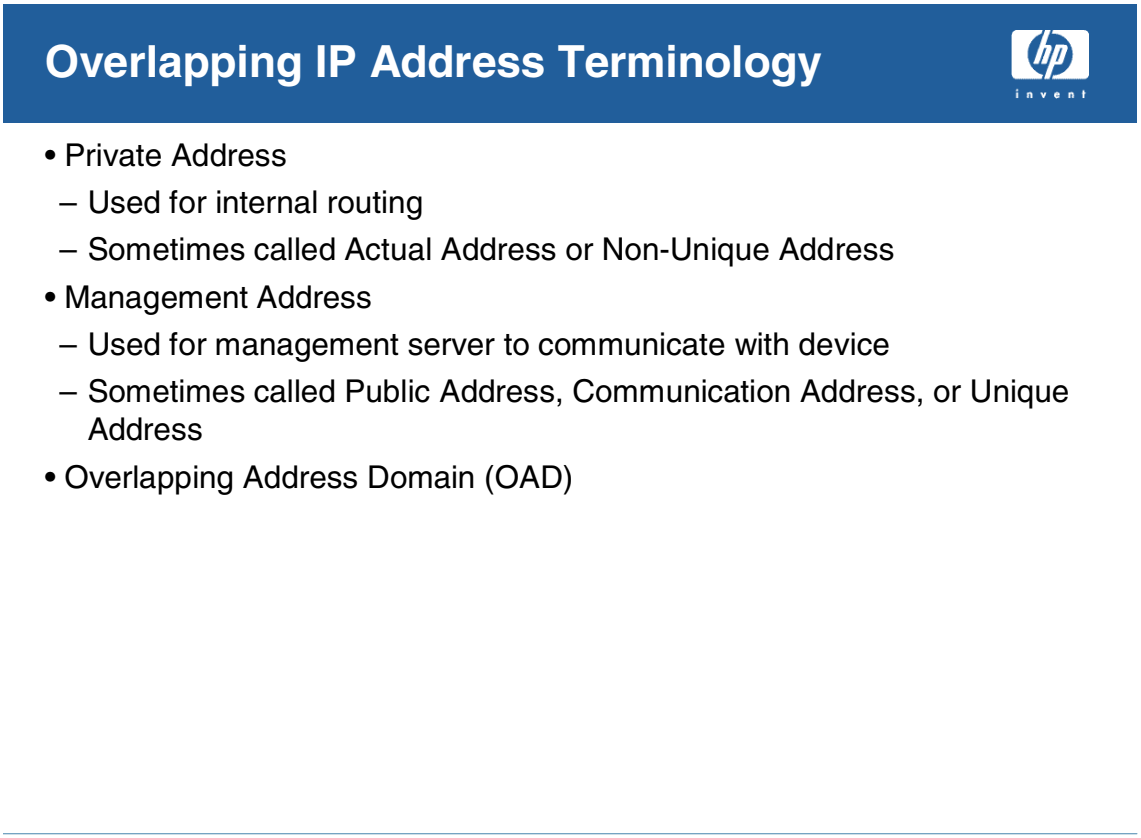
NNM Shows Overlapping Address Domains



Mouseover shows
"Overlapping Address Domain"
"Private IP"
"Management IP"

©2004 Hewlett-Packard Development Company, L.P.

Overlapping IP Address Terminology



The HP terminology for overlapping IP addresses is the following:

Private address: this is the address used while routing packets on the device. That is, a packet would have this as the source IPv4 address if it was emitted from the interface. SNMP queries to the device, e.g., to the IpCidrForwardTable, would return private addresses in the payload.

Management address: this is the address the management server uses to communicate with the device. This address must be unique across all IPv4 addresses visible to the instance of the management station. In case a private-IP network is nested within another private-IP network, the management address is relative to OAD=0, not the network that is immediately containing the private-IP network.

Overlapping Address Domain: the terminology used to denote a consistent set of addresses. We assume that the end-to-end assumption holds within this domain. An OAD has a unique, positive, unsigned 32-bit integer identifying it, called the Overlapping Address Domain (OAD). Multiple zones can belong to a single OAD.

In addition, other terminology that can be used to describe the pair of addresses associated with overlapping IP addresses. These refer to the external network / internal network view of the IP address. These pairs are named:

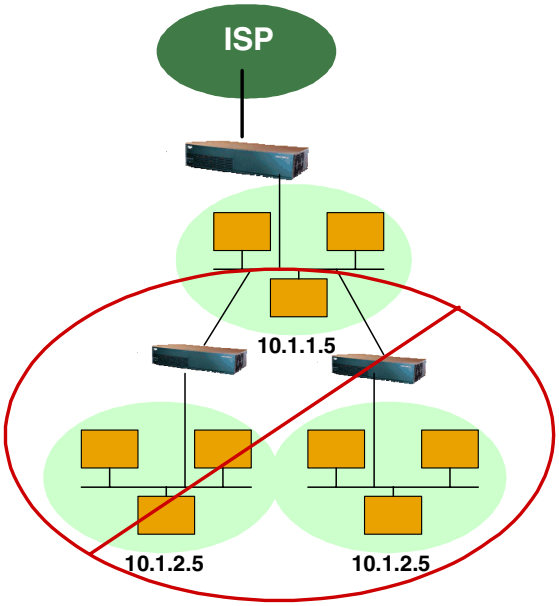
- public address / private address (typical in a NAT environment)

- communication address / private address
- unique address / non-unique address

Overlapping IP Support Limitations

Overlapping IP Support Limitations

- Processes unaware of OAD
  - netmon (discovery, community strings)
  - IPv6
  - OSPF
  - MPLS
- Cannot have nested OADs



U5089S ©2004 Hewlett-Packard Development Company, L.P. 12

The following functionality is not OAD IP aware:

- `netmon` functionality, including device discovery and community string discovery

  If the user does not need device discovery for the address domain in which the management station is located (OAD=0), `netmon` can be turned off. In most OAD IP deployments, NNM topology will not be aware of the device.

  Other NNM functionality – only Extended Topology has OAD capabilities
- Overlapping IPv6 addresses
- OSPF support: If OSPF is deployed in IPv4 networks with overlapping addresses, these are different OSPF processes across spaces with overlapping addresses. That is, the OSPF areas must be distinguished by OAD. Since OSPF data does not flow into the Extended Topology database, it is not included in OAD IP management in NNM 7.0.
- MPLS support. MPLS management is not OAD sensitive.
- Nested OAD scenarios will not be supported using a single station. If there are OAD islands within a network with overlapping addresses (that is, the gateway to the island is not connected with the OAD=0 network), the management station must have access to each island.
- Islands, or overlapping address domains may not communicate directly with each other.

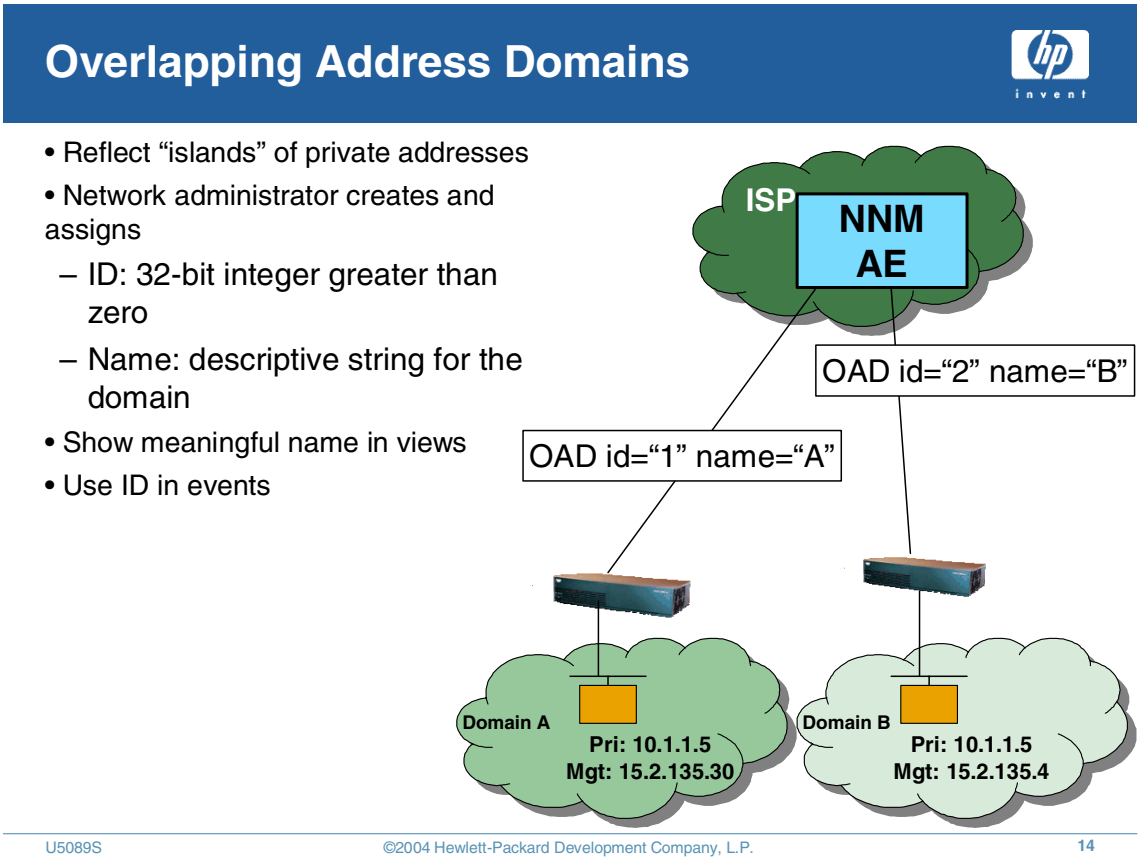
Traffic Flows To/From Customer Networks

- ICMP
  - ECHO REQUEST/REPLY
  - PORT/HOST/NET UNREACHABLE
  - TTL EXCEEDED
- SNMP
  - UDP Port 161
  - UDP Port 162
- Syslog
  - UDP Port 514
- Optionally
  - ssh (TCP port 22)
  - telnet (TCP port 23)
  - Traceroute: high UDP ports (configurable)

 ©2004 Hewlett-Packard Development Company, L.P.

If there is a firewall between the NNM management station and any nodes tied to a specific OAD, you must configure this firewall to pass ICMP (port 7), SNMP (port 161), and SNMPTRAP (port 162) packets between the management station and the managed nodes. If you want to log on to any OAD nodes (telnet) using Dynamic Views from the management station, you will need to open port 23 as well. This minimal relaxation of the firewall is required to support network management through it. By restricting the communications to the management station only, there is virtually no loss of security.

Overlapping Address Domains



A key concept for managing overlapping address using NNM is that of an **Overlapping Address Domain** (OAD). An OAD denotes a set of IPv4 addresses that are internally non-overlapping, consistent and (typically) directly routable from each other without manipulation of the IPv4 header. For example an OAD might represent the private IP addresses of a small business, specific department, or a specific workgroup in a large company. It corresponds to the notion of Address Realm in RFC 2663.

In NNM, an Overlapping Address Domain is denoted by an unsigned 32-bit integer called the Overlapping Address Domain (OAD) ID. The Overlapping Address Domain value of 0 is reserved for the address domain for the network to which the management station is directly attached.

Configuration Overview

Configuration Overview

- Preparation
  - SNMP configuration based on management addresses
  - Static NAT mapping for mapped addresses on router
  - DNS set up based on routable addresses

1. Create directory for each domain
2. Create **`dupip.conf`** file for each
3. Create **`dupip.seed`** file for each
4. Check syntax
5. Load into Extended Topology configuration GUI
6. Deploy changes to running configuration

1. For each OAD that you define, create a separate directory.
2. Within each new directory create a `dupip.conf` file and add commands to this file that define the OAD.
3. Create a `dupip.seed` file for each OAD you want to manage.
4. Run the `ovdupip` command to make sure your file entries are syntactically correct. (The Extended Topology configuration GUI also tests the syntax.)
5. Go to the Extended Topology Configuration web page.
6. Select the Overlapping Address Domains tab.
7. Click [`Refresh Configuration and Activate Changes`] to read any changes or additions you made. If there is an error in the configuration, the GUI displays the error message and the configuration is not applied. Successful changes or additions will affect the next discovery cycle.

WARNING **Do not allow `netmon` to discover and forward the management address of OAD (NAT'd) devices that you place in the seed file. When the OAD component reads the seed file and attempts to create an object in the Extended Topology database, the netmon-exported one already exists and causes a catastrophic failure on the insertion.**

Any device listed in an OAD seedfile which `netmon` might discover should be added to the `netmon.noDiscover` file to avoid sending it to Extended Topology OAD discovery. Use `netmon.noDiscover` rather than `bridge.noDiscover` so that the devices do not appear (without full connectivity information) in the ovw display.

Configure Overlapping Address Domains

Configuring OADs in dupip.conf

- Create a directory for each domain in
  - UNIX: **$OV_CONF/nnmet/dupip**
  - Windows **%OV_CONF%\nnmet\dupip**

- In each domain, create **dupip.conf**

```
OverlappingAddressDomain id="2" name="myCompany"
  Mapping privateIP="10.1.2.*" managementIP="172.1.2.*"
  Mapping privateIP="10.2.3-4.*" managementIP="172.2.3-4.*"
Routable managementIP="162.1.2.*"
```

- Use the router's NAT table to create the **dupip.conf** entries

©2004 Hewlett-Packard Development Company, L.P.

The key to setting up the address mapping is the dupip.conf configuration file. Each Overlapping Address Domain (OAD) needs to have a `dupip.conf` file which specifies the mappings for that OAD. The configuration files are stored in sub-directories under `$OV_CONF/nnmet/dupip` on UNIX or `%OV_CONF%\nnmet\dupip` on Windows. (The sub-directory name can be anything.)

1. For each OAD that you define, create a separate directory beneath the following directory:

   - Windows: `%OV_CONF%\nnmet\dupip`
   - UNIX: `$OV_CONF/nnmet/dupip`

   This is NOT the set of configuration files that are used at run-time. The user configuration files are transferred to the run-time directory when you click [`Refresh Configuration and Activate Changes`] in the GUI.

   There is no specific naming convention, so you can choose a friendly name for each directory. As an example, suppose you have a group of private IP addresses in an OAD for a store called Red. You would create one of the following directories for this OAD:

   - Windows:`%OV_CONF\nnmet\dupip\red`
   - UNIX: `$OV_CONF/nnmet/dupip/red`

2. Within each new directory, `red` in this example, you need to create a `dupip.conf` file and add commands to this file that define the OAD. You would create the following file for this OAD:

- Windows: `%OV_CONF\dupip\red\dupip.conf`
- UNIX:`$OV_CONF/dupip/red/dupip.conf`

```
$OV_CONF/nnmet/dupip/red/dupip.conf:

OverlappingAddressDomain id="5" name="Red"

    Gateway IP=”125.1.1.1”

    Gateway IP=”128.1.1.1”

    Routable managementIP=”15.1.1.1”

    Routable managementIP=”15.2.*.*”

    Mapping privateIP="10.1.1.1" managementIP="16.1.1.1"

    Mapping privateIP="10.2.1.1" managementIP="17.1.1.1"
```

If you are managing additional domains for a Blue store, your configuration might look like the following:

```
$OV_CONF/nnmet/dupip/blue/dupip.conf:

OverlappingAddressDomain id="10" name="Blue"

    Gateway IP=”234.1.1.1”

    Mapping privateIP="10.1.1.1"    managementIP="18.1.1.1"

    Mapping privateIP="10.2-10.*.*" managementIP="18.2-10.*.*"
```

Refer to the

- Windows:`%OV_CONF%\dupip\dupip.conf`
- UNIX: $`OV_CONF/dupip/dupip.conf`

file for some examples and instructions on how to add information to the file.

dupip.conf Commands and Attributes

dupip.conf Commands

- **OverlappingAddressDomain id="number" name="string"**
  - One per file
- **Mapping privateIP="IP addr" managementIP="IP addr"**
  - Many per file
  - Can wildcard
- **Routable managementIP="IP addr"**
  - Directly addressable interfaces
  - Can wildcard
- **Gateway IP="IP addr"**
  - Edge devices (typically the NAT device)
  - No wildcards
- **NextHop IP="IP addr"**
  - Devices connected to the gateway
  - No wildcards

 ©2004 Hewlett-Packard Development Company, L.P.

The commands in `dupip.conf` are followed by attributes specific to that command and are of the syntax `attr="value"`.

`OverlappingAddressDomain` This command defines an OAD. Gateway, Routable, and Mapping commands which follow this are for this address domain. One and only one OverlappingAddressDomain command is required per configuration file. It must be the first command in the file.

`id`	This is the id of the OAD, and must be a unique positive integer which is not 0. (REQUIRED)
`name`	A user defined string which identifies this OAD. This string will be appended to the source column in the alarm browser. Because of this, we recommend using fairly short names (4-6 characters) so they can easily be read in the source column. At this time, only UTF-7 characters are allowed for this attribute. (REQUIRED)

Example: `OverlappingAddressDomain id="5" name="Bear"`

`Gateway` This command specifies gateways you want to manage for a particular OAD. Multiple Gateway commands can follow an OverlappingAddressDomain command. Wildcards are not allowed.

`IP` This is a gateway IP address for the OAD. It must be a Management IP address. (REQUIRED)

Example: `Gateway IP="15.20.155.8"`

`Routable` This command specifies a Management IP address which is routable. Multiple Routable commands can follow an OverlappingAddressDomain command. Wildcards are allowed in these mappings.

NOTE: When adding these Routable addresses, make sure NNM has not discovered them.

`managementIP` This is the IP address to use for communication to the interface/device. It must be a unique address, or range of addresses. (REQUIRED)

Example: `Routable managementIP="17.5.10-15.*"`

`Mapping` This command specifies the Private and Management IP mapping. Multiple Mapping commands can follow a OverlappingAddressDomain command. Wildcards are allowed in these mappings. The mapping will be invalid if the wildcards for the private and management IPs do not line up at each octet.

`privateIP` This is the real IP address for the interface/device (REQUIRED)

`managementIP` This is the IP address to use for communication to the interface/device. It must be a unique address, or range of addresses. (REQUIRED)

Example: `Mapping privateIP="10.1.5-10.*" managementIP="17.5.10-15.*"`

Create a Seedfile For Each OAD

- File name **dupip.seed**
- List of management IP addresses
  - One per line
  - Hostname optional. Must be resolvable at MS if used.
  - No wildcards
  - ONLY these IP addresses are discovered.

dupip.seed:

```
15.1.1.1   BigRouter.domain.com
15.2.5.8   FileServer.domain.com
16.1.1.1   SalesServer.domain.com
17.1.1.1   PresidentPC.domain.com
```


©2004 Hewlett-Packard Development Company, L.P.

3. You need to create a seed file for each OAD you want to manage. The seed file defines the discovery zone for the OAD, which appears in the `Overlapping Address Domains` tab of the Extended Topology configuration interface.

   Each seed file contains a list of the management IP addresses you wish to manage for a given OAD. Enter one management IP address per line. You may give its hostname for display in dynamic views, otherwise the IP address is shown.

   You may use # to comment out lines in this file.

```
$OV_CONF/nnmet/dupip/red/dupip.seed:
15.1.1.1  BigRouter.domain.com
15.2.5.8  FileServer.domain.com
16.1.1.1  SalesServer.domain.com
17.1.1.1  PresidentPC.domain.com
$OV_CONF/nnmet/dupip/blue/dupip.seed:
18.1.1.1 c4k32.dom.com
18.2.1.1
18.10.1.5  h375.dom.com
```

The hostname must be resolvable to the management address at the management station. When

you provide the optional hostname, NNM does a DNS lookup to see that the hostname given resolves to the IP address given. This is done regardless of what has been set up for `ipNoLookup.conf` and `snmpnolookupconf`.

Check Syntax Using ovdupip

- **ovdupip -ui a** verifies syntax of user-editable files

©2004 Hewlett-Packard Development Company, L.P.

4. Once you have your `dupip.conf` and `dupip.seed` files configured, you should run the `ovdupip` command to make sure your file entries are syntactically correct. If there are errors in the files, this tool tells you what is wrong, and where to look to remedy the problem. See the *ovdupip* reference page in NNM online help (or the UNIX manpage) for more information.

NOTE To make sure your file entries are syntactically correct, run the `ovdupip` command each time you modify one of your existing `dupip.conf` files or add a new `dupip.conf` file.

Any time a `dupip.conf` or `dupip.seed` file is changed, you should test it before applying those changes. To do that use `ovdupip -u` in conjunction with any of the other available options. The `-u` option tells `ovdupip` to run the command using the user-editable configuration files rather than the system runtime files. This program parses the file, and reports any errors.

To troubleshoot problems with NNM, use the `ovdupip` command (without the `-u` option) to review NNM's running configuration.

`ovdupip` has the following usage:

SYNOPSIS

```
ovdupip -[u]p managementIPaddress
ovdupip -[u]m privateIPaddress OADid
ovdupip -[u]e OADid
ovdupip -[u]h
ovdupip -[u]i OADid [infoFields]
ovdupip -[u]g OADid [gatewayFields]
ovdupip -[u]n OADid [nextHopFields]
ovdupip -[u]d OADid [mappingFields]
ovdupip -[u]s OADid [seedfileFields]
```

Where *OADid* refers to the id of the Overlapping Address Domain.

There are several fields for selecting the information you want to see. These fields specify which data are to be displayed, and in what order. Using the options shown below, put in the order you wish to have them output. By default all fields are displayed in the order listed below.

infoFields

- `i` means to display the `OAD id`
- `n` means to display the name for the `OAD id`

gatewayFields

- `i` means to display the `OAD id`
- `a` means to display the IP address of the gateway

nextHopFields

- `i` means to display the `OAD id`
- `a` means to display the IP address of the `nextHop`

mappingFields

- `i` means to display the `OAD id`
- `p` means to display the private IP address in the mapping
- `m` means to display the management IP address in the mapping

seedfileFields

- `i` means to display the `OAD id`
- `f` means to display the file name of the seedfile

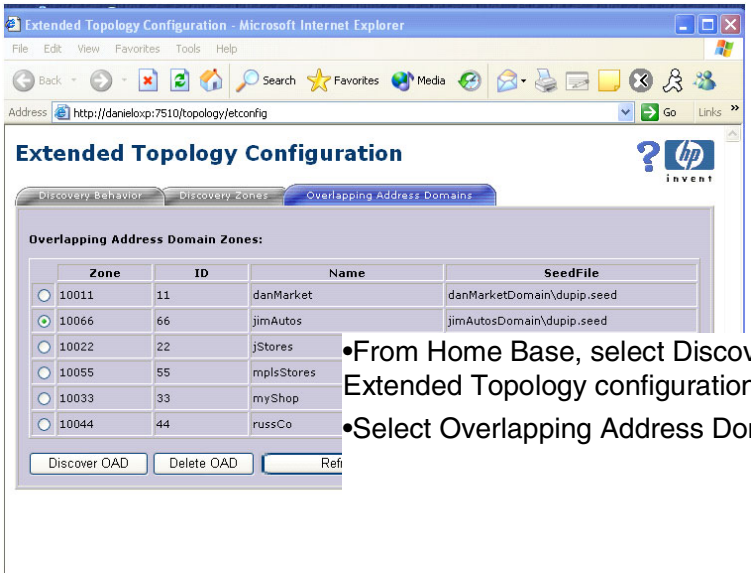
Here are a few examples of output from `ovdupip` using the sample `dupip.conf` files shown previously.

```
$ ovdupip -d a
5 15.1.1.1 15.1.1.1
5 10.1.1.1 16.1.1.1
5 10.2.1.1 17.1.1.1
5 15.2.*.* 15.2.*.*
```

```
10 10.1.1.1 18.1.1.1
10 10.2-10.*.* 18.2-10.*.*
15 10.*.*.* 30.*.*.*
15 20.5-10.1.* 31.55-60.8.*

$ ovdupip -i a
5 Red
10 Blue
15 Green

$ ovdupip -i 10 ni
Blue 10

$ ovdupip -p 18.5.45.20
10 10.5.45.20

$ ovdupip -m 10.1.5.5 15
30.1.5.5
```

Load Changes Into Extended Topology

Load Changes into Extended Topology



- From Home Base, select Discovery Status tab and Extended Topology configuration button.
- Select Overlapping Address Domains tab

©2004 Hewlett-Packard Development Company, L.P.

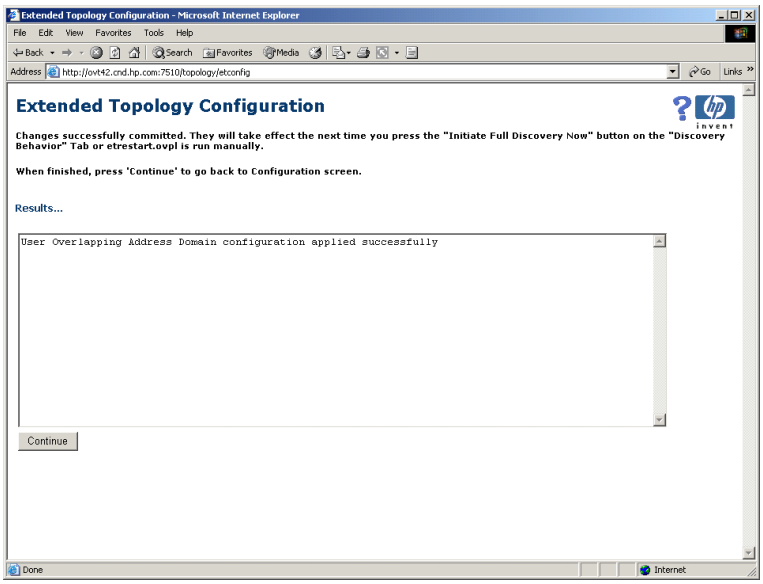
The configuration files in `$OV_CONF/nnmet/dupip` will not affect your running software. You must apply them in order for them to be copied into a system directory which is used by the OV software. This is done by going to the Extended Topology Configuration GUI via Home Base.

5. Go to the Extended Topology Configuration web page. You can get there from Home Base by selecting the Discovery Status tab, then selecting the Extended Topology Configuration command button.
6. Select the Overlapping Address Domains tab.

Deploy Changes to Running Configuration

Deploy Changes to Running Configuration

- All editing is done in user space and does not affect running configuration.
- Changes are copied to running configuration when you click [**Refresh Configuration and Activate Changes**].



 ©2004 Hewlett-Packard Development Company, L.P.

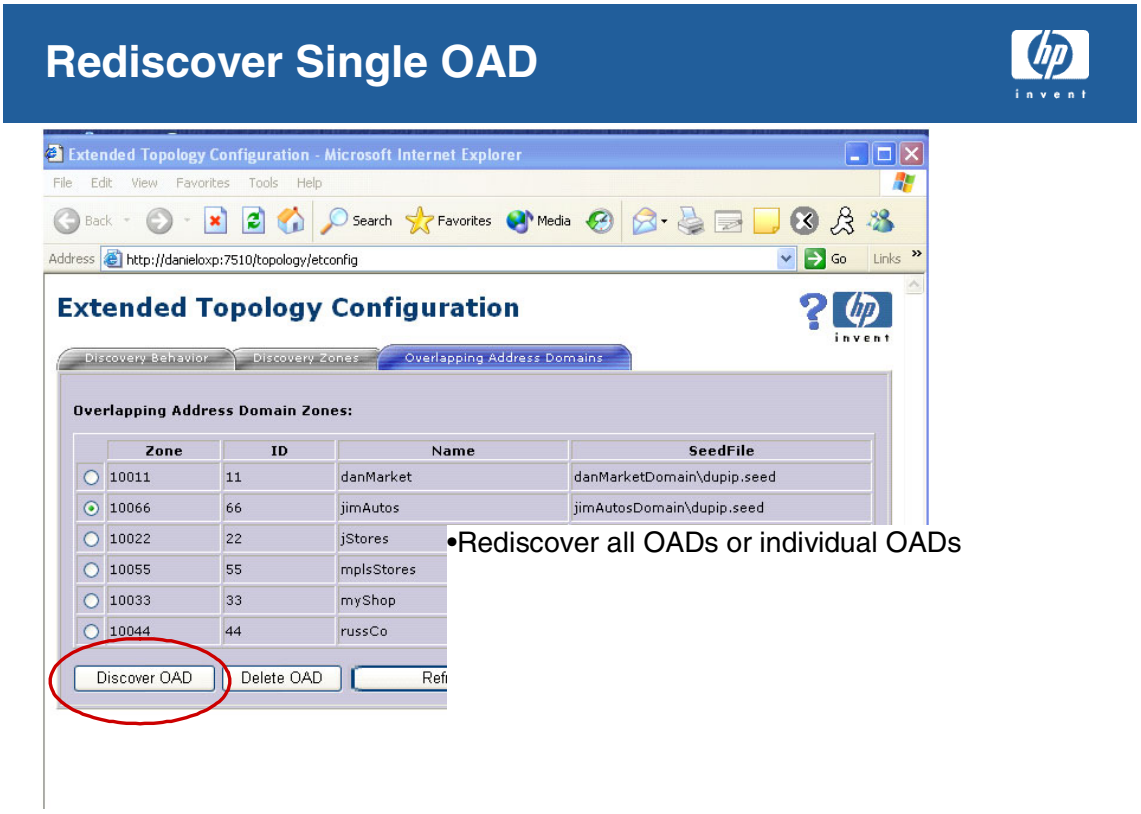
7. Select [Refresh Configuration and Activate Changes] to read, test, and deploy any changes or additions you made. If Extended Topology determines your changes are free of errors, it creates a zone for every defined OAD. These changes or additions will affect the next discovery cycle.
8. To immediately initiate a complete discovery of all zones, execute, as Administrator or root, the etrestart.ovpl command. You can also select [Initiate Full Discovery Now] from the Discovery Behavior tab located in the Extended Topology Configuration menu.

   If there are any errors in the configuration files, they are reported in a pop-up message, and the changes are NOT applied.

When a successful [Apply] operation happens, all of $OV_CONF/nnmet/dupip is copied to $OV_CONF/nnmet/.dupip on UNIX or %OV_CONF%\nnmet\dupip on Windows, which is where the runtime code actually reads configurations from. The files in this directory SHOULD NOT be edited by users; you must edit the files in $OV_CONF/dupip (%OV_CONF\dupip) instead, and then [Apply] the changes.

When this is successfully executed, the processes re-synchronize on their own every 30 seconds. From this point on, the processes operate with the new configuration. Processes do not have to be stopped and restarted.

Rediscover a Single Zone



9. If you add *new* devices to or delete devices from a single OAD (zone), you can save time by initiating an Extended Topology discovery on that single zone.To initiate a discovery cycle for a specific zone, use the following procedure:

   a. From `Home Base`, select the `Discovery Status` tab.

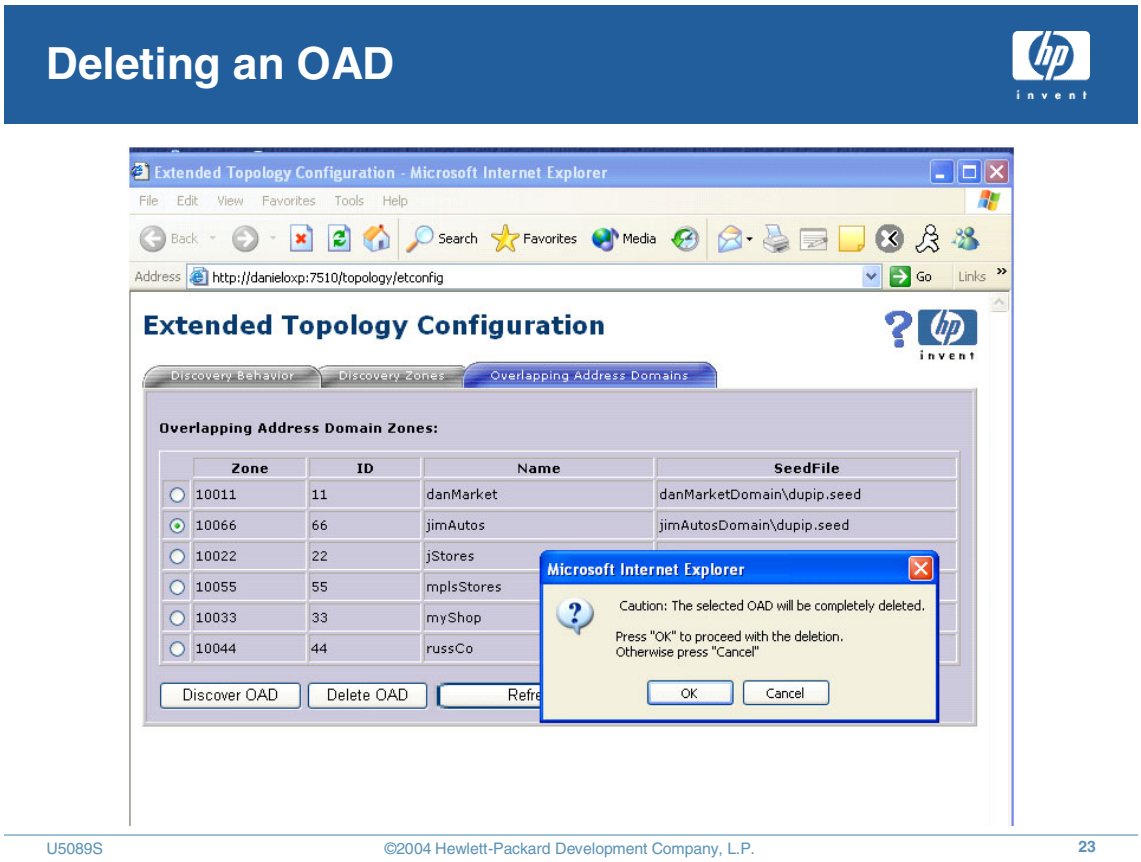
   b. Select [`Extended Topology Configuration`].

   c. Select the `Overlapping Address Domains` tab.

   d. Select the option button to the left of the zone you want to discover.

   e. Select [`Discover Zone`].

Deleting an OAD



If you are no longer responsible for an overlapping address domain, you may remove it from your configuration to eliminate unnecessary polling. From the Overlapping Address Domains Tab, select the domain and click [`Delete OAD`].

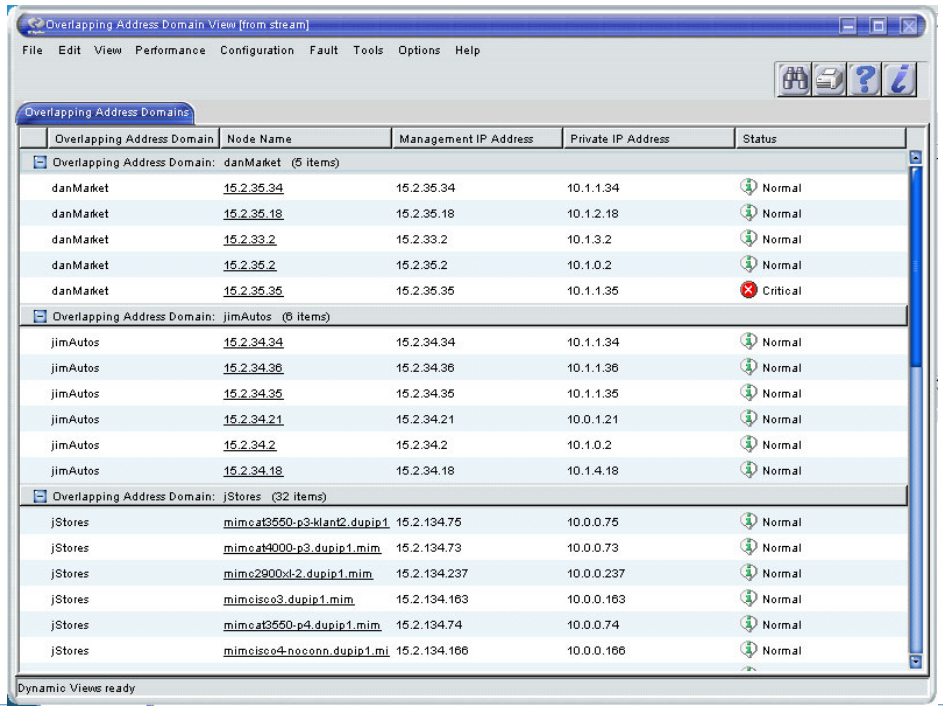
The interface verifies that you want to delete the OAD and removes it from the configuration.

Each time the configuration changes, whether by a deletion or by [`Refresh Configuration and Activate Changes`], the system saves a backup of the current configuration under the directory `$OV_CONF/nnmet/.dupip.bak`. You may access this directory to recover your configuration as long as you do so before another configuration change is made. Only one level of backup is retained.

The status on the Discovery Status tab is updated to reflect that an OAD has been deleted.

OAD View

Overlapping Address Domain View



 ©2004 Hewlett-Packard Development Company, L.P.

NNM has an Overlapping Address Domain (OAD) View that shows the OADs in the network. This is a tabular view that contains the OAD Name, OAD Id and Number of nodes in each OAD.

You can get to a specific OAD from this top level view. This view contains all the nodes in each OAD with their status.

The view to support overlapping domains are Neighbor View, Path View, VLAN View, and HSRP View. All other views will not be able to navigate to OAD nodes.

The mouseover shows the management IP address, private IP address, OAD textual name, and OAD Number, if available. The view has a menu item to view/find by management IP address.

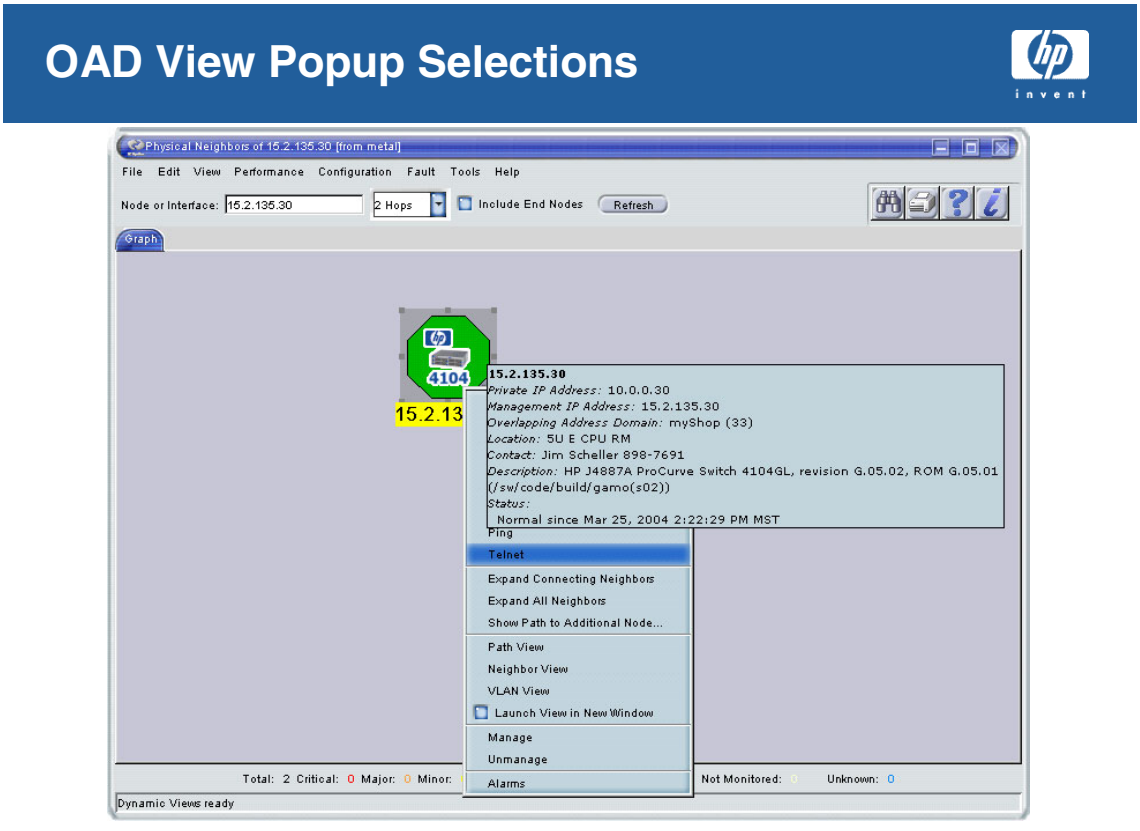
The following view capabilities are available for OADs:

- Neighbor View from Alarm
- Neighbor View enter Public Address
- Mouseover of Extended Topology-only attributes
- Status of Extended Topology-only OAD nodes from topology
- Dynamic Status update of Extended Topology-only OAD nodes
- Launch menu items (telnet) with public address
- Extended Topology-only node details by UUID (OAD node details)

At this time you are unable to:

- View private/management address in labels
- Enter private address (and OAD) into Neighbor View
- Filter for OAD in Node View (filter based on OAD)
- Launch an Internet view showing OADs
- Path view for private IP addresses
- OADs in OSPF views

OAD View Popup Selections

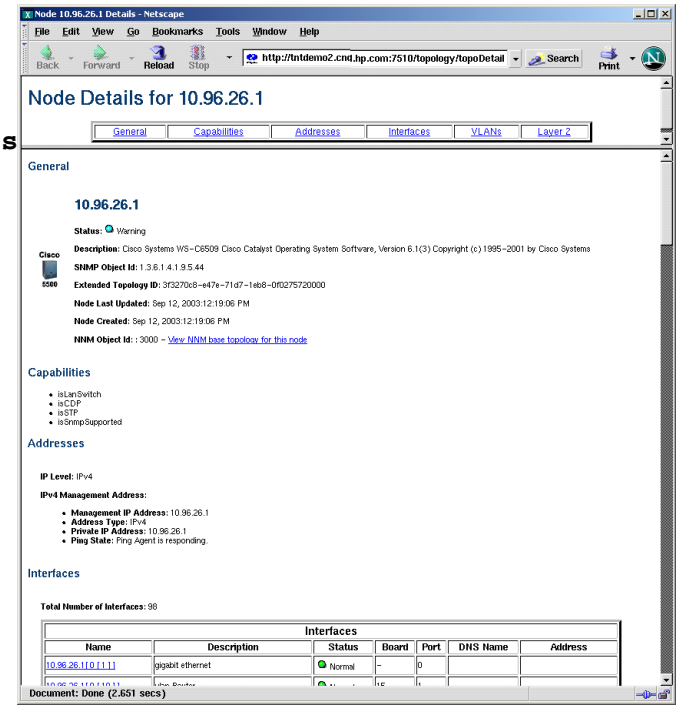


From the OAD view, you can look at the details for a specific node, poll it, telnet or traceroute to it, or start Neighbor View.

Viewing OAD Node Details

Viewing Node Details

- In the Alarm Browser, select **`Actions:Views:Source Details`**.
- IN OAD View, select **`Details`** on the popup menu.
- OAD IP nodes don't have ovwId's – can NOT use **`ovtopodump.ovpl`** to get attributes

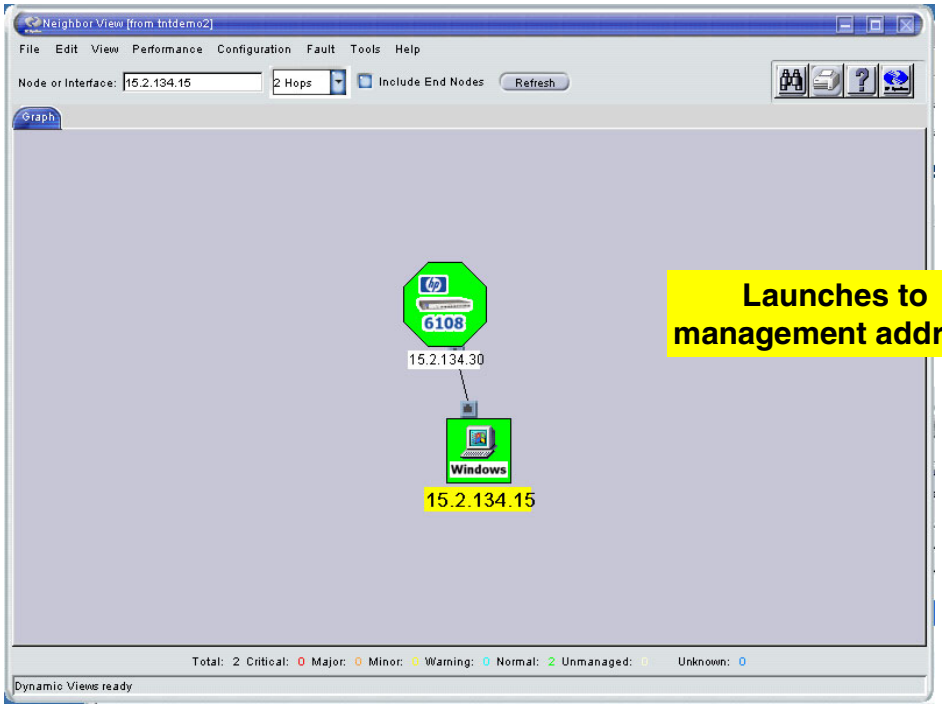


©2004 Hewlett-Packard Development Company, L.P.

Node Details (`ovtopodump.ovpl`) will not be available for OAD nodes, because they have no `ovwId` to get `ovtopmd` information.

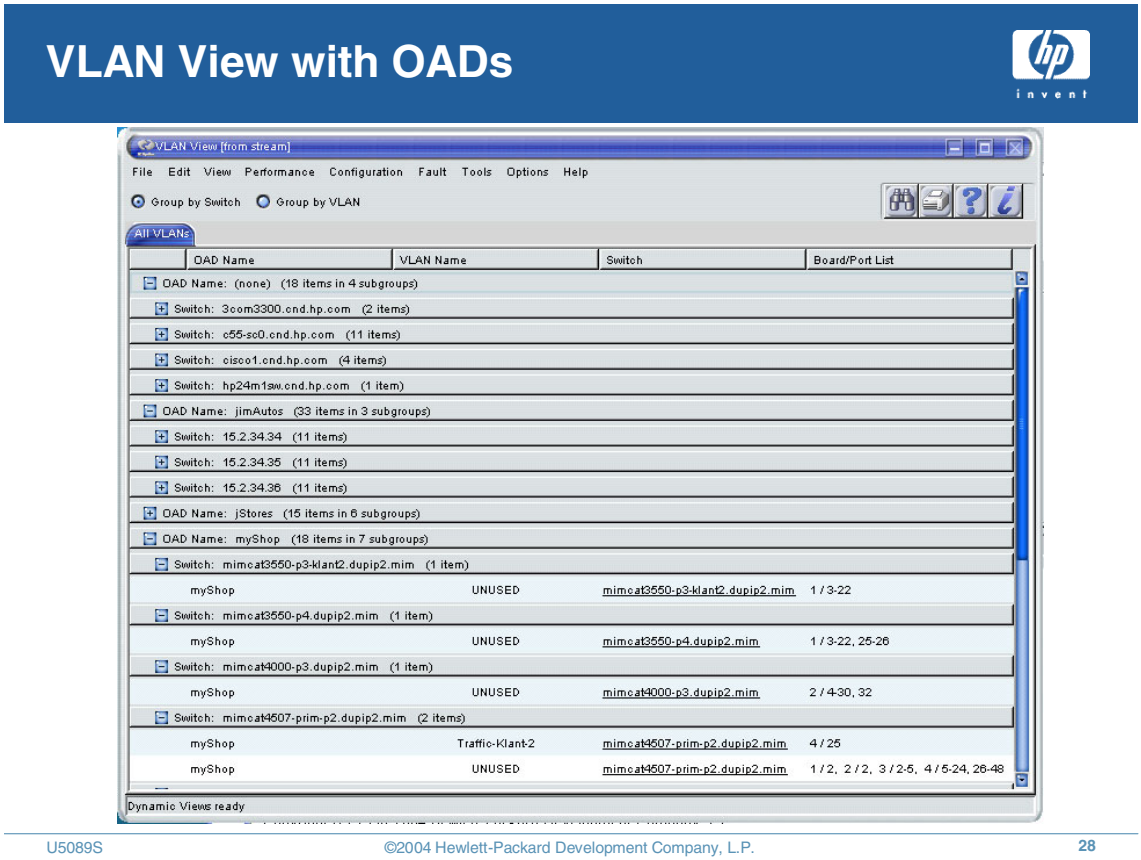
Neighbor View in an OAD Environment

OAD Neighbor View



©2004 Hewlett-Packard Development Company, L.P.

VLAN View with OADs



Extended Topology can discover and monitor VLANs that are contained within a single OAD domain.

When viewing the VLAN and HSRP data, the operator needs to be able to see consistent data for the VLANs and HSRP groups organized into the separate Overlapping Address Domains.

For example, if an ISP operator needs to monitor status of VLAN nodes in their OAD environment, the operator expects to be able to manage each customer (domain) environment independently. The operator uses the VLAN view to see the information grouped by domain, such that it is easy to identify issues isolated to a particular customer.

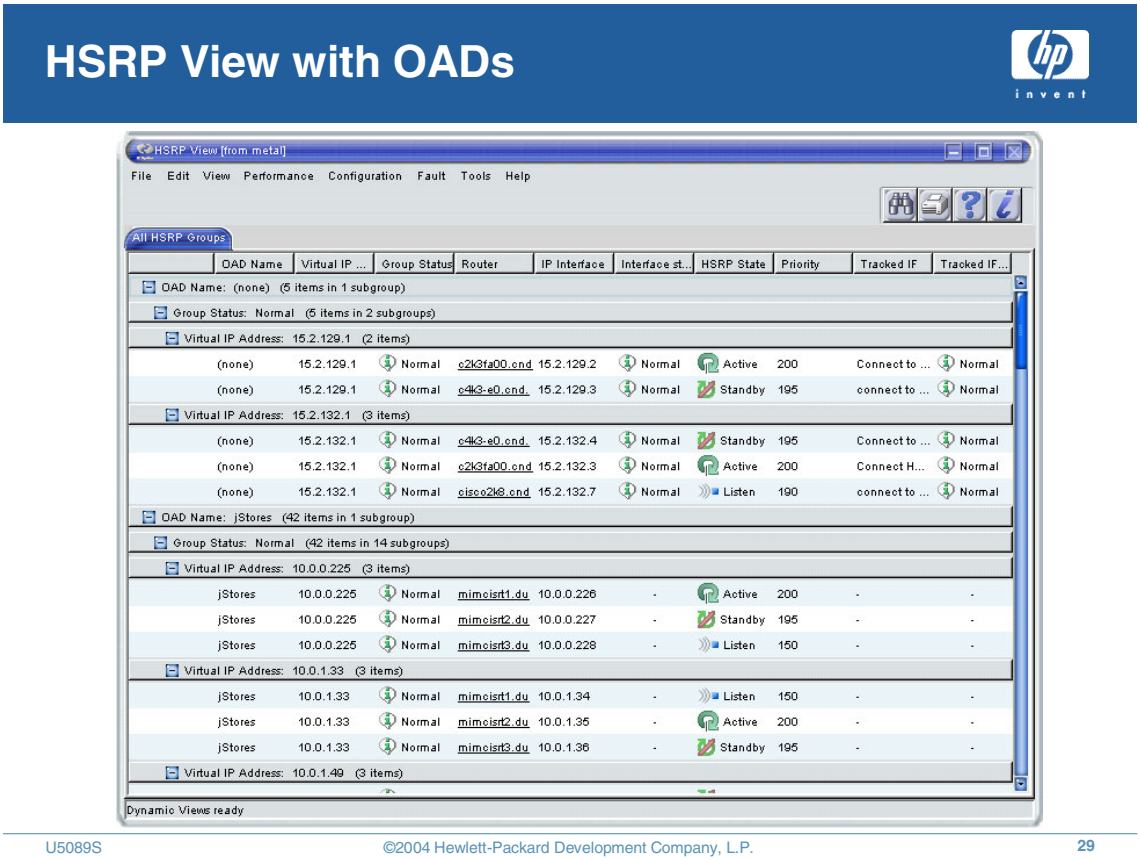
These views show the OAD Name as a column in the table when there is one or more domain present.

The data is grouped primarily by OAD name, and secondarily by whichever selection is made by the user in the radio button (i.e. “Group by Switch” or “Group by VLAN”).

In the case of a single domain, the “OAD Name” column is present, but no OAD grouping name occurs. In the case of a non-OAD environment, the column is absent.

Management addresses of members of a VLAN must be unique across all OADs.

HSRP View with OADs



Extended Topology can discover and monitor HSRP groups that are contained within a single OAD domain.

The HSRP view includes the OAD name when there is more than one domain present. The data is grouped primarily by OAD name, secondarily by Group Status, and third by Virtual IP address.

In the case of a single domain, the “OAD Name” column is shown, but no OAD grouping occurs.

The virtual IP address associated with the nodes participating in the HSRP group must have a management address, which is no different from any other managed node in OAD.

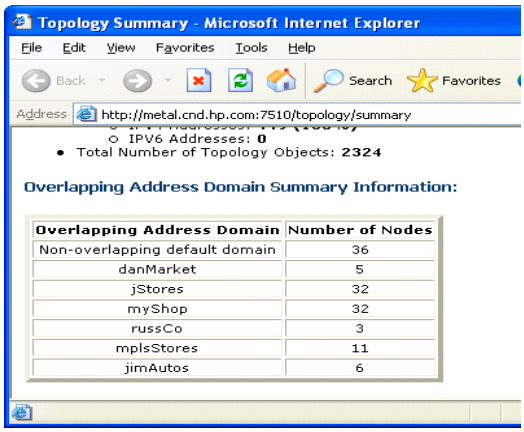
One or more overlapping address domains can have same HSRP virtual IP addresses on their devices. These virtual IP addresses are mapped in the NAT address translation configuration (dupIP configuration).

The private IP addresses on the interfaces that support HSRP within an OAD are unique to that domain.

OAD in Topology Summary

Overlapping IP Addresses in Topology Summary

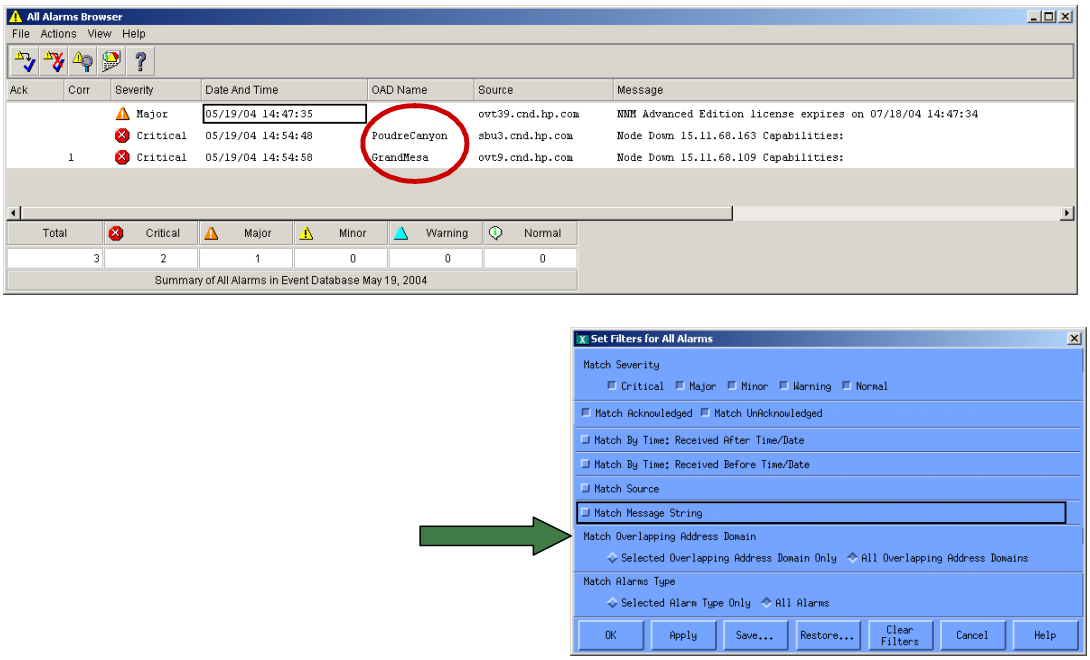
– Can see the Overlapping Address Domain



©2004 Hewlett-Packard Development Company, L.P.

OADs in the Alarm Browser

Filtering the Alarm Browser on OAD



 ©2004 Hewlett-Packard Development Company, L.P.

The alarm browsers show the Overlapping Address Domain (OAD) name in a separate column. (The column is hidden if there are no OADs.) You can filter on the OAD.

In the native Alarm Browser, select an alarm in the Alarm Browser and when you select `View:Set Filters`, you see a radio button `Selected Overlapping Address Domain Name Only`. (If no OADs are configured, the button is unavailable.)

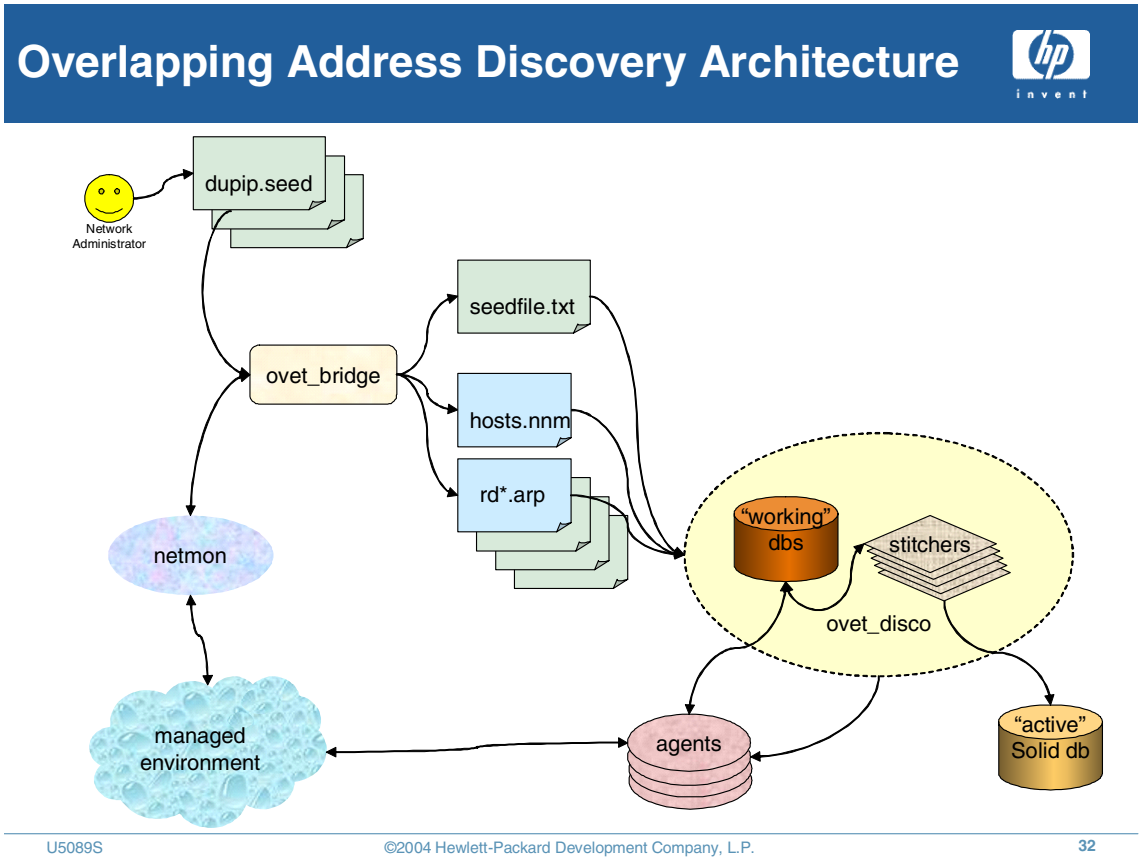
In the web Alarm Browser, select the by `OAD Name` tab. From the pull down list of OAD names, [`Add`] OADs to the filter. You can select one or more OAD names from this list and [`Add`] them to the selected OAD box.

In the Additional Actions window, you can “Sort Alarms by OAD name” for all scopes of actions.

Events carry an added varbind for the OAD. If it is not present, NNM assumes the OAD ID of zero (0), which indicates that the device is not part of an overlapping domain.

Some event varbinds that contain the source IP may return the private address. Investigate each even to decide whether to use $2 or $R to get the management address of the source IP.

OAD Discovery Architecture



Overlapping Address Domain (OAD) is used to enable single management station to manage multiple overlapping private address domains.

`ovet_disco` sends the private IP address and OAD to the Extended Topology database for both Nodes and Interfaces. Extended Topology database looks up the private IP Address and OAD and returns the management IP address. The private IP address, OAD and management IP Address are stored in the Extended Topology database.

There are private addresses where the public address is not known; in this case the public address will be populated as null in the database. This may occur if discovery encounters additional interfaces on devices, that are not in the NAT.

In a non-overlapping environment, the private address is the same as the public address and the OAD is 0.

When an SNMP trap is received, `ovtrapd` looks to see if the source address has a mapping to an OAD. If so, `ovtrapd` adds varbinds to the event for the OAD ID and the management IP address.

Alarm Browsers look for the OAD ID varbind and add "@OADname" to the source just before display.

Troubleshooting Overlapping Address Domains

Troubleshooting

- To verify healthy operation
  - Check OADs listed in Extended Topology Configuration GUI
  - Check Topology Summary for number of devices in OADs
  - Check OAD view for list of OADs and devices
  - Check Neighbor View for OAD devices

©2004 Hewlett-Packard Development Company, L.P.

Troubleshooting Overlapping Address Domains

Troubleshooting

- $OV_CONF/nnmet/dupip
  - Verify OADs exist as you expect
- $OV_CONF/nnmet/.dupip
  - After a successful apply, should be exactly what is in dupip
- $OV_BIN/ovdupip –ui a
  - Fix any errors and activate changes
- $OV_BIN/ovdupip -i a
  - System config should never have an error
- $OV_CONF/nnmet/etconfig.xml
  - Should have one zone for each OAD defined. This file is automatically generated when a successful “Refresh Configuration and Activate Changes” operation occurs. If this file doesn’t have all of your OADs, you have not activated the changes successfully.
- $OV_DB/nnmet/seedfile.txt
  - Automatically created from dupip.seed files before discovery starts. Shows the zone number and the OAD id for the IP as well.
- $OV_BIN/ovet_topodump.ovpl -nodeIf OADid
  - Show node and interface information for the OAD
- $OV_BIN/snmpwalk
  - Verify snmp access to problem device

 ©2004 Hewlett-Packard Development Company, L.P.

The system configuration (`.dupip`) should never have an error because it is only created from a valid user configuration.

`ovdupip` can give other information about the configuration as well. See it’s man page, or its usage.

`etconfig.xml` should not be edited by hand. It is created during an apply from the Extended Topology Configuration GUI. If it does not exist, or does not display an OAD you think should be there, you have not successfully activated the changes. Run ‘`ovdupip –ui a`’ to make sure there are no errors, and that your OAD is listed. If all looks fine, then go back to the Extended Topology Configuration GUI and click [`Refresh Configuration and Activate Changes`]. Your `etconfig.xml` file should have your OAD listed.

`seedfile.txt` is also automatically generated and you should not edit it by hand. Look at it to see that your OADs have the IP addresses you expected. If not, you’ll have to update the `dupip.seed` file and activate the changes.

Lab Exercises

 ©2004 Hewlett-Packard Development Company, L.P.

OAD Lab A: Overlapping Address Domain Discovery

Objectives:

- Become familiar with setup and discovery of Overlapping Address Domains.

At the completion of this module you will be able to:

- Configure overlapping address domains.
- View overlapping address domains.

CAUTION Successful execution of this lab depends on following the directions given by your instructor AT THE TIME those instructions are issued.

Assumptions

- Student has previously completed the Lab System Setup Procedures.
- Lab directories for `Lab_oad` and `Lab_deploy` are available.

Directions

1. Change working directory to `$OV_CONTRIB/NNM7labs/Lab_oad`:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_oad
   ```

2. When the instructor informs you to proceed, execute the setup script:

   ```
   setupLab_oad.ovpl
   ```

3. In a command window, change to the `bin` directory:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_oad/bin
   ```

   Execute the copy script `xferConfigFiles.ovpl`. This will copy files to the `$OV_CONF/nnmet/dupip` directory.

4. Using an editor, examine the `dupip.conf` and `dupip.seed` files in `$OV_CONF/nnmet/dupip/leftnet` and `$OV_CONF/nnmet/dupip/rightnet`. These correspond to the left and right halves of the diamond vlan configuration. Each of these two directories should have a `dupip.seed` and a `dupip.conf` file.

   a. `dupip.seed` files: note these look like `host` files with the management version of the addresses (compare leftnet and rightnet versions).

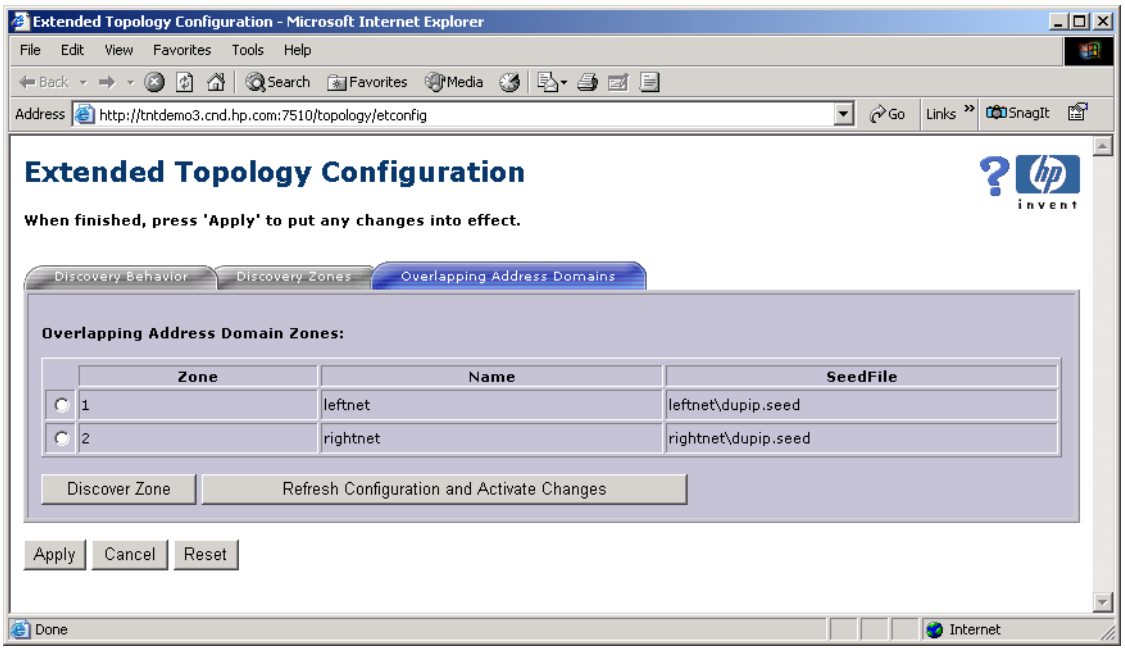
   b. `dupip.conf` files: note these map a private IP space (the 192.168.168 network, which is the same for both the leftnet and rightnet versions) to a management IP space (this is different, it is 10.96.26.193-205 in the leftnet case, and 10.96.26.225-237 in the rightnet case).

5. From the Home Base, select the Discovery Status tab and click [`Extended Topology Configuration`]. After supplying the passwords, you should see the Extended Topology Configuration window. Select the Overlapping Address Domains tab in this window.

6. Click [`Refresh Configuration and Activate Changes`].

   a. You should see a popup window that says “User Overlapping Address Domain Configuration applied successfully.” Click [`Continue`].

b. You should see two zones now configured, for leftnet and rightnet.



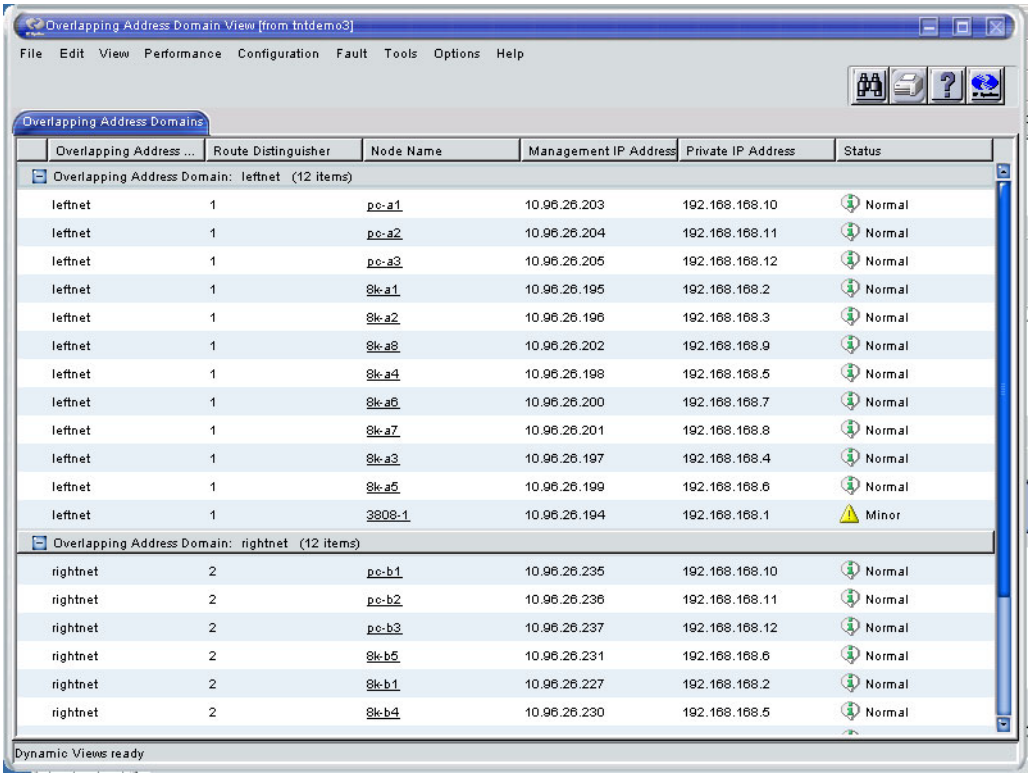
7. Now do a full Extended Topology discovery:

   a. Select the Discovery Behavior Tab, and click [`Initiate Full Discovery Now`].

   b. Allow discovery to complete. Throughout the discovery, you should see it discover 9 nodes in the first zone, 12 nodes in the second zone, and 12 nodes in the third zone.

8. From Home Base, launch the Overlapping Address Domain view. Expand the groups. You

should see two groups of 12 nodes each:



Note the following:

a. The groups are named leftnet and rightnet.

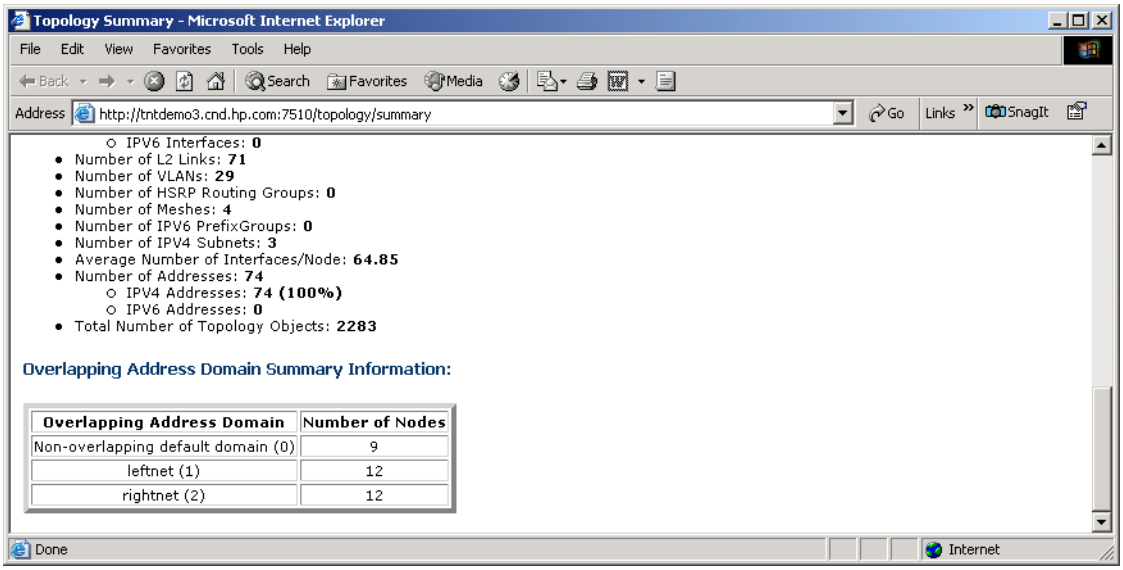
b. The private IP addresses are the same but the management IP addresses are different.

9. Note from the OAD view, you can click a node name to examine node details.
10. Launch a Neighbor View of 8k-a2 from the OAD view by selecting a node, then using the right-mouse button popup menu.
11. Examine the VLAN View in the OAD environment.
12. Examine the OAD information in the Topology Summary:

    a. From Home Base, select the Discovery Status tab.

    b. Click [`View Topology Summary`].

c. Scroll to the bottom.



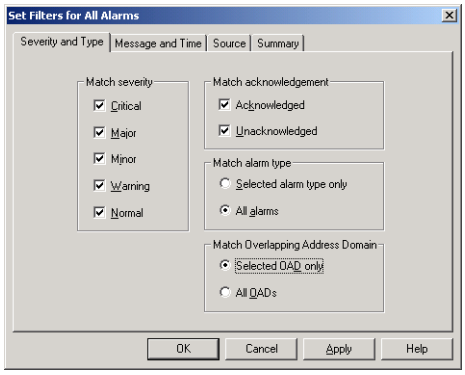
13. Examine filtering capabilities in the Alarm Browser.

    a. In a command window, type `$OV_BIN/xnmevents`. This launches the native Alarm Browser.

    b. Select `View:Set Filters`.

    c. Select any OAD alarm in the browser.

    d. Choose `Selected Overlapping Address Domain Only`, and [`Apply`]. The browser only shows alarms associated with the OAD selected.



    e. From the web Alarm Browser started from Home Base, select `View:Set Filters`.

    f. Select the by OAD Name tab.

    g. Select an OAD name from the list and click [`Add`].

    h. Click [`Apply`].

14. Examine the configuration information used by NNM in `$OV_DB/nnmet/hosts.nnm` and `$OV_DB/nnmet/rd*.arp`.

15. Clean up the configuration by executing `$OV_CONTRIB/NNM7labs/Lab_oad/bin/deleteConfigFiles.ovpl`.

16. Open the Extended Topology Configuration interface from Home Base and click [`Refresh`

Configuration and Activate Changes].

8 Active Problem Analyzer

Module Objectives

Active Problem Analyzer (APA): Advanced Network Fault Monitoring, Analysis and Diagnosis

Version C.00
U5089S Module 8 Slides

At the completion of this module the student will be able to:

- Describe the objectives and operation of the Active Problem Analyzer.
- List the environments in which APA provides status for devices.
- Compare and contrast APA functionality with netmon.
- Configure APA operation.
- Describe event flows through APA.
- Describe how neighbor and down stream analysis is handled.
- Initiate an APA demand poll and describe its actions.
- List events which cause APA to take action and what those actions are.
- Identify devices which are not monitored by APA.
- Describe APA's handling of unconnected ports.

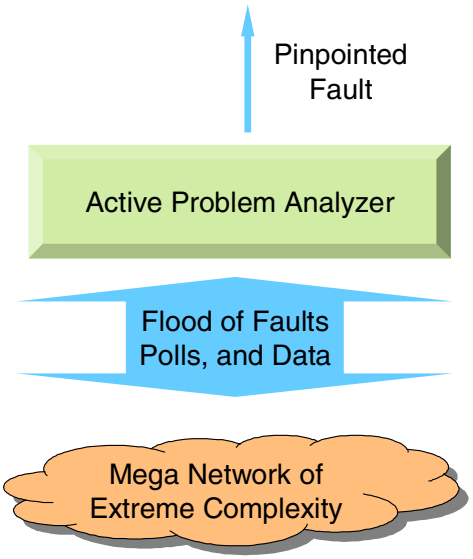
- Describe the sequence of processing when interfaces are renumbered on a device.
- Describe APA's configuration polling and when it occurs.
- Configure APA handling for important nodes as secondary failures.
- Interpret the APA statistics display to verify health of the software.

Active Problem Analyzer

Active Problem Analyzer (APA)

Primary Goals:

- High Level of Network Polling Scale (not yet quantified)
- More Accurate Network Fault Determination, particularly in redundant environments.
- Easily deployed, configured, and customized to meet the customers needs.
- Extensible Architecture for the future of NNM Extended Topology.



U5089S ©2004 Hewlett-Packard Development Company, L.P. 2

This new component, the Advanced Problem Analyzer, `ovet_poll`, goes beyond the status polling of `netmon`, and the analysis possible in ECS. The poller addresses these key requirements:

- New level of status polling scalability (compared to `netmon`), reducing the customers' Total Cost of Ownership (TCO) by reducing the number of systems required for monitoring an existing complex network.
- More accurate fault determination (compared to `netmon`), particularly in highly redundant and multiple VLAN environments, where multiple paths and transient network conditions make analysis by the existing NNM code incomplete, thereby reducing the Mean Time To Repair (MTTR).
- Easily deployed, configured, and customized to meet the customer needs, focusing on Quick Time To Value (QTTV). ***For 7.0/7.01, the focus is on out-of-the-box configuration. Future releases will address this further.***
- Create an extensible architecture for the future of NNM/Extended Topology polling and status monitoring.

As this is the first release of this component, these primary requirements are met more or less depending on project trade-offs. However, it is important that at the end, the focus is on creating the most accurate fault determination (root cause) based on the available topology and polling information, as this is ultimately the value delivered to the customer.

Timely Network Fault Detection

The primary purpose of the poller/analyzer is to detect and accurately report the root cause of network faults. To do this, the poller performs periodic queries and tests of the network state via a variety of protocols and methods. The output of this detection is an event notification of the detected fault, potentially with other faults identified as secondary to this fault.

Network Neighbor Analysis

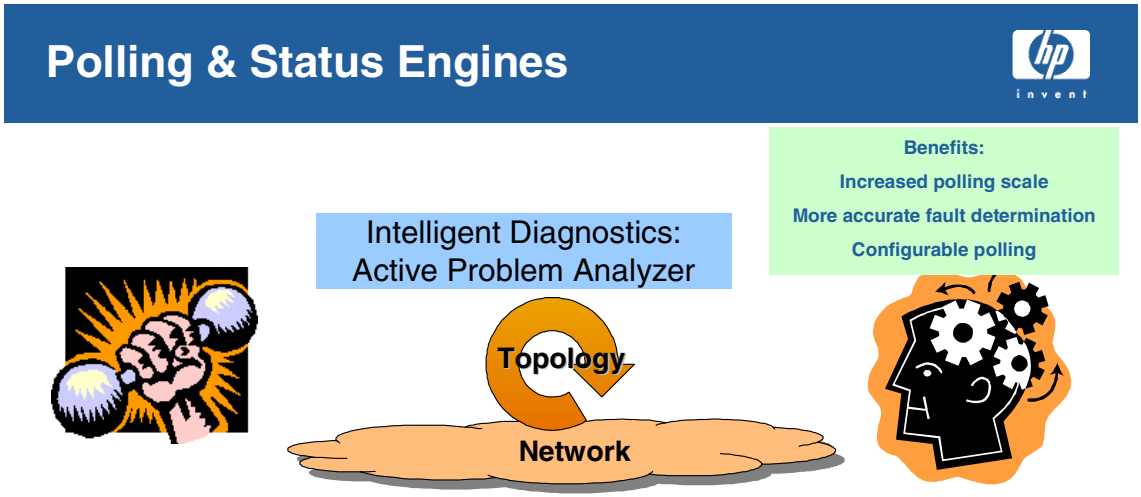
Accurate identification of the root cause of faults that affect a broad range of network elements is critical, with the correct handling of secondary effects a concern as well. For example, the correct identification of impacted devices. This implies a need for accurately determining connectivity and between particular elements. The APA takes a new approach in this area focused on neighbor analysis algorithms.

Network Status/State Maintenance

In addition to generating network fault notifications, the poller is responsible for maintaining the state of topology objects it is monitoring. This includes both the setting of status to up/down in the topology, as well as state fields for other monitored attributes (e.g. the HSRP state of an interface). Status and state values are maintained for three purposes:

- Status and attributes needed for dynamic views
- Use by other applications that may want to act on the changes in state
- As needed by the poller itself to make an accurate fault determination

NNM AE Polling and Status Engines



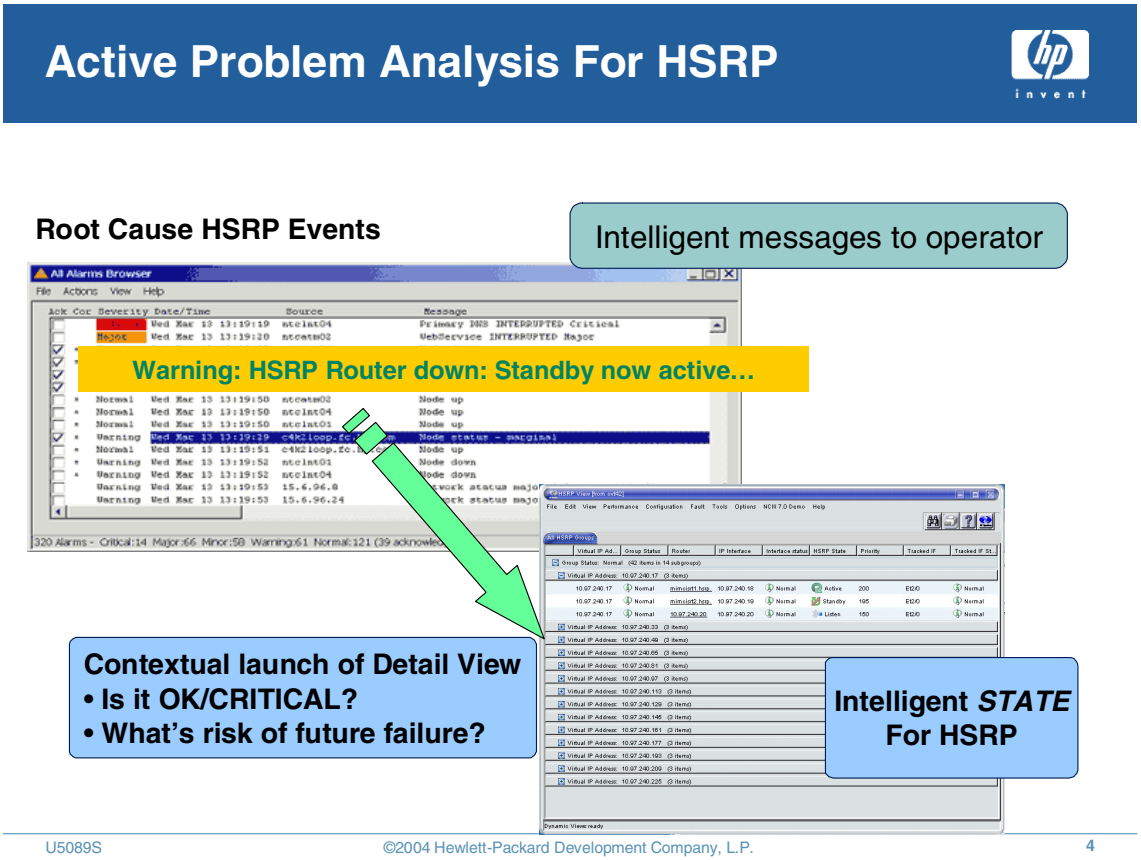
The Active Problem Analyzer is comprised of an intelligent poller and a status analyzer engine to increase scalability as well as allow you to understand the status or "state" of what is happening. It understands how all the protocols operate to increase the out of the box intelligence provided.

Active Problem Analyzer will be able to take the output of event-based root cause analysis, collect additional information through targeted polling and data collection (for example via device syslogs), apply built-in logic specific to the problem scenario, and determine the state of network facilities involved in failures. Such information is critical to understanding whether intervention is required. In the current product, the analyzer works from polling information gathered by APA during its regular polling cycle.

APA analyzes the connectivity details from the extended topology database to provide you with accurate HSRP and OAD view status information. It provides you with this information in the following ways:

- It analyzes information from neighboring interfaces in order to provide better diagnostic information.
- It generates alarms indicating the root cause of an HSRP or OAD problem. You will spend less time searching for the root cause of a failure.
- It maintains node, interface, and address status attributes.
- It modifies Dynamic View node status colors using node status and other node attributes.

Active Problem Analysis for HSRP



One of the first problem scenarios available in Active Problem Analyzer is Cisco Hot Standby Router Protocol (HSRP). NNM correlation services intercept raw events from HSRP router groups and correlate them into higher level, more meaningful events. Those events, with their enriched data, are then passed to Active Problem Analyzer for further processing.

Active Problem Analyzer automates much of the additional investigation that network operations or engineering would have to undertake to determine the health of the HSRP group and decide if intervention is required. Here is what happens:

1. The APA takes over after the HSRP events are correlated, producing a summary HSRP event from the multiple raw HSRP events. The APA looks at this event and identifies the HSRP group which needs to be analyzed further. It then retrieves the layer 2 HSRP topology for just that group from the NNM Extended Topology database.
2. Active Problem Analyzer then identifies the specific routers and interfaces required for HSRP redundancy to operate successfully. These interfaces are polled to determine their exact state. From this, the state of the HSRP group state is determined and an intelligent event is displayed in the Alarm Browser.
3. APA checks to see if the router that failed was actually replaced by one of the standby HSRP routers and if the failed router and its interfaces have recovered and are available for the next failure to occur. You can predict whether the HSRP function is able to withstand additional failures.

4. The HSRP group and its state are available in tabular form for the operator from a single mouse click from the HSRP event. If further investigation is required, you can easily navigate to adjacent network views—the layer-2 with VLANs switch fabric connecting the two routers, for example.

This automatic analysis and representation provide far more information that a simple event and is necessary to correctly identify the root cause of the failure. Network managers not only receive a notification that a fault has occurred and what the source is, they can see the current state of the HSRP group.

You must have a valid license for the Advanced Routing Smart Plug-in (SPI) for NNM to use the HSRP functionality.

NOTE APA discovers HSRP information and monitors HSRP status in a general IP environment. It does not discover HSRP information or monitor HSRP status in an OAD environment.

Status Polling Engine

Polling engine

- Highly scalable, using a high performance poller that actively looks for potential problems.
- Multi-threaded for optimal performance
- Capable of performing ICMP or SNMP polls
- Intelligent in using network bandwidth
  - Packs many SNMP requests into one
  - Queries state of multiple interfaces in a single SNMP query.
  - Only polls active/connected interfaces
- Extended Topology-based

©2004 Hewlett-Packard Development Company, L.P.

`netmon` and APA monitor separate areas. APA can take over completely for `netmon` status polling when configured to do so. This includes monitoring device status via ICMP and SNMP, as well as adding other types of monitoring, actively polling for additional information to determine a more accurate root cause, and based on the Extended Topology database.

Some key points about the Poller/Analyzer as it relates to Extended Topology:

- There are two key outputs from the poller:
  - A stream of root-cause events pinpointing network failures are sent to the NNM classic event system.
  - Object status is updated in the Extended Topology database as a result of the polling. Initially these updates are in the form of setting status values for the monitored topology objects.
- The Poller relies heavily on the Extended Topology database for accurate analysis. The accuracy of analysis depends largely on the accuracy of the topology in the Extended Topology database.
- An XML based configuration approach provides the basis of the monitoring of the environment. This file defines the polling behavior of the system based on a class/filter system provided by the Extended Topology database. Specific instances of changes are supported.
- The status determination from the Extended Topology is reflected back to the NNM topology

for consumption by `ipmap` and other NNM based applications.

- While the poller focuses on replacing the IP status polling of `netmon`, the intent is to provide additional forms of status monitoring for more network fault management. For example, the monitoring of HSRP groups.

Does APA use any of the layer 2 information from the netmon process?

Answer: APA uses the layer 2 connectivity information stored in the extended topology database. It is important to remember that APA still relies on the `netmon` process for device discovery.

How do you configure SNMP information for APA?

Answer: APA uses the same SNMP configuration information as the `netmon` process.

Can I use APA to select and demand poll a device?

Answer: With APA, you cannot select a device and execute a demand poll on the device.

Can APA ping the virtual IP address?

Answer: This is not supported in this release.

Analysis Engine

Analysis engine

- Provides accurate root cause of network faults by combining poll results with topological state of objects
  - Tight integration with topology model, including active state of topological objects like HSRP group.
- Takes into account changes in relationships between objects, without needing to “recompile” in order to determine root cause.
  - Example: change in the HSRP “master” for a redundant router group
- Analysis can trigger new polls to validate or to provide additional information to get to the root cause. Trigger could be based on:
  - The result of a previously requested poll
  - Changes to attribute(s) of topological objects, such as object reachability going to down, unreachable or indeterminate

U5089S ©2004 Hewlett-Packard Development Company, L.P. 6

The poller is composed of two primary blocks, the PE (Polling Engine) and the SA (Status Analyzer). The job of the PE is to provide polling scalability beyond what `netmon` delivers. Large use of threads in the design is a big part of this. If a poll of an address, node or interface fails, the PE delivers this failure to the SA which does the following:

1. Analyzes the situation to determine the root cause (not to be confused with root cause analysis done through ECS).
2. Update the status in Extended Topology database of primary failure (root cause) entities and some number of secondary failure (impacted) entities.
3. Emit events for the primary failure entities. Some of these events are new and have no counterpart in NNM. Others are new versions of similar NNM events, but referencing Extended Topology database objects.
4. Emit events for some number of secondary failure entities and do so in a manner that works with the ECS ConnectorDown circuit such that drill down operates as expected.
5. Provide the required integration with NNM such that some amount of status is shown in `ovw` and some number of expected NNM events are generated.

Does APA consider Spanning Tree Protocol (STP) when analyzing failures?

Answer: APA considers the effects of the STP when diagnosing the root cause of unresponsive interfaces. It suppresses transient failures due to spanning tree reconvergence which results in

fewer transient alarms.

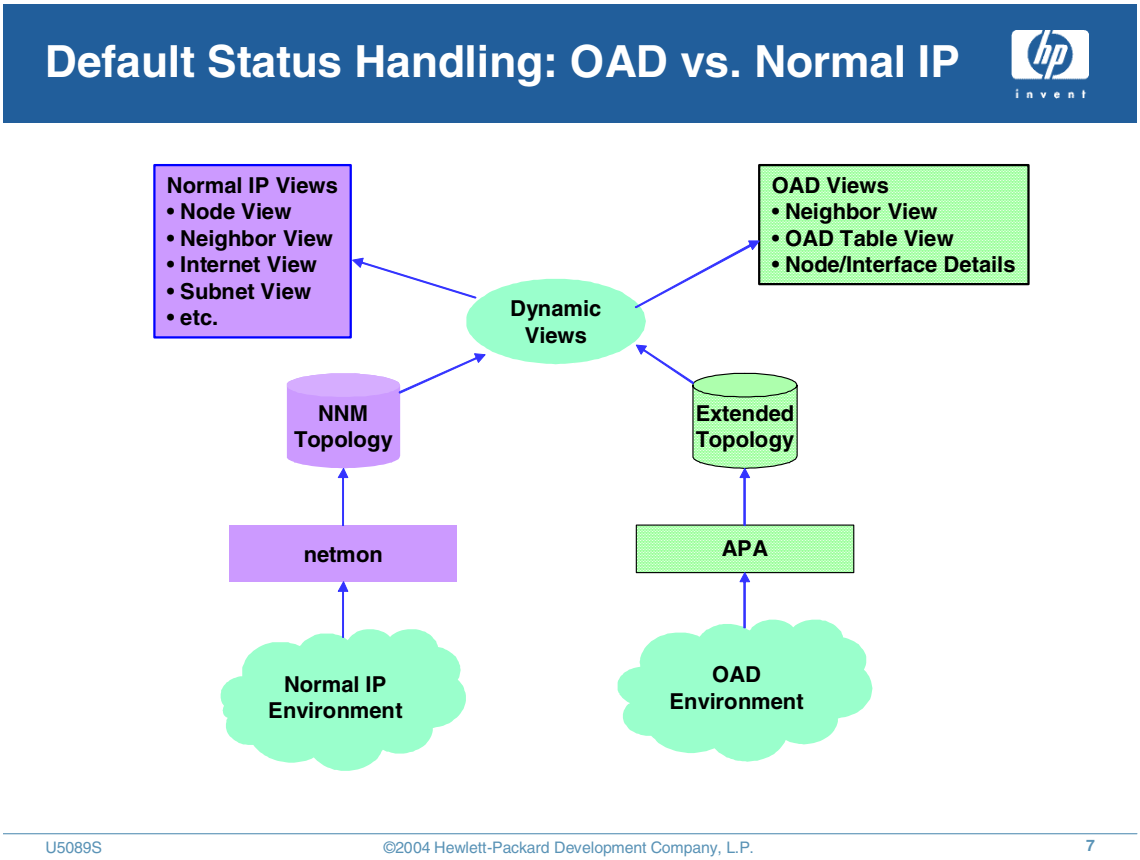
Does APA consider meshed switches when diagnosing network faults?

Answer: APA analyzes meshed environments when calculating the root cause of unresponsive interfaces.

Does APA analyze alarms sent to it from networked devices?

Answer: If a device sends an SNMP trap to NNM, APA does not do any further analysis on this event.

APA Handles Overlapping Address Domains



The Active Problem Analyzer supports Static NAT Overlapping Address Domain environments with NAT'd or routable addresses. In particular, the poller provides basic polling and analysis capabilities within an OAD environment. `netmon` or APA may be chosen to poll non-overlapping IP nodes. Only APA polls OAD nodes.

In the case of the polling and monitoring of the environment, the poller uses the management address when communicating with the target node, interface, or IP address. For objects without a management address, the object is ignored.

In the case of the analysis portion, there are two primary concerns. First, the events generated include the appropriate Overlapping Address Domain ID for identifying the object correctly. Second, the analysis itself needs to be careful of making conclusions in an OAD environment, handling crossing boundaries between zones, and making sure that payloads are interpreted correctly in an OAD context.

The poller supports the following features in an OAD environment:

1. Basic Status Polling
   - The management address is used for all SNMP and ICMP communication.
   - Objects without a management address will not be polled.
   - For interfaces with unreachable IP addresses, their status is determined solely via SNMP.

2. Fault Analysis
    - Features of basic fault analysis that work in a non-OAD environment also work in an OAD environment, including fault analysis in meshed and VLAN environments.
    - Status is updated for the correct objects in the Extended Topology database.
3. Event Generation and Handling
    - Target objects in events include the OAD, so that correct GUI launching can occur, for example from a Status Alarm to a Neighbor View.

Choosing Your General Poller

- Netmon still does discovery to find nodes
- APA can monitor all nodes in hosts.nnm
  - Turns netmon off for status polling of those nodes
- Stick with netmon if you are
  - Running on a Management Station with Collection Stations
  - Managing IPX
  - Using SPI for Frame or MPLS

U5089S ©2004 Hewlett-Packard Development Company, L.P. 8

By default APA monitors OAD and HSRP status only. If you want, APA can largely replace the `netmon` process for monitoring your whole network, but there are a number of factors to consider before you take that step.

NOTE Regardless of whether you have the `netmon` process or APA doing your general IPv4 network monitoring, the `netmon` process is still essential for device discovery.

Before you switch over to APA monitoring, however, you should understand the differences between the `netmon` process and APA. While APA offers several significant advantages over the `netmon` process, some users will find one or two specific characteristics of the `netmon` process that make it indispensable to them.

Situations that Require netmon Polling

You may have NNM installed as a management station in an NNM distributed environment. *Do not enable APA on NNM when it functions as a management station.* You can enable APA on NNM if it only functions as a collection station.

If you purchased and installed *HP OpenView Network Node Manager Smart Plug-in for Frame Relay* (version 2.0) or *Smart Plug-in for MPLS IP VPN* (version 1.0), you *must* continue using the

`netmon` process to poll your general IPv4 interfaces. *Do not enable APA on NNM to monitor your general IPv4 interfaces.*

APA does not support IPX networked devices, IPX interfaces, or any devices not residing in the `hosts.nnm` file.

Advantages of APA

APA offers more accurate diagnostic information and faster throughput for greater scalability. Here is a brief summary of the advantages APA offers when diagnosing network failures:

- It analyzes information from neighboring interfaces and verifies that the failure is real before generating the alarm.
- It identifies link failures based upon Extended Topology connectivity.
- It suppresses transient failures due to spanning tree reconvergence.
- It polls devices using both SNMP and ICMP as it deems appropriate for the situation.
- It improves polling performance by ignoring unconnected interfaces and utilizing multi-threaded processes.
- It updates status information for objects in both `ovw` and Dynamic Views.

How does APA affect NNM performance?

Answer: APA is multi-threaded, suppresses secondary alarms, and generates one alarm for a failure's root cause in most cases. This results in a faster polling engine.

Returing Status to ovw

Returning Status to ovw

- APA notifies ovtopmd of Normal->Critical or Critical->Normal status changes
- No status change to Unknown for secondary failures
- Dynamic Views get status from Extended Topology database

©2004 Hewlett-Packard Development Company, L.P.

If you have APA take over the general IPv4 interface polling, the `netmon` process stops monitoring these interfaces. As APA makes status changes to the extended topology database, it looks to see if the object has an NNM object ID (also appears in the NNM topology database) and sends updates about the statuses of these interfaces to the `ovtopmd` process.

`ovtopmd` generates status events as log-only and passes this status information to the `ipmap` process, and ultimately, the `ovw` user interface.

APA only updates the status of the device or interface identified as the primary failure and synchronizes only the primary node or interface status with the `ovtopmd` process and the `ovw` user interface. The result is that APA only updates the `ovw` user interface with the primary fault status. Once the faulty device is functioning properly, APA polls the device and the status returns to normal.

APA considers nodes located outside of the fault area to be secondary failures. APA does not update the `ovw` user interface with the status of these nodes. While they appear Unknown (blue) in dynamic views which take their status from the Extended Topology database, these nodes continue to appear Normal (green) or their previous status in `ovw`.

Dynamic Views get their information from the extended topology database if it is available for a node, returning to the `ovtopmd` process for status if necessary. Devices not being monitored by APA when APA has all status polling responsibilities are shown as having an `unknown` status.

If you view the `Node Details` or `Interface Details` pages of a device, it displays information as

follows:

- If APA is monitoring a node, the pages display extended topology and APA status information.
- If the `netmon` process is monitoring a node, the page displays status from the `ovtopmd` process.

How does APA derive node and interface status?

Answer: APA uses ICMP to determine address status. It separates interfaces from addresses in order to monitor interface status using both ICMP or SNMP.

Enabling and Disabling APA

Enabling and Disabling APA

- **ovet_apaConfig.ovpl -enable APAPolling**
  - Enables APA
  - Disables netmon status polling
- **ovet_apaConfig.ovpl -disable APAPolling**
  - Enables netmon status polling
  - APA remains responsible for HSRP and OADs

©2004 Hewlett-Packard Development Company, L.P.

If you choose to have APA monitor your general IPv4 interfaces, it disables the polling feature of the `netmon` process and enables APA.

NOTE APA takes its polling list exclusively from the extended topology database, which includes general IPv4 interfaces discovered by the `netmon` process and placed in the `hosts.nnm` file. You need to make sure that your important nodes are not blocked by the `bridge.noDiscover` file.

- To enable APA polling and disable status polling using `netmon`:

  ```
  ovet_apaConfig.ovpl -enable APAPolling
  ```

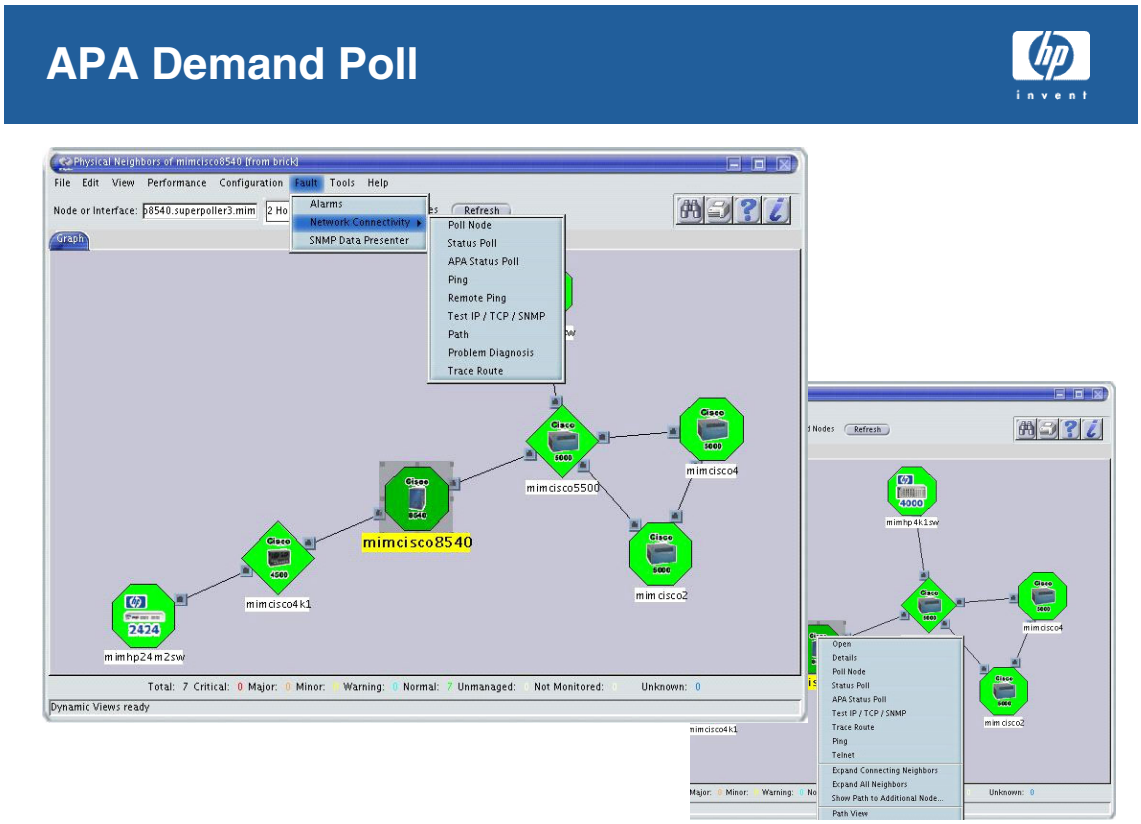
- To disable APA polling and enable status polling using `netmon`:

  ```
  ovet_apaConfig.ovpl -disable APAPolling
  ```

NOTE Your may choose to enable APA status polling, then disable it later. If you do this, APA continues to monitor addresses that you designate as belonging to an OAD, query HSRP routers, and report HSRP group status information.

See the `ovet_apaConfig.ovpl` reference page (or the UNIX manpage) for more information.

APA Demand Poll



You can demand poll (check status) a node or an HSRP group that is being monitored by APA.

For example, if a node has gone down and APA has marked the node as down in the Extended Topology database, when you fix the problem, you want APA to poll the node to show that the node is now up. (APA would discover this during the polling interval, but you may want to speed the process.) You can perform a demand poll on the node by selecting `Fault:Network Connectivity:APA Status Poll` and APA polls the node, updates the status of the node, and displays the results in an output window. In addition, a demand poll can sometimes be used to fix status on objects that did not analyze correctly or propagate status correctly.

To issue an APA demand poll for a node from the command line, type `ovet_demandpoll.ovpl` *nodename* where nodename is the host name or an IP address associated with the node.

You can specify an Overlapping Address Domain for the entity with the `-r` option, as in `ovet_demandpoll.ovpl -r` *OADID nodename*. If this option is used then the nodename or virtualIP is assumed to be a private IP address.

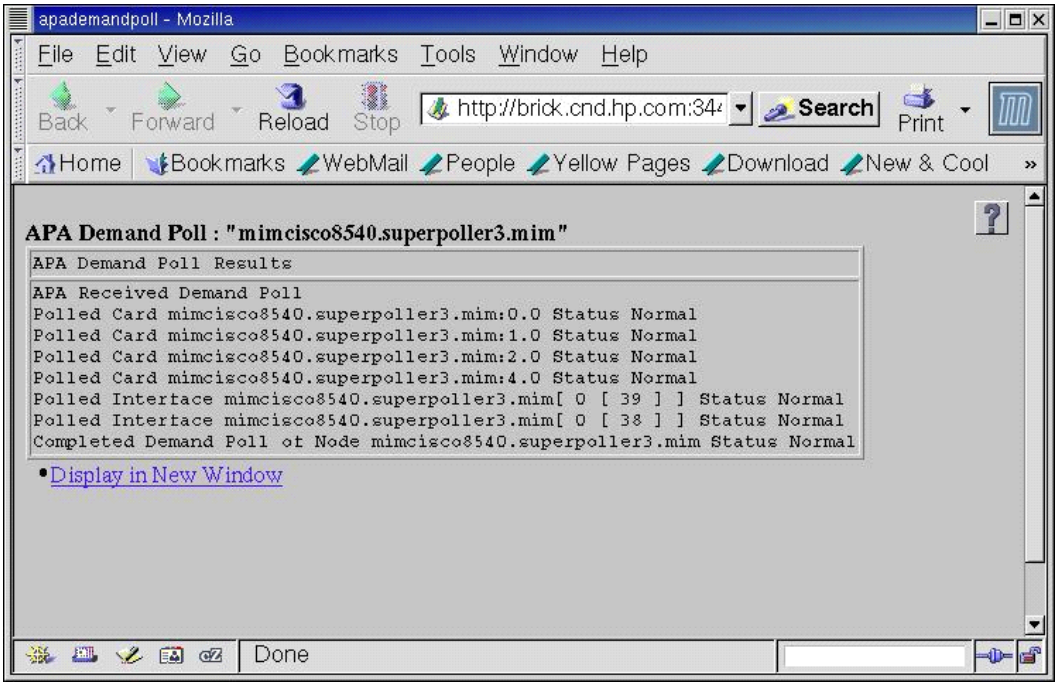
To specify an HSRP group to poll, provide the `-v` option and the virtual IP address for the HSRP group. The command looks like `ovet_demandpoll [-r OADID] -v virtualIP ...`

The demand poll functionality does not support demand polling from remote machines. Note that the Dynamic Views can accomplish this using WebApp.

The demand poll will only perform a status poll and will not perform a configuration poll.

Demand Poll Results

APA Demand Poll Output



 ©2004 Hewlett-Packard Development Company, L.P.

Demand poll output for nodes indicates the status of each sub entity (board or interface) contained in the node and the status of the node. The following is sample output from `ovet_demandpoll.ovpl`.

```
# /opt/OV/bin/ovet_demandpoll.ovpl mimcisco8540
APA Received Demand Poll
Polled Card mimcisco8540.superpoller3.mim:0.0 Status Normal
Polled Card mimcisco8540.superpoller3.mim:1.0 Status Normal
Polled Card mimcisco8540.superpoller3.mim:2.0 Status Normal
Polled Card mimcisco8540.superpoller3.mim:4.0 Status Normal
Polled Interface mimcisco8540.superpoller3.mim[ 0 [ 39 ] ] Status Normal
Polled Interface mimcisco8540.superpoller3.mim[ 0 [ 38 ] ] Status Normal
Completed Demand Poll of Node mimcisco8540.superpoller3.mim Status Normal
```

Note: If all interfaces were contained in boards then only the status for the boards would be displayed. In the above example the node has 2 interfaces which are not associated with boards.

The following is sample output for a poll of an HSRP group.

```
[C:/Program Files/HP OpenView/bin] ovet_demandpoll.ovpl  -v 15.2.129.1
APA Received Demand Poll
Interface c4k3-e0.cnd.hp.com[ 0 [ 3 ] ] HSRP State StandBy
Interface c2k3fa00.cnd.hp.com[ 0 [ 4 ] ] HSRP State Active
HSRP Group 15.2.129.1-0 Status Normal
```

The APA can be in the process of loading topology information or processing topology from a new

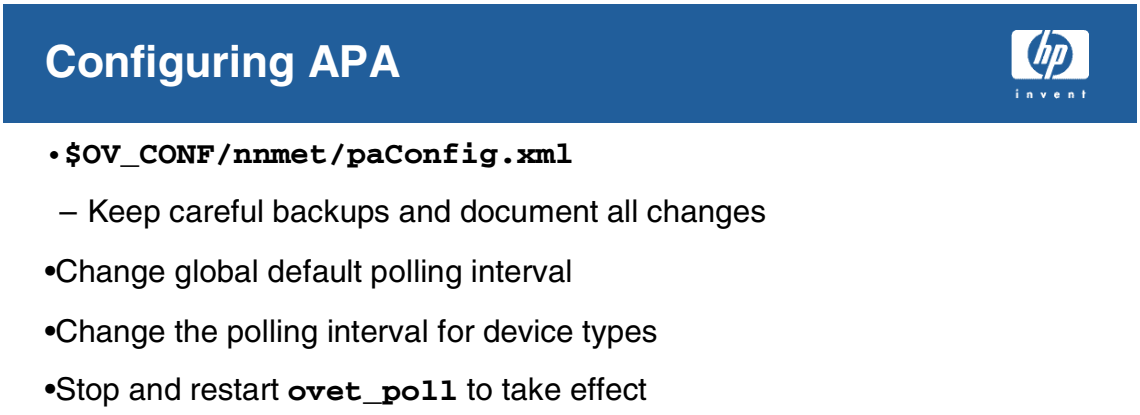
discovery. If a demand poll is performed while topology is not ready, the following error is displayed.

```
The APA is not currently able to process demand poll requests because the topology is either
not loaded or is being updated. Please try again later
```

If you select a node that is polled by `netmon`, not by APA, you will see

```
The APA is not currently polling the general IP environment please use ovet_apaConfig.ovpl to
turn on APA status polling for the general IP environment.
```

Configuring APA



You may need to adjust APA to optimize it for your particular environment. Prior to making any APA adjustments, it is a good idea to understand how APA is performing.

Adjusting APA Polling Parameters

APA allows you to manually adjust the status polling frequencies. You can adjust polling intervals based upon extended topology filters. Using this method, you can apply configuration attributes based upon the capability, or class, of a device.

NOTE You must have a basic understanding of XML concepts, terms, and syntax before attempting to edit the `paConfig.xml` file. Without that knowledge, it is very easy to make serious errors. This course assumes that you have adequate expertise in XML terminology and syntax, and does not define XML terms or explain XML tagging or syntax.

You can make modifications to APA polling frequencies. However there are only a few parameters you will want to modify. The following file contains these adjustable parameters:

- *Windows*: `%OV_CONF%\nnmet\paConfig.xml`
- *UNIX*: `$OV_CONF/nnmet/paConfig.xml`

The paConfig.xml file contains the following sections:

- *Polling Engine*

  You can easily recognize this section, as the following tags begin the Polling Engine section of the xml file:

  ```
  <subSystemConfig>
      <name>PollingEngine</name>
      <title>Polling Engine</title><subSystemConfig>
  ```

- *Status Analyzer*

  You can easily recognize this section, as the following tags begin the Status Analyzer section of the xml file.

  ```
  <subSystemConfig>
      <name>StatusAnalyzer</name>
      <title>Status Analyzer</title>
  ```

- *Talker*

  You can easily recognize this section, as the following tags begin the Talker section of the xml file.

  ```
  <subSystemConfig>
      <name>Talker</name>
      <title>Talker SubSystem</title>
  ```

- *StatusBridge*

  You can easily recognize this section, as the following tags begin the Status Bridge section of the xml file.

  ```
  <subSystemConfig>
      <name>StatusBridge</name>
      <title>Status Bridge</title>
  ```

By modifying this file, you can adjust global polling parameters as well as the polling frequency of classes of devices such as routers, switches, and end nodes.

Changing the Default Polling Interval

Suppose you want to modify the default polling interval for devices not belonging to a specific device class. To do this, use the following procedure:

1. Create a backup copy of the paConfig.xml file prior to making any changes.

CAUTION	Be sure to keep careful records and backups of any and all changes to the paConfig.xml file.

2. As Administrator or root, edit $OV_CONF/nnmet/paConfig.xml.
3. Look for the following polling interval text:

```
<classSpecificParameters>
<!-- Default parameters for when no class of device is specified. -->
<defaultParameters>
            <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>
```

Change the bold number to the number of seconds you want APA to wait before again polling devices that do not belong to any defined device class. Make sure to save your changes.

4. As `Administrator` or `root`, run the following commands:

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

     This applies your changes to the APA polling process.

Changing the Polling Interval by Device Class

The `paConfig.xml` file references filters that allow you to modify polling intervals by device class. The polling interval of a device class takes precedence over the default polling interval.

Suppose you want to modify the router polling frequency.

Look for the `isRouter` filter text that looks as follows:

```
<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>
```

Modify the bold number to the number of seconds you want APA to wait before repolling devices that pass the `isRouter` filter. Make sure to save your changes.

The filter name is shown in bold print in the above example. Using a similar procedure, you can modify the polling frequency for devices that pass other topology filters. Look for filters with the following names:

- isSwitch
- isEndNode

Using Topology Filters

 ©2004 Hewlett-Packard Development Company, L.P.

The `paConfig.xml` file uses extended topology filters to limit the devices to which you make polling configuration changes. To get a listing of the filter names, use the `ovet_topodump.ovpl -lfilt` command. To verify the nodes selected by a filter, run the `ovet_topodump.ovpl -node -filt` *`filter_name`*. See the *ovet_topodump.ovpl* reference page (or the UNIX manpage) for more information.

Implementing a Topology Filter

The Polling Engine subsystem in the `paConfig.xml` file contains a `Polling Settings configGroup` that contains polling configuration parameters. Within this `configGroup`, the set of parameters located beneath the `<classSpecific Parameters>` xml tag control how APA polls devices if they do not pass any of the extended topology filters within the configGroup list.

If a device belongs to any of the device classes specified beneath the `<classSpecification>` xml tag, then APA polls the device according to the parameters contained within the specified class.

For example in the xml listing below, the first referenced filter, `isRouter`, contains a polling interval integer that you can adjust.

```
<classSpecifications>
            <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>
```

You can use extended topology filters to add device classes into the `paConfig.xml` file to categorize devices in your environment. For example, if you want to create one location in the `paConfig.xml` file to adjust the APA polling of your HP ProCurve devices, use the following procedure.

1. Run the `ovet_topodump.ovpl -lfilt` command to see a list of the existing filters. The output will look something like this (in a single column on the screen):

```
AccessPort                  AdminDownInterface          AdminUpInterface
AlcatelDevices              BayDevices                  CiscoDevices
CiscoRouter                 ConnectedEndNode            ConnectedIF
ConnectedIFInInfrDev        ConnectedNode               ExceptionInfrDev
ExceptionNode               ExtremeDevices              InfrDev
InterfaceInEndNode          InterfaceInInfraDev         InterfaceInRouter
InterfaceInSwitch           NodeWithDownInterface       NonSnmpNode
NonSnmpSwitch               NortelDevices               NotConnectedIF
NotConnectedNode            NotConnectedSnmpRouter      NotConnectedSnmpSwitch
ProcurveDevices             SnmpEndNode                 ThreeComDevices
TrunkPort                   UnconnectedAdminUpRouterIF  UnconnectedAdminUpSwitchIF
UnconnectedEndNode          isATM                       isCDP
isEndNode                   isFrameRelay                isHSRP
isRouter                    isSnmp                      isSwitch
myPetNodes
```

2. Run the `ovet_topodump.ovpl -node -filt ProcurveDevices` command to get a list of all of the ProCurve devices in your environment.
3. If the list of devices that this command produces contains the devices you want to manage as a class, complete this step. As `Administrator` or `root`, open the `paConfig.xml` file and find the following xml tags.

```
<classSpecifications>
        <!-- Specify the parameters and the class of devices these
             parameters are for. -->
        <classSpecification>
```

4. Add the following filter text beneath the <classSpecification> xml tag. Do not place the new text within the text of any of the existing filters. The filters you reference in paConfig.xml are prioritized from the top down, so the order in which you add filters matters. Make sure you add your filters before or after other filters as appropriate.

```
<filterName>ProcurveDevices</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>
```

Make sure to save your changes.

5. As Administrator or root, run the following commands:

   - ovstop -c ovet_poll
   - ovstart -c ovet_poll

     This applies your changes to the APA polling process.

Controlling HSRP Polling

To disable monitoring of HSRP groups, rerun setupExtTopo.ovpl and answer No when asked whether you want to monitor HSRP. The following procedures adjust HSRP status polling only, leaving discovery intact.

Enable or Disable Global HSRP Group Polling

You can enable or disable global parameters in the `paConfig.xml` file that override any class-based parameters. For example, to enable or disable HSRP group polling, use the following procedure:

Look for the following HSRP group polling enabling text:

```
<parameter>
            <name>HSRPPollingEnable</name>
            <title>Enable HSRP Polling</title>
            <description>
              Enable/Disable polling of HSRP Groups.
            </description>
            <varValue>
              <varType>Bool</varType>
              <value>true</value>
            </varValue>
```

```
</parameter>
```

Modify the bold `true` to `false` if you want to disable the global HSRP polling or `true` if you want to enable the global HSRP polling. Make sure to save your changes.

Enable or Disable HSRP Polling for Devices not Belonging to a Device Class

You may want to modify parameters for devices not belonging to a device class. For example, to enable or disable HSRP group polling for routers and interfaces that do not belong to a device class, use the following procedure:

Look for the following HSRP group polling enabling text:

```
<parameter>
                    <name>hsrpEnable</name>
                    <title>Enable HSRP Polling</title>
                    <description>
                      Enable/Disable polling of HSRP Group.  If the router/interface
                      does not have hsrp enabled then setting this parameter to
                      true does nothing.
                    </description>
                    <varValue>
                      <varType>Bool</varType>
                      <value>true</value>
                    </varValue>
                  </parameter>
```

Modify the bold `true` to `false` if you want to disable HSRP polling or `true` if you want to enable HSRP polling. This parameter enables or disables HSRP polling *for routers, ports, and interfaces*. Make sure to save your changes.

NOTE If you do not want APA polling your HSRP groups, run `setupExtTopo.ovpl` and answer no when asked if you want to enable HSRP polling.

Polling Protocols for Network Devices

The Active Problem Analyzer determines the appropriate polling protocol for each topology object based on the object type.

Routers receive an SNMP query on connected interfaces and pings on all IP addresses.

Switches receive the SNMP query on all connected interfaces.

End nodes receive a ping of all IP addresses.

Controlling Polling Protocols

- Default polling via SNMP to false or true
- Default polling via ping to false or true

©2004 Hewlett-Packard Development Company, L.P.

Enable or Disable SNMP Polling of Devices not Belonging to a Device Class

This is an example of adjustable parameters for devices not belonging to a device class. For example, to enable or disable SNMP polling for devices that do not belong to a device class, use the following procedure:

Look for the following SNMP polling enabling text:

```
<classSpecificParameters>

              <!-- Default parameters for when no class of device is specified. -->
              <defaultParameters>
                <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>
```

Modify the bold `true` to `false` if you want to disable SNMP polling for devices that do not belong to a device class, or `true` if you want to enable SNMP polling for devices that do not belong to a device class. Make sure to save your changes.

Enable or Disable ICMP Polling of Devices not Belonging to a Device Class

This an example of adjustable parameters for devices not belonging to a device class. For example, to enable or disable ICMP polling for devices that do not belong to a device class, use the following procedure:

Look for the following SNMP polling enabling text:

```
<classSpecificParameters>

    <!-- Default parameters for when no class of device is specified. -->
    <defaultParameters>
      <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>
        <parameter>
          <name>pingEnable</name>
          <title>Enable polling via ICMP</title>
          <description>
            Enable/Disable polling of a device via ICMP.
          </description>
```

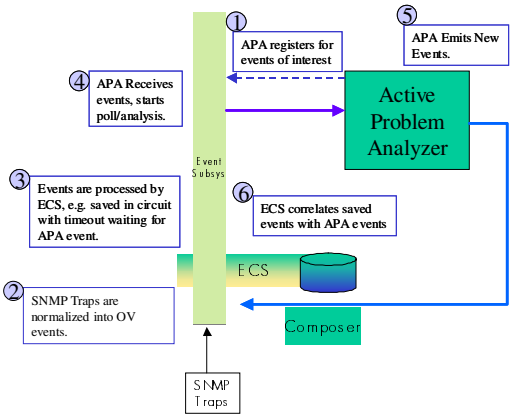
```
    <varValue>
      <varType>Bool</varType>
      <value>true</value>
    </varValue>
  </parameter>
```

Modify the bold `true` to `false` if you want to disable ICMP polling for devices that do not belong to a device class, or `true` if you want to enable ICMP polling for devices that do not belong to a device class. Make sure to save your changes.

APA Event-Triggered Polling

APA Event-Triggered Polling

- Initiate a timely status poll in response to an external stimulus:
  - A trap from a device (i.e. linkup)
  - A syslog event
  - RAMS/OSPF Adjacency Failure
- Correlate stimulus with any resulting events
- Related to but distinct from demand poll
  - Demand poll is user initiated, with a desire for output of results.
- Significant performance improvement over waiting for default polling cycle.



 ©2004 Hewlett-Packard Development Company, L.P.

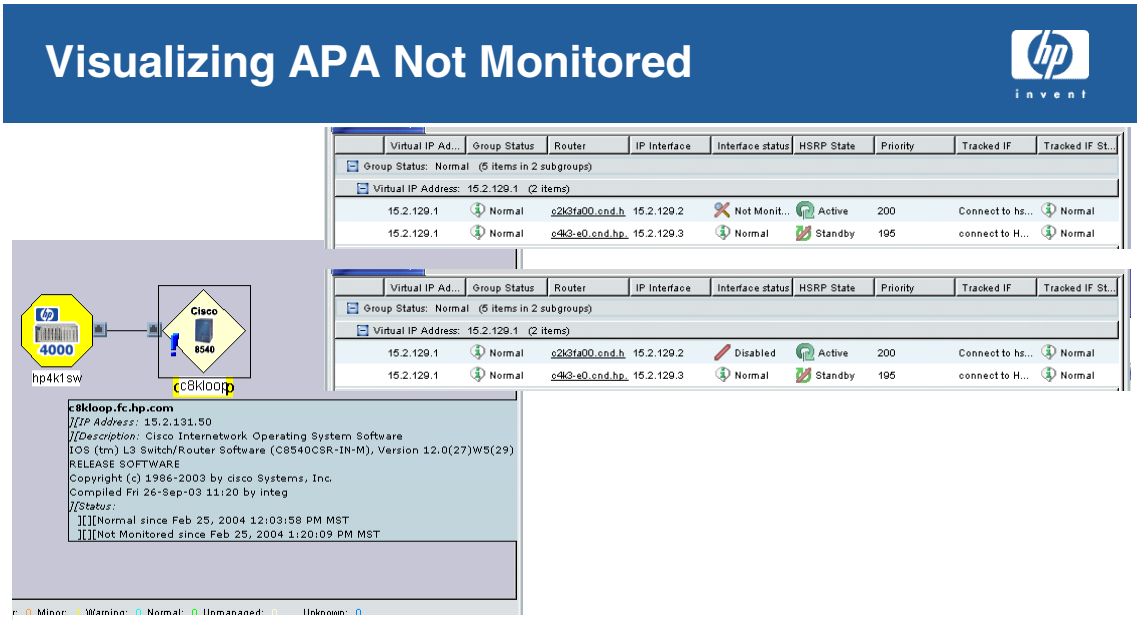
OpenView events, which include SNMP traps from network entities can trigger APA polling and analysis. This allows you to get faster results from the APA. You no longer have to wait for a polling cycle to see certain faults in the network under management, as the SNMP traps will trigger a poll of the failed device.

Correlators monitor incoming traps and syslog messages and may notify APA to initiate a poll on a device. By default, only one trigger is sent in a 5 minute window. When APA receives an event, it does the most focused possible analysis to determine what has changed in the network. The following table describes event-triggered analysis:

Input Trap or Event	**Analysis Triggered in Status Analyzer**
Generic SNMP Link Down/Up trap	interface analysis
HSRP State Change Trap	HSRP group analysis
Generic SNMP Cold start/Warm start Cisco Cold start / Warm start	node analysis. This will also trigger a configuration poll.
CISCO board trap or syslog message	configuration poll, plus a board added/deleted/issue triggers a node analysis and a board up/down triggers a board analysis

Input Trap or Event	Analysis Triggered in Status Analyzer
Syslog or RAMS specific trap/event information	APA poll to validate state or data
CISCO specific trap for link down or link up	interface analysis and configuration poll
Cisco Port Aggregation trap or syslog message	interface analysis

Visualizing APA Not Monitored



APA handles Not Monitored and Disabled statuses.
Dynamic pie, graph, HTML and table views show the new status.
Created two new icons to represent not-monitored and disabled in tables.
Reused the unmanaged and disabled colors.

 ©2004 Hewlett-Packard Development Company, L.P.

Customers do not like the NNM 7.01 APA behavior of setting object status in dynamic views to ‘green’ when in fact APA has been configured to not poll the object. Customers want the dynamic views to use a different color to represent these unpolled objects. The request is that the status be something other than “Normal” or “Unknown.” Normal means that there is an active assertion that everything is OK, and Unknown is used to indicate secondary failures. They also want disabled interfaces to appear with the Disabled status.

If you determine that APA is polling a set of nodes and/or interfaces unnecessarily, you can reconfigure APA to ignore these nodes/interfaces. When operators view status for these not-polled nodes/interfaces, the table shows a Not Monitored symbol identifying the object as not being monitored.

For example, you may want to disable (adminDown or Testing) an interface on a network device. When operators view status for these disabled interfaces, the table shows the disabled symbol identifying the object as being administratively downed.

APA sets the status value for unmonitored devices to “No Status” in the Extended Topology database. In NNM 7.5, dynamic views use Not Monitored (tan) for ball and port icons and the Not Monitored icon for tables.

Icons for disabled objects use the Disabled color (sienna) for Disabled status. A new icon for Disabled has been created.

When APA is monitoring all devices in the environment and status is passed back through `ovtopmd` to `ovw`,

- Not Monitored objects appear as Normal
- Disabled objects appear as Critical
- Objects that do not appear in the Extended Topology database are set to Unknown

Status calculation for Problem Diagnosis displays does not involve APA. The status of the path is derived from the ping status of all the nodes on the path. Unset status from APA is not considered in this calculation.

Setting Objects to Not Monitored

Setting Objects to Not Monitored

- ovet_toposet
  - -a to allow polling or –s to suppress polling
  - [-node [objid | nodeName]]
  - [-nodeif [objid | nodeName]] [-if [objid | interfaceName| ifAlias| ifDescription]]
  - [-board [objid | nodeName]] [-index <index>] [-subindex <subindex>]
  - [-addr [-ipv4|-ipv6] [IP Address]] [-OADId <OADId>]

- Overrides paConfig.xml when paConfig.xml allows monitoring.
- Does not override when paConfig.xml disables monitoring.

©2004 Hewlett-Packard Development Company, L.P.

You can allow or suppress the APA monitoring of objects such as nodes, interfaces, boards, and addresses. Each operation you perform on an object also includes all contained objects.

When you allow or suppress polling for a(n)	The utility will poll or stop polling
node	all the contained objects of the node. In this case interfaces and boards of the node and the addresses of all the interfaces of the specific node.
interface	all the contained objects of the interface. In this case all the addresses of the particular interface.
board	all the contained objects of the boards. In this case all the interfaces the board contains and all the addresses the interfaces contain.
address	only the specified address.

`ovet_toposet` script has the following command line options.

```
ovet_toposet -h

ovet_toposet [-a|-s] [-node [objid | nodeName] ]

ovet_toposet [-a|-s] [-nodeif [objid | nodeName] ]  [-if [objid |
interfaceName|ifAlias|ifDescription] ]

ovet_toposet [-a|-s] [-board [objid | nodeName] ] [-index <index>] [-subindex <subindex>]

ovet_toposet [-a|-s] [-addr [-ipv4|-ipv6] [IP Address]] [-OADId <OADId>]
```

Parameters:

`-h` = help: This option shows the help for the command.

`-s` = suppress: This option is used to stop polling a particular object.

`-a`= allow: This option is used to allow polling for a particular object.

`-node [objid | nodeName] ]`: This –node option is used to manage/un-manage a node. This option manages/un-manages all the containment objects of the particular node. i.e. all the interfaces, boards and all the addresses of the interfaces.

`-nodeif [objid | nodeName] ] -if [objid | interfaceName|ifAlias|ifDescription]]`: The –nodeif option is used to manage/un-manage an interface. This option manages/un-manages all the containment objects of the particular interface. i.e. all the addresses of the interfaces.

`-board [objid | nodeName] ] [-index <index>] [-subindex <subindex>]`: The –card option is used to manage/un-manage a board. This option manages/un-manages all the containment objects of the particular board. i.e. all the interfaces and all the addresses of the interfaces. If for a particular instance of the board, all the interfaces are participating in an aggregated link, then that aggregated link will also be managed/un-managed. However if some of the interfaces of the board are participating in a aggregated link or all interfaces are participating in various aggregated links, then the aggregated links will not be changed.

`-addr [-ipv4|-ipv6] [IP Address]] [-OADId <OADId>]`: This option is used to manage/un-manage an address. This option manages/un-manages the particular address. The –ipv4/-ipv6 option refers to IP version – either 4 or 6. One can supply the public IP address or private IP with OADId option. The -OADId is an optional parameter. However this parameter needs to be supplied if the IP address supplied is private IP Address.

The command overrides `paConfig.xml` specifications for objects which `paConfig.xml` would allow to be polled. You cannot use this to start monitoring an object that has polling disabled in `paConfig.xml`.

Events and Status

APA Event/Status Handling

- Goal:
  - One viewable event per fault (root cause)
  - Accurate diagnosis
  - Related events linked
  - Good performance in the event system
- Design
  - Status is something potentially viewable. (May have a heuristic involved in the computation.)
  - Events are decoupled from Status (May set the status on all interfaces and nodes but only send out primary failure events.)

U5089S ©2004 Hewlett-Packard Development Company, L.P. 21

APA may issue one meaningful event for multiple updates to status. It is important to distinguish between the status and the event. In contrast, `netmon` and `ovtopmd` couple object status with the event emitted.

How APA Looks at Failures

APA diagnoses network failures by doing additional failure analysis. During this failure analysis, APA does not generate many alarms for nodes that are not immediately adjacent to the detected fault. It attempts to generate one alarm showing the root cause of a failure.

If APA determines that a node is down, it generates an `OV_APA_Node_Down` alarm for the node and sets the status to critical. This alarm name implies a primary failure.

Failures detected on the other side of a known fault are a result of that fault, and APA calls them secondary failures. You can see these secondary failures by drilling down from the corresponding primary failure. This eliminates alarms from the Alarm Browser for devices that are not known to have failed.

Understanding APA and Event Reduction

When you enable APA, it disables the status polling feature of the `netmon` process and establishes the `ovet_poll` process as the polling engine. The `ovet_poll` process contains new alarms types. The following list describes the new alarms that APA may display:

- *OV_APA_Message*: APA generates this alarm to reflect changes in the network that are best communicated through text. This alarm is mainly used for internal troubleshooting.
- *OV_APA_ADDR_UP*: APA generates this alarm when it detects that a network entity's address status goes from *down* to *up*.
- *OV_APA_ADDR_DOWN*: APA generates this alarm when it detects that a network entity's address status goes from *up* to *down*.
- *OV_APA_ADDR_Intermittent*: APA generates this alarm when it detects that a network addresse's status has gone *down* and *up* multiple times.
- *OV_APA_ADDR_UNREACHABLE*: APA generates this alarm when it detects that an address is unreachable due to another failure, such as the addresse's interface being down.
- *OV_APA_IF_UP*: APA generates this alarm when it detects that a network entity's interface status goes from *down* to *up*.
- *OV_APA_IF_DOWN*: APA generates this alarm when it detects that a network entity's interface status goes from *up* to *down*.
- *OV_APA_IF_Intermittent*: APA generates this alarm when it detects that an interface's status has gone *down* and *up* multiple times.
- *OV_APA_IF_UNREACHABLE*: APA generates this alarm when it detects that an interface is unreachable due to another failure.
- *OV_APA_NODE_UP*: APA generates this alarm when it detects that a node's status goes from *down* to *up*.
- *OV_APA_NODE_DOWN*: APA generates this alarm when it detects that a node's status goes from *up* to *down*.
- *OV_APA_NODE_Intermittent*: APA generates this alarm when it detects that a node's status has gone *down* and *up* multiple times.
- *OV_APA_NODE_UNREACHABLE*: APA generates this alarm when it detects that a node is unreachable due to another failure, such as and upstream node being down.
- *OV_APA_CONNECTION_UP*: APA generates this alarm when it detects that a connection's status goes from *down* to *up*.
- *OV_APA_CONNECTION_DOWN*: APA generates this alarm when it detects that a connection's status goes from *up* to *down*.
- *OV_APA_CONN_Intermittent*: APA generates this alarm when it detects that a network connection's status has gone *down* and *up* multiple times.
- *OV_APA_CONNECTION_UNREACHABLE*: APA generates this alarm when it detects that a connection is unreachable due to another failure.

NOTE APA does not actually generate the `IntermittentStatus` alarms, they are generated by the `OV_PollerPlus` correlators. These correlators are contributed software and, if enabled, generate the `OV_APA_ADDR_Intermittent`, `OV_APA_IF_Intermittent`, `OV_APA_NODE_Intermittent`, and `OV_APA_CONN_Intermittent` alarms. This only happens after you enable

correlators in the `OV_PollerPlus` namespace and after APA generates multiple Down events within the time window specified in those correlators.

The `OV_PollerPlus` correlators are contributed, and therefore unsupported. If you still want to enable the `OV_PollerPlus` correlators, add a `PollerPlus.fs` entry to the `NameSpace.conf` file and redeploy the correlators with the Correlation Composer window. For specific deployment information, refer to the *HP OpenView Correlation Composer's Guide*.

For more information about APA alarms, see the *trapd.conf* reference page (or the UNIX manpage).

As mentioned earlier, APA generates alarms for primary failures in most cases. This eliminates alarms from the Alarm Browser for devices that are not known to have failed. NNM also includes several of the new APA alarms in existing ECS correlations such as `ConnectorDown` and `Pair Wise` correlations.

For more information about alarm reduction with NNM, see the *Managing your Network* manual.

Comparing NNM and APA Status Handling

NNM and Extended Topology Status Handling

Status Handling in NNM:	Status handling in Extended Topology:
•Address is part of Interface object	•Separate Interface and Address objects, with potentially multiple addresses per interface, and each with its own state.
•Interface status can be derived from ICMP or SNMP, not both.	•Both ping an IP address AND query an interface state via SNMP.
•Interface status propagates and is sole basis for node, segment, network status.	•Some objects have their own basis for status, e.g. HSRP, with a desire to mix the various state values for status.
•A “Node Down” event is both the notification of a problem AND the notification of a status change.	•Conceptually, the poller separates event generation from state/status updating in the topology.
•Status change events are emitted by the topology.	•The topology has a notification API, which uses the event system as in NNM.
•ipmap listens to the various status change events to update ovw.	•There is a status bridge to reflect the status from NNM/ET backwards into NNM.
•Secondary failures always generate events for each object.	
•Dynamic Views gets status from NNM topology for nodes/interfaces.	•Dynamic Views gets status from Extended Topology for OAD and HSRP objects.

U5089S ©2004 Hewlett-Packard Development Company, L.P. 22

Understanding How APA and the netmon Process Cooperate

Table 8-1 compares APA with the `netmon` process. This should help clarify some of the differences between the `netmon` process and APA.

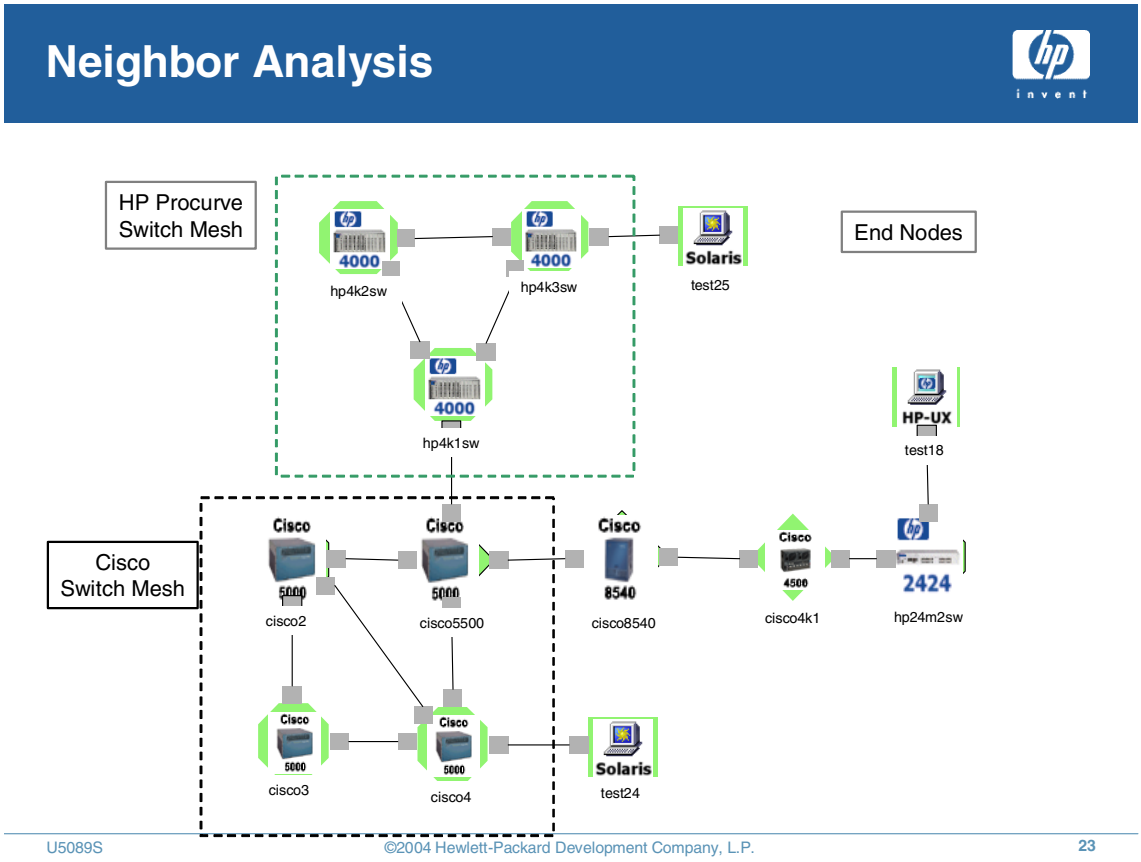
Table 8-1 Comparing APA and the `netmon` Process

NNM Prior to Enabling APA	NNM with APA Enabled
NNM’s polling list comes from `ovtopmd`.	APA takes its polling list exclusively from the extended topology. You need to make sure that your important nodes are not blocked by the `bridge.noDiscover` file.
APA does not generate status alarms for devices in the general IP environment. NNM alarms appear as they normally do.	NNM generates APA alarms only. It does not generate NNM status alarms. See the *trapd.conf* reference page (or the UNIX manpage) for more information.

Table 8-1 Comparing APA and the netmon Process (Continued)

NNM Prior to Enabling APA	NNM with APA Enabled
NNM events participate in event correlation and appear in the Alarm Browser.	APA events participate in event correlation and appear in the Alarm Browser after verification.
Root cause analysis is based on `netmon` process information available to ECS. Alarms are sent for every device affected by a failure. Events that communicate topology changes to other NNM processes may be logged and not used by ECS.	Root cause analysis is based on APA's more detailed information available to ECS. A single alarm is sent only for the actual root cause, not one for each affected device.
NNM considers an address to be an interface object.	There are separate interface and address objects. Extended Topology and APA allows for multiple addresses per interface with each interface having its own status.This clarity of status allows better propagation to node status.
The `netmon` process derives device interface status using ICMP (or SNMP if the device is a switch). You can add address information to the `netmon.snmpStatus` file to have NNM use SNMP status polling for specific interfaces. Doing this turns off ICMP polling for the targeted interfaces. See the *netmon* reference page (or the UNIX manpage) for more information.	APA uses both ICMP and SNMP to determine interface status, depending on the APA configuration.
`ovw` shows propagated interface status. Status propagation is the sole basis for node, segment, and network status.	Some objects have their own basis for status. For example, HSRP analyzes the various status values to determine HSRP group status.
Internally, the `ovtopmd` process generates status change events for each device affected by a failure.	The `ovet_poll` process generates status alarms. It does not send secondary failure alarms. APA changes the status in the extended topology and only the root cause event is sent through the event system.
The `ipmap` process listens to the various status change alarms and updates node, segment, and network status.	Dynamic Views listen for and display topology changes.The `ovet_poll` process sends Extended Topology node status to `ovw` nodes, segments, and networks.
Dynamic Views get status from the `ovtopmd` process for nodes and interfaces.	Dynamic Views get status from the extended topology. If you enable APA for IP, the `ovtopmd` process gets its IP status from APA.

Neighbor Analysis: Connected Ports



From a user's perspective, the behavior is:

If a node is not responding to a status poll, its immediate neighbors (as determined by the Extended Topology discovery) are queried:

- If exactly one neighbor is responding to SNMP/ICMP, and reports exactly one interface connected to the target node as down, then:
  - — The interface is considered a critical interface failure.
  - — A root cause connection down event is emitted.
  - — The node is considered "unreachable." A correlated node unreachable event is emitted.

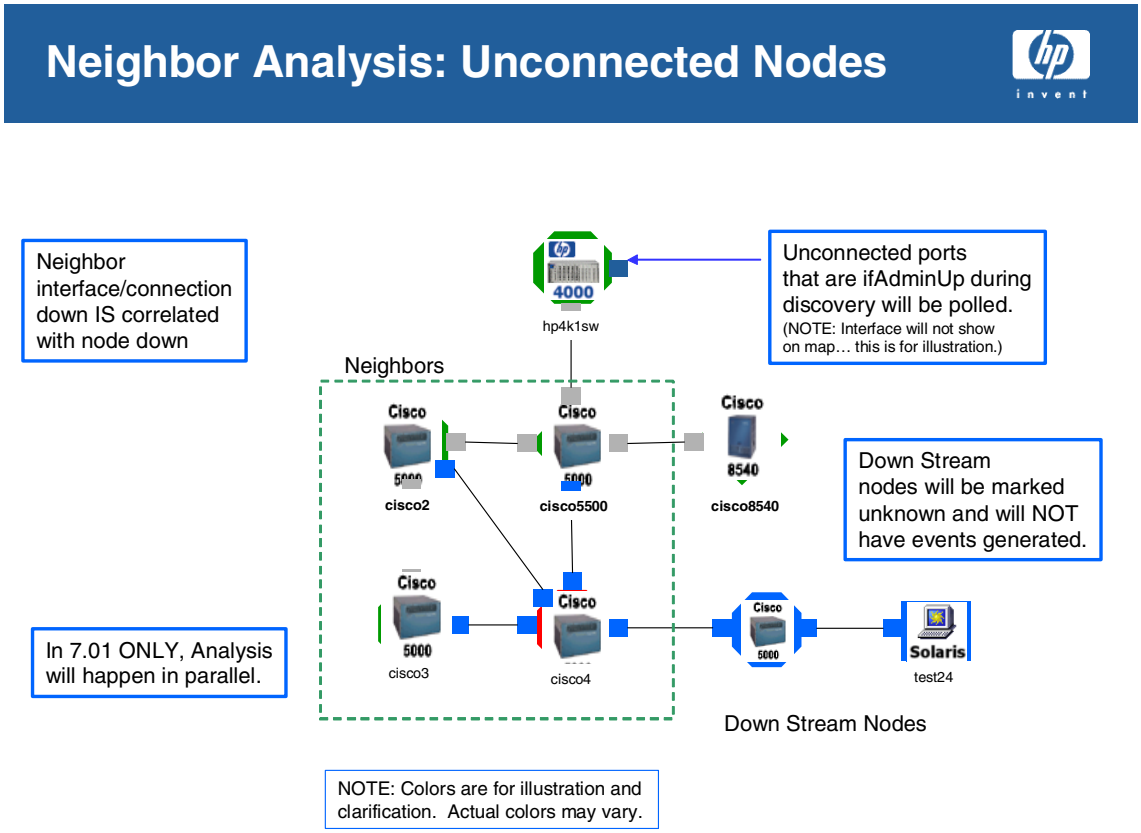
NOTE Note that the "Down" event always indicates the primary failure and a secondary failure is always indicated by an "Unreachable" event.

- If more than one neighbor is responding to SNMP/ICMP and the nodes report an interface connected to the target node as down, then:
  - — The node is considered as down. A root cause node down event is emitted.
  - — The interface is considered unreachable. A correlated connection unreachable event is emitted.

- If at least one neighbor is responding to SNMP, but no neighbors report a connected interface as down, then:
  - The node is considered down. A root cause node down event is emitted.
- If no neighbors are responding: If no neighbors are responding, then this is a secondary failure, (you are in the Far-From-Fault area) and some other primary failure has been sent to the Operator.
  - The node is considered “unreachable”.
  - NO node unreachable event is emitted. Although the status is changed internally, the internal event is not propagated to the Operator to avoid excess noise.
- If there are no neighbors, the node is considered an “island” node:
  - The node is considered down.
- For handling “important” devices, a flag is provided in `paConfig.xml` that forces the node unreachable (secondary failure) event to always be emitted.

SPECIAL CASE: Egress routers (and NAT devices) in the OAD configuration will be considered as down even if no neighbor is responding, essentially treating them ALWAYS as “important”. Egress routers are denoted in `dupip.conf` via the “NextHop” setting.

Neighbor Analysis: Unconnected Ports



The Advanced Problem Analyzer (poller) polls interfaces/addresses for the purposes of reporting status and events about the node being polled. The determination of whether to poll a particular interface or address is made based on a set of attributes of the node and interface. In particular, the poller pays attention to whether the interface is known to be connected to another node in the Extended Topology, and to the class of the device being polled (router, switch, end node). The actual configuration to make these determinations is done via the specification of appropriate topology filters and class definitions in the poller configuration file.

One problem with this approach is that some devices will not be in the Extended Topology, or will not connect correctly. In particular, end nodes that do not support SNMP in an OAD environment will likely not be connected correctly, or not be available in the Extended Topology.

The unconnected port algorithm attempts to partially address this issue by giving the user a way to enable and control which ports are polled based on the `ifAdminStatus` of the ports in question. More specifically, the algorithm is that:

- The Extended Topology discovery process discovers and updates the `ifAdminState` field in the topology during both full and incremental discovery, similar to other interface attributes such as `ifSpeed`.
- The Extended Topology attribute `ifAdminState` handling is set and updated from the discovery results. This includes recording the appropriate updates into the Audit trail after an incremental discovery.

- The poller honors the setting of the `ifAdminState`, and through filter configuration polls interfaces that either are known to be connected in the topology, OR which have an `ifAdminState` of `Up`.
- That poller configuration is set to OFF by default for switches, as the default behavior for switches is that a large number of ports are `ifAdminState` = up, but `ifOperStatus` = down, meaning that for most customers this would be an undesirable setting. To make use of this feature successfully, a customer must be willing to manually adjust the `ifAdminStatus/State` of ports on switches to match ports that are known to be connected.
- On initial discovery, `ifAdminState` is populated correctly for all interfaces on all nodes.
- On incremental discovery, changes in `ifAdminState` are detected and updated in the Extended Topology database.
- Classes control the poller configuration such that:
  - — Unconnected routers interfaces which are administratively up are enabled for SNMP and ICMP polling.
  - — Unconnected switch interfaces which are administratively up are disabled for SNMP and ICMP polling.
  - — Unconnected interfaces will have SNMP and ICMP polling disabled.

NOTE Due to the way filtering and class definitions work, there is a “fall through” kind of behavior, so that the first class definition that matches for a particular type of object applies. By having multiple versions, this makes it very easy to switch the behavior for a class of device, simply by changing the settings for that type of device. This implies the ability to define filters that are *interface* filters that include a component of the node attributes in the filter definition.

The poller detects changes in `ifAdminState` on incremental discovery, and updates the polling parameters and handling based on the new state.

The key UI issue is that there is no visual representation of unconnected interfaces in Neighbor Views, so the interface in question is not easy to examine without going to the Node Details views. The node and interface details show the correct values.

Important Nodes Filter

Important Nodes Filter

- Ensures that status alarms from important nodes appear in the Alarm Browser
- Visualization status is not affected by the important node filter.

- Filter is already in paConfig.xml
- Add important nodes to

  $OV_CONF/nnmet/topology/filter/MyHostID.xml

  <IPv4><address>15.172.20.38</address></IPv4>
- Can use wildcards, ranges

©2004 Hewlett-Packard Development Company, L.P.

The APA `ovet_poll` executable now treats secondary failure nodes differently if you have specified the nodes to be Important using the Extended Topology Filtering mechanism and `paConfig.xml`.

By default, APA does not generate alarms for devices that are located in the Far-From-Fault area. If a failure occurs, and a series of important devices (or other nodes for that matter) are located in the Far-From-Fault area, then APA would not generate any alarms for these nodes by default.

If you want NNM to display the alarms for important nodes in the Alarm Browser, you can change this behavior by designating a list of important nodes. This list enables APA to generate the `OV_APA_NODE_UNREACHABLE` and `OV_APA_Connection_UNREACHABLE` alarms for these devices, and display them in the Alarm Browser.

As in NNM 7.01, Far-From-Fault nodes do not generate any alarms and their status goes to Unknown because they are secondary, regardless of an important node filter.

Important Nodes Far From Fault

By default if a node is Far-From-Fault (secondary failure) then no alarms are generated at all. In NNM 7.5, if a Far-From-Fault node matches the Important Node filter specified in `paConfig.xml`, a Node Unreachable alarm is generated that is not suppressed by the ConnectorDown ECS correlation.

Far-From-Fault nodes that are important will have appropriate OV_APA_Node_Unreachable alarms generated for them. However, nothing in the event points to the root cause. Since the notion of Far-From-Fault is derived from Neighbor Analysis, we do not really know that cause of the fault because the node is not near the fault.

Important Nodes in the Fault Area

By default, secondary failures in the Fault Area are only visible by drilling down into the primary failure alarm.

When you identify a node as important and it is in the Fault Area (just before the primary failure in NNM's path), APA generates a Connection Unreachable alarm for the important node. Secondary failure Interface and Address alarms are not visible regardless of whether the node is important or not (except via drill down).

Note: This is a Node-Level feature. APA has no such thing as an Important Interface. The use of isImportantNode is limited to node alarms and connection alarms. It is not used for interface, address, board, or aggregated port alarms.

Configuring APA to Display Alarms from Important Nodes

Use the following procedure to designate these important nodes:

1. Make a backup copy of the following file:
   - *Windows*: `%OV_CONF%\nnmet\topology\filter/MyHostID.xml`
   - *UNIX*: `$OV_CONF/nnmet/topology/filter/MyHostID.xml`
2. As `Administrator` or `root`, edit the `MyHostID.xml` file.
3. Look for the last line containing the following tags:

   ```
   <IPv4><address>0.0.0.0</address></IPv4>
   ```

4. Change the text shown in bold in the previous step to match your needs using the syntax in one of the following examples. In these examples, x, y, and z represent replaceable IP address fields:

   ```
   <IPv4><address>x.y.z.*</address></IPv4>
   ```

   ```
   <IPv4><address>x.y.z.1-99</address></IPv4>
   ```

   You can add as many copies of this line as you need to specify all of your important nodes.

   Make sure to save your changes.

5. `ovstop ovet_poll`.
6. `ovstart ovet_poll`.

The 7.01 `paConfig.xml` file specifies two user configurable parameters, isImportantNodeUpToDown and isImportantNodeDownToUp. NNM 7.5 simplifies this to isImportantNode which will apply to transitions either direction.

APA Configuration Polling

APA Configuration Poll

- Gathers and analyzes detailed information for all interfaces and boards on a node
  - –Interface Details:
  - – ifDescr
  - – ifPhysAddress
  - – ifName
  - – ifAlias
  - –Board Details:
  - – serialNumber
- Scheduled through paConfig.xml or on-demand
- Detects interface renumbering
- Detects board renumbering
- Additions and deletions show up as renumbering

U5089S ©2004 Hewlett-Packard Development Company, L.P. 26

A component within APA , called the Configuration Poller, works with the Polling Engine and Status Analyzer. The Configuration Poller gathers board and interface information from nodes (ifAlias, ifName, ifDescr, PhysAddress etc.). Through this configuration polling, APA is able to detect when interfaces have been renumbered.

The Configuration Poller performs SNMP queries to obtain necessary configuration data. The data is compared with the current Extended Topology database. If there are differences, the Configuration Poller notifies the Status Analyzer to look more closely.

If the interfaces have been renumbered, APA issues an event. Discovery notes in its message that a rediscovery is needed and the user performs a rediscovery.

Sample Scenario: on a Cisco 4507R Router

A router has 2 supervisory boards in it. One is active and the other is in standby mode. If one fails, the other takes over. It turns out that if you cold start the router, Cisco automatically switches from the previously active board to the standby. This causes interface renumbering. The ifAlias values changed as well.

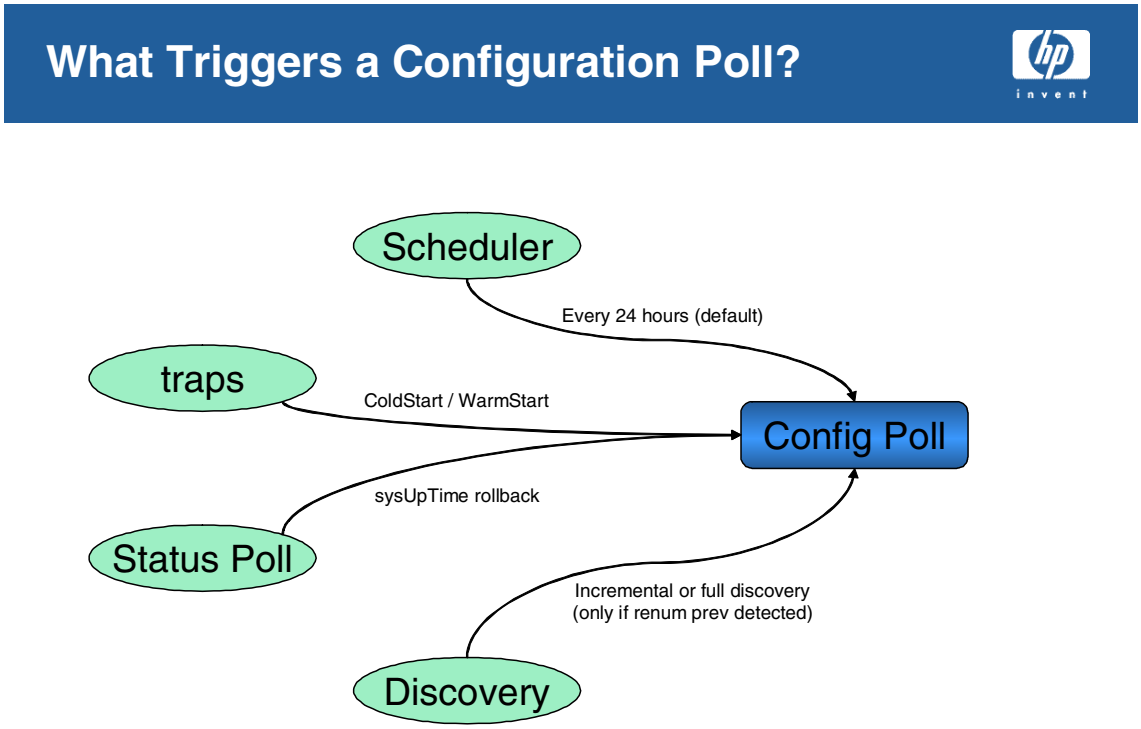
When APA receives the cold start trap, it triggers a configuration poll. During the configuration poll on the device, the Configuration Poller notices that the interface numbers are different, and

informs the Status Analyzer.

The Status Analyzer takes a closer look and determines that an interface renumbering has occurred on the device. APA generates a renumbering event which shows up in the Alarm Browser. The user sees the alarm, and follows the advice to do a new discovery so Extended Topology has an updated view of the devices/interfaces.

The rediscovery initiates a new configuration poll and it no longer detects renumbering for the node (the configuration poll finds the node now matches the newly-discovered Extended Topology database). APA issues a log-only event to cancel the original renumbering event.

What Triggers a Configuration Poll?



Configuring APA Configuration Polling

By default, configuration polling is done once per day. It may also be triggered by certain traps coming into the management station from devices in the network. An APA demand poll also triggers an APA configuration poll (different from a netmon configuration poll, which is not triggered).

You can configure how often configuration polls occur or disable them entirely in the `ConfigPollSettings` section of `paConfig.xml`. You can also specify some devices that should not receive configuration polls. Stop and restart `ovet_poll` for your changes to take effect.

Interface Renumbering

Interface Renumbering

- When can it happen?
  - Reboot of a device
    - Cisco 4507R router with 2 supervisory cards
  - Hot-swap of cards
  - Software interface module dies and new instance created
- Why do we care?
  - Indication that Topology is no longer in sync with reality
  - APA analysis could be suspect

©2004 Hewlett-Packard Development Company, L.P.

The interfaces or boards on a network device could be renumbered in the following administrative scenarios:

- Adding a new board to an existing device
- Moving an existing board from one slot to another in an existing network device
- Power cycling a network device

For accurate root cause analysis, the Extended Topology database needs to match the actual environment.

The NNM correlators access the interface object ID instead of the ifIndex when comparing events. This allows the proper correlation between traps and APA events. The ifIndex referenced in the APA event may no longer be accurate, but the correlation will still occur between the appropriate events.

APA Interface Renumbering Analysis

APA IF Renumbering Analysis

- Occurs during Configuration Poll
- Analysis is the comparison of the details from
  - the Extended Topology database (the state ET is operating with)
  - SNMP Gets (the state of the node right now)
- Any difference means renumbering was detected

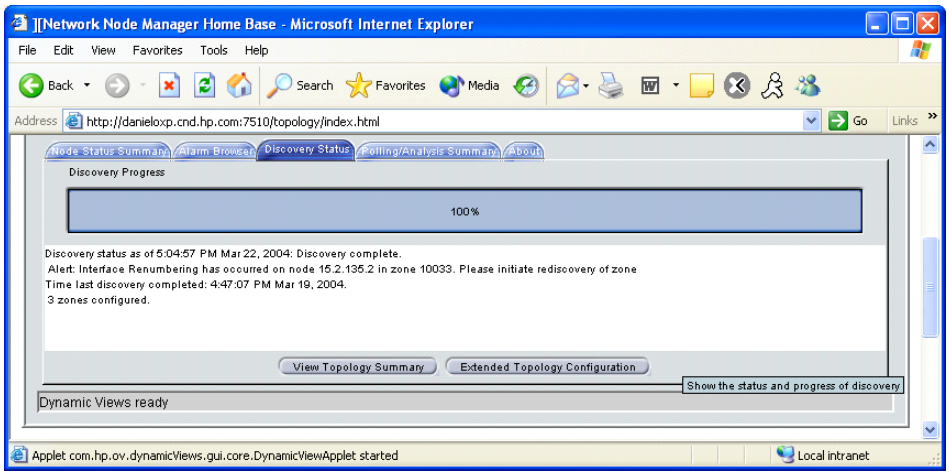
- NOTE: A Configuration Poll is not performed within two minutes of another Configuration Poll on the same node.

©2004 Hewlett-Packard Development Company, L.P.

Interface Renumbering Display

Interface Renumbering Display

- APA does configuration polling
- Issue event if interface renumbering has occurred
  - Alarm in Alarm Browser
  - Discovery updates the ovstatus –v ovet_disco message
- User initiates a new discovery

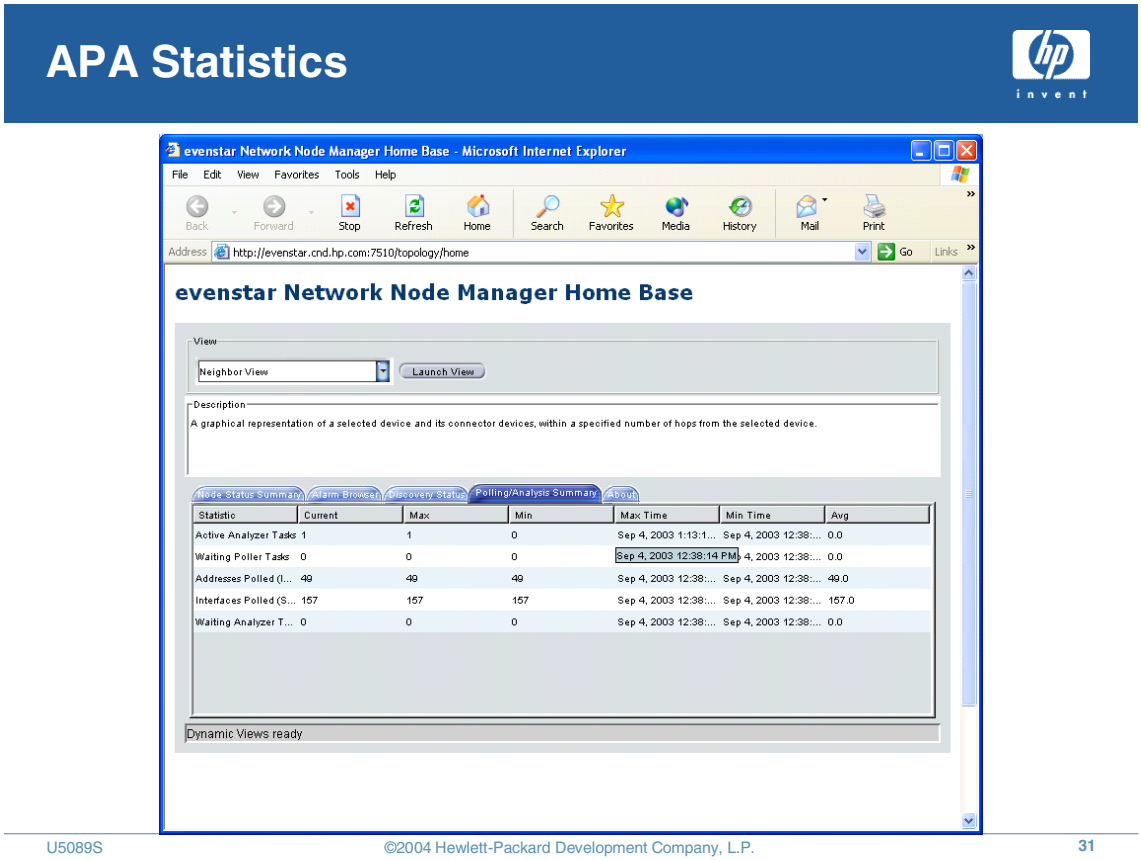


 ©2004 Hewlett-Packard Development Company, L.P.

APA places an alarm in the Alarm Browser indicating that it has detected renumbering on a specific device. Once the rediscovery is completed, this alarm is removed from the browser.

Discovery receives the internal event from APA that renumbering has occurred and rediscovery is needed and updates the “ovstatus -v ovet_disco” message with appropriate details (which zone and which device) to alert you about the need for rediscovery. If multiple renumberings are identified and they are in different zones, you are notified to do a full discovery. (If discovery is currently running and that zone has not been processed yet, the message is suppressed.)

APA Statistics



NNM collects information that shows how APA is performing. From Home Base, select the `Polling/Summary Analysis` tab. The resulting view shows information from the Active Problem Analyzer (and not from NNM's netmon process). APA statistics are collected on five-minute intervals. The statistics are as follows:

- *Active Analyzer Tasks*: This represents the number of polling results that are currently under analysis. This number should trend toward zero when the network is stable. If this number never trends toward zero, you may need to increase the number of threads in the status analyzer thread pool. Only if a poll detects a change in results does something get analyzed. This numbers includes both basic polling analysis, e.g. ping and SNMP time-outs/responses, as well as HSRP analysis results.

  If the number of `Active Analyzer Tasks` begins to steadily increase, then you are experiencing network problems. Look in the Alarm Browser for an alarm that indicates the root cause of your network problem.

- *Waiting Poller Tasks*: This represents the maximum number of polling tasks that were waiting to be completed during the last polling interval. In other words, the poller executes up to the polling thread count configuration number of tasks in parallel. If that number is exceeded, some tasks may need to wait for a bit. This is the maximum number of tasks that had to wait during the last statistics reporting interval. Track the quantity of `Waiting Poller Tasks` that is normal for your environment. If this number begins to increase, it indicates that the APA poller is unable to keep up with the polling load.

To remedy this, review the following list and make adjustments if necessary.

— Make sure NNM and Extended Topology are only monitoring your critical devices.

— Increase the default polling interval. See “Changing the Default Polling Interval” on page 23 for more information. Do not adjust this number too low, as that can create performance problems when large quantities of network problems occur.

— Increase the polling intervals of any of the device classes. See “Changing the Polling Interval by Device Class” on page 24 for more information. Do not adjust this number too low, as that can create performance problems when large quantities of network problems occur.

— Increase the polling engine thread pool size. Do not adjust this number too low, as that can create performance problems when large quantities of network problems occur.

- *Addresses Polled (ICMP)*: This represents the number of addresses that were pinged during the last statistics reporting interval. Track the quantity of `Addresses Polled (ICMP)` that is normal for your environment. This number is an indication of how busy your APA polling engine is. Make sure NNM and Extended Topology are only monitoring your critical devices.
- *Interfaces Polled (SNMP)*: This represents the number of interfaces that were queried for status through SNMP during the last reporting interval. Track the number of `Interfaces Polled (SNMP)` that is normal for your environment.This number is an indicator of how busy your APA polling engine is. Make sure NNM and Extended Topology are only monitoring your critical devices.
- *Waiting Analyzer Tasks*: This represents the number of polling results waiting to be analyzed. When a series of failures occur, this number rises. If this number continues to rise over several polling cycles, it indicates a serious issue in the network. A temporary surge, which then trends toward zero, is normal when a fault or change occurs.
- *HSRP Groups Polled*: This represents the number of HSRP groups that were queried for status during the last reporting interval. Track the `HSRP Groups Polled` number that is normal for your environment. This number is an indication of how busy your APA polling engine is. Make sure NNM and Extended Topology are only monitoring your critical HSRP devices.

Lab Exercises

 ©2004 Hewlett-Packard Development Company, L.P.

Setting APA As Your Default Poller

1. Click on the Polling/Analysis Summary tab of NNM's Home Base. You'll note that all the numbers are 0 in this summary.
2. Change the default poller for your system from `netmon` to APA.

3. Look at the Polling/Analysis Summary again in NNM's Home Base. Within 5 minutes, you should start to see numbers change in this summary.

APA Demand Polling

CAUTION Successful execution of this lab depends on following the directions given by your instructor AT THE TIME those instructions are issued.

Directions

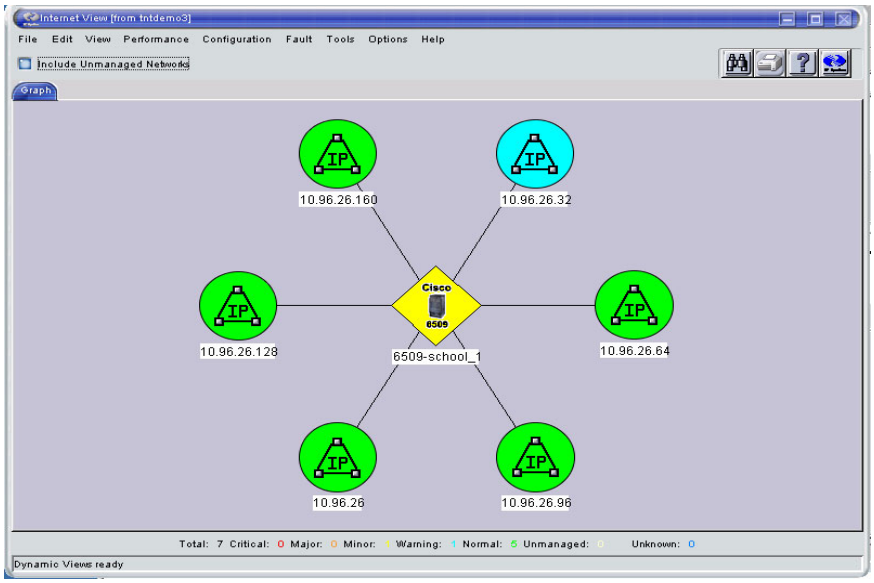
1. Change your working directory to `$OV_CONTRIB/NNM7labs/Lab_deploy`:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_deploy
   ```

2. When the instructor informs you to proceed, execute the setup script:

   ```
   setupLab_deploy.ovpl
   ```

3. When the setup script completes start Home Base. Launch the Internet View.
4. Expand the groups and locate any symbol that may be unmanaged or unknown. If unmanaged, select it, and use the menu `Edit:Manage`.
5. The map should appear as shown.



6. Commence the Extended Topology discovery. Use the Home Base Discovery tab or execute:

   ```
   etrestart.ovpl -verbose
   ```

7. From the NNM Home Base, launch a Node View. Select “All” for the Show Nodes field and “Normal” for the Status >= field, then click the Refresh button. Note the status of the nodes in the resulting view.
8. When the instructor gives you the go-ahead, right click on the 6509-school_1 node. Select “APA Status Poll” from the resulting pop-up menu. Wait for the APA status poll to complete in the resulting APA Demand Poll window.
9. Close the APA Demand Poll window and return to the Node View window.
10. Observe the status of the 6509-school_1 node.
11. When the instructor gives you the go-ahead, repeat the steps to obtain a new status for the 6509-school_1 node.

9 Configuring Extended Topology Discovery of OSPF

Module Objectives

Configuring Extended Topology Discovery of OSPF

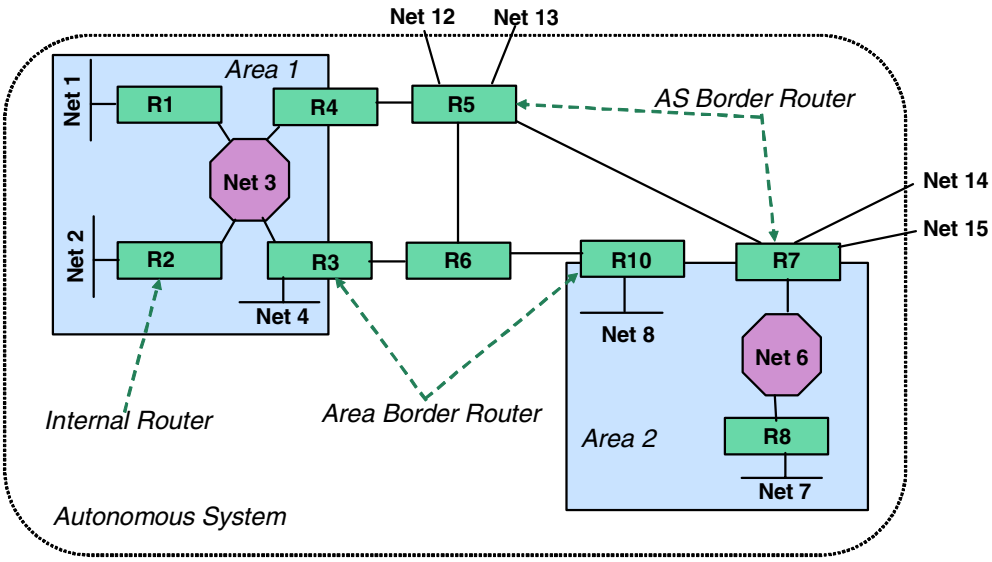
Version C.00
U5089S Module 9 Slides

After completing this module, you will be able to:

- Describe the purpose and use of OSPF.
- Configure Extended Topology for OSPF monitoring.
- Troubleshoot OSPF monitoring.

Open Shortest Path First Background

OSPF Background



©2004 Hewlett-Packard Development Company, L.P.

Open Shortest Path First is a routing protocol that allows routers to collect and share information to build a topology of the network. As a “link-state” routing protocol, it provides more accurate routing and faster response to changes than older “distance-vector” routing such as RIP.

With OSPF, when a router notices a change in its routing table, it multicasts just that change to neighboring routers. This contrasts with RIP, which sends the entire routing table every 30 seconds whether there has been a change or not.

OSPF is intended for interior routing within a collection of networks managed by a single authority (an “autonomous system”). OSPF routers may be connected by LAN or WAN (ATM or frame relay). Although OSPF operates within an autonomous system group, it can send internal route information to other autonomous system groups. Border Gateway Protocol is used for this communication.

Each OSPF autonomous system may be divided into “areas,” where routing is internalized and hidden from other areas. This two-level hierarchical approach reduces the amount of routing information on the network. An area may have a range of addresses, but only a single route is advertised externally.

Routers within an area build a topology database containing link information for the entire area. External links are handled by “area border routers” whose job is to generate a summary of the routes within the area and exchange that information with other area border routers. Area border routers maintain a topology state database for each area to which they belong.

Routers which also know about the rest of the world, beyond the OSPF area, are called Autonomous Border Rotuers.

The group of all area border routers, or the OSPF backbone, is referred to as Area 0. The backbone is responsible for distributing routing information between non-backbone areas. When routing a packet between two non-backbone areas the backbone is used. The path that the packet will travel can be broken up into three contiguous pieces: an intra-area path from the source to an area border router, a backbone path between the source and destination areas, and then another intra-area path to the destination. The algorithm finds the set of such paths that have the smallest cost.

Each OSPF router has a unique router ID, usually the highest IP address of a router's active interface.

When routers discover each other, they create an adjacency relationship in which they create a connection for exchanging routing information. In a LAN environment where many routers are connected, they may designate a lead router to communicate with routers outside the LAN to reduce the amount of traffic.

Classification of Routers

Internal routers: A router with all directly connected networks belonging to the same area. These routers run a single copy of the basic routing algorithm.

Area border routers: A router that attaches to multiple areas. Area border routers run multiple copies of the basic algorithm, one copy for each attached area. Area border routers condense the topological information of their attached areas for distribution to the backbone. The backbone in turn distributes the information to the other areas.

Backbone routers: A router that has an interface to the backbone area. This includes all routers that interface to more than one area (i.e., area border routers). However, backbone routers do not have to be area border routers. Routers with all interfaces connecting to the backbone area are supported.

AS boundary routers: A router that exchanges routing information with routers belonging to other Autonomous Systems. Such a router advertises AS external routing information throughout the Autonomous System. The paths to each AS boundary router are known by every router in the AS. This classification is completely independent of the previous classifications: AS boundary routers may be internal or area border routers, and may or may not participate in the backbone.

References

Cisco Documentation: *http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/ospf.htm*

OSPF Design Guide, *http://www.cisco.com/warp/public/104/1.html*

OSPF version 2 RFC: *http://www.cis.ohio-state.edu/cs/Services/rfc/rfc-text/rfc2178.txt*

IETF OSPF Charter: *http://www.ietf.org/html.charters/ospf-charter.html*

Running OSPF Discovery

Running OSPF Discovery

- Running OSPF discovery results in Extended Topology discovering OSPF information about your network.
- OSPF discovery is based on two OSPF MIBs:
  - RFC 1850 & RFC 1253
  - Routers of interest must support these MIBs and be SNMP-accessible
- Extended Topology discovers which areas your OSPF devices are in and how the areas are related.

- OSPF seed file is located in
  UNIX: **`$OV_CONF/nnmet/Ospf.cfg`**
  Windows: **`%OV_CONF%\nnmet\Ospf.cfg`**
- Run **`ospfdis.ovpl`** to execute discovery.

©2004 Hewlett-Packard Development Company, L.P.

OSPF management requires purchase of the NNM Advanced Routing SPI.

To use the OSPF view, you must first run OSPF discovery. Running OSPF discovery results in Extended Topology discovering OSPF information about your network. OSPF discovery and status is as dynamic as you configure it to be. It is independent of other Extended Topology polling.

OSPF discovery is founded on two MIBs: RFC 1850 and RFC 1253. All routers of interest must support one of these MIBs. Furthermore, they must be discovered by NNM, and be accessible to NNM via SNMP.

During the OSPF discovery, Extended Topology discovers which area OSPF devices are located in, and how these areas relate to one another.

You must manually run OSPF discovery:

1. Edit the file:
   UNIX: `$OV_CONF/nnmet/Ospf.cfg`
   Windows: `%OV_CONF%\nnmet\Ospf.cfg`
   using the following guidelines. There are OSPF configuration instructions included within the `Ospf.cfg` file.
   - Add the IP address of a router being managed by NNM to seed OSPF discovery.
   - You can set the OSPF discovery range using either an OSPF area INCLUDE or

EXCLUDE list, but not both, as shown in the `Ospf.cfg` file.

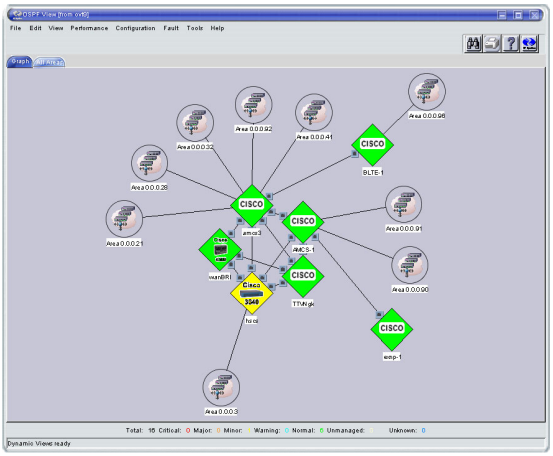
- If you use an INCLUDE list, Extended Topology discovers only the areas in the INCLUDE list.
- If you use an EXCLUDE list, Extended Topology discovers all areas except those on the EXCLUDE list.

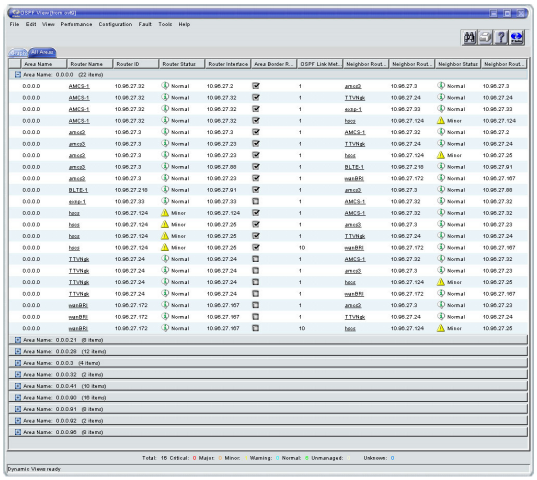
2. Run `ospfdis.ovpl`. You must run `ospfdis.ovpl` each time you modify `Ospf.cfg` for OSPF discovery changes to affect the OSPF view.
3. Once OSPF discovery completes with no errors shown on your screen, check for errors in the `$OV_PRIV_LOG/ospfdis.err` (`%OV_PRIV_LOG%\ospfdis.err` on Windows) file.
4. If there are errors in the `ospfdis.err` file, fix any errors that may impact any devices or areas you are interested in viewing and rerun the `ospfdis.ovpl` command. You do not need to fix all errors as many are natural to discovery at the edges of your OSPF domain.
5. Repeat step 4 until you are satisfied with your OSPF view.

Extended Topology OSPF View

Extended Topology OSPF View

- Graphical representation of an OSPF Area
- Default: Area 0 (0.0.0.0)
- Table of routers, filtered by area, with Router ID, IP addresses, and neighbors





U5089S ©2004 Hewlett-Packard Development Company, L.P. 4

On systems with NNM/Extended Topology installed, the OSPF Backbone view shows you a graphical representation of an OSPF (Open Shortest Path First, a hierarchical routing protocol with two levels) area.

By default, Extended Topology launches your web browser to the OSPF backbone (area 0.0.0.0).

The OSPF table presents information about the *routers* in the OSPF area, including neighbor information. The table shows whether a device is an Area Border Router (ABR) or not. It also shows the neighbors of the device. An ABR connects to both areas. Troubleshooting generally starts here.

From the OSPF Backbone view, you can double-click on an area icon to navigate to a view of that area that shows the routers that participate in that OSPF area.

Use the OSPF Backbone view to visualize all of the OSPF areas and to navigate to a view for a specific area. The OSPF views are useful when troubleshooting connectivity and performance problems in the network. You may want to bookmark specific views for easy access.

With NNM alone, you would have to partition your internet view into containers to show OSPF areas. Now you get more information without the overhead to maintain it.

Common OSPF Errors

If you can walk through the neighbors table, you will get good OSPF discovery data. Otherwise, fix your SNMP access before doing discovery.

Troubleshooting OSPF Discovery

Someone knowledgeable about OSPF and the discovered environment should remedy problems found in the `$OV_PRIV_LOG/ospfdis.err` (`%OV_PRIV_LOG%\ospfdis.err` on Windows) file.

Some common OSPF discovery errors are:

- Device access problems: Device SNMP community strings are missing or incorrect. Find and add the device community names using NNM's SNMP Configuration menu item from the NNM Options menu. This could be the source of numerous `ospfdis.err file` entries.
- Seed device is inaccessible: Make sure the seed device you add to `Ospf.cfg` has been discovered by NNM and is accessible from the system that Extended Topology is running on.
- Discovered device is inaccessible: Make sure the device has been discovered by NNM and is accessible from the system Extended Topology is running on.
- No seed in `Ospf.cfg`: Edit `Ospf.cfg` and add a seed router IP address.

Extended Topology expecting IP addresses: Make sure the seed router IP address is valid.

- Using both INCLUDE and EXCLUDE area lists: Make sure you have not configured both an INCLUDE and an EXCLUDE area list.
- INCLUDE or EXCLUDE area list errors: Make sure you configure the lists correctly in `Ospf.cfg`.

Lab Exercises

- Discover simulated environment
- Configure OSPF discovery
- Rediscover the environment
- Break the environment and rediscover

©2004 Hewlett-Packard Development Company, L.P.

Lab A: OSPF Full Discovery

Objectives:

Gain hands-on experience with OSPF discovery and views.

At the completion of this exercise you will be able to:

- Execute the OSPF discovery
- Properly configure OSPF discovery configuration according to defined limits
- Recognize and troubleshoot common OSPF discovery problems

CAUTION Successful execution of this lab depends on following the directions given by your instructor AT THE TIME those instructions are issued.

Assumption:

- Student has previously executed the `$OV_CONTRIB/NNM7labs/initialsetup.ovpl` script.

Directions

1. `cd $OV_CONTRIB/NNM7labs/Lab_ospf`
2. When the instructor informs you to proceed, execute the OSPF Lab setup script:

   `setupLab_ospf.ovpl`
3. When the setup script completes, start the Extended Topology discovery. From the Home Base Discovery tab, [`Initiate Full Discovery Now`] or execute:

   `etrestart.ovpl -verbose`

NOTE	Although we restart Extended Topology processes, we do not use the Extended Topology level 2 views except for the OSPF view throughout the OSPF lab.

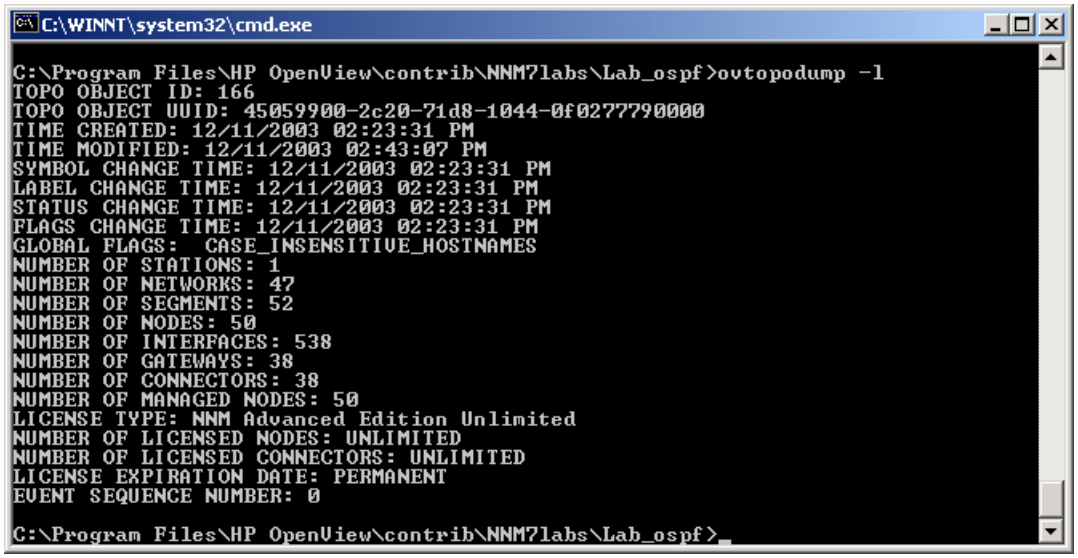
4. Confirm availability of NNM Extended Topology views. Load the `VLANs view` using Home Base.

NOTE	You should see an empty VLAN table. If you get a message indicating: "`NNM Extended Topology is not currently available,`" you must either re-execute `etrestart.ovpl` or wait for it to complete.

5. Confirm the full OSPF topology has been properly loaded into NNM, execute:

   `ovtopodump -l`

   You should see the following information:



6. Edit the OSPF configuration file to confirm correct designation of the OSPF Seed router:

   `vi $OV_CONF/nnmet/Ospf.cfg`

   This file contains the OSPF-specific discovery configuration. Of particular importance is the SEED {IP-ADDRESS} designation (at least one is required) AND the INCLUDE and

EXCLUDE statements.

For LAB A (Full OSPF Discovery), ensure the `SEED IP_ADDRESS =10.96.27.32`.

Next ensure that ALL the `INCLUDE` and `EXCLUDE` lines are commented out (using the "#" character). Save and exit the file editor.

The `Ospf.cfg` file should look similar to:

```
=====================================================Start
#The next line will use the seed 10.96.27.32 for the start of the discovery.
SEED  10.96.27.32
#
# The next line will discover areas 0,1,2,3,4,7,9
# INCLUDE 0.0.0.0 0.0.0.1 0.0.0.2 0.0.0.3 0.0.0.4 0.0.0.7 0.0.0.9
#
# The next line will exclude areas 99 and 100 from the discovery.
# EXCLUDE 0.0.0.99 0.0.0.100
=======================================================END
```

7. Execute the OSPF discovery:

   `ospfdis.ovpl`

   You will see the following:

   ```
   #> SEED is 10.96.27.32
   INCLUDE list is
   EXCLUDE list is
   ```

8. While the `ospfdis.ovpl` discovery process is still running, examine the contents of the OSPF data base folder:

   ```
   #> ll $OV_DB/ospf
   total 0
   -rw-rw-r--   1 root       sys              0 Mar 18 09:49 ospfdis.lock
   -rw-rw-r--   1 root       sys              0 Mar 18 09:49 ospfdis.tmp
   ```

   The `ospfdis.lock` and `ospfdis.tmp` files indicate an active `ospfdis.ovpl` process. No final OSPF data is available for the OSPF view.

9. Confirm the discovery process completed correctly, showing 46 paths and 1 error, indicated by the output in the shell:

   ```
   #> OSPF discovery results, 46 paths found, 1 errors found.
   See $OV_PRIV_LOG/ospfdis.err for any errors reported.
   ```

NOTE You should have 50 paths found. Please inform the instructor if you do not! (You may have to re-run the `setupLab_ospf.ovpl` for Lab1, and/or re-run the `nmdemandpoll` tool and/or re-execute `$OV_BIN/etrestart.ovpl -verbose`.

10. View the contents of `$OV_PRIV_LOG/ospfdis.err` file. We will return to this file later.

11. Once the OSPF discovery process completes, view the contents of the OSPF database folder:

    ```
    #> ll $OV_DB/ospf
    total 14
    -rw-rw-r--   1 root       sys           6572 Mar 18 09:51 ospfdis.data
    ```

12. The `ospfdis.data` file contains the OSPF the connectivity data/references database used to generate the OSPF view. View the contents of this file (`more`, `cat` ,etc.)
    `more $OV_DB/ospf/ospfdis.data`

    Contents of `$OV_DB/ospf/ospfdis.data`

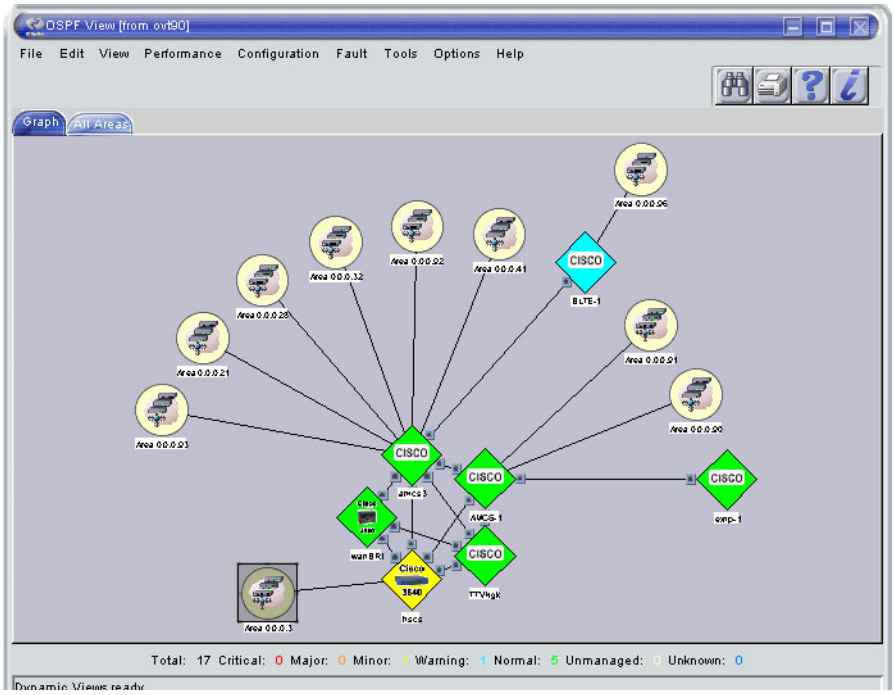
    ```
    =====================================================Start
    ```

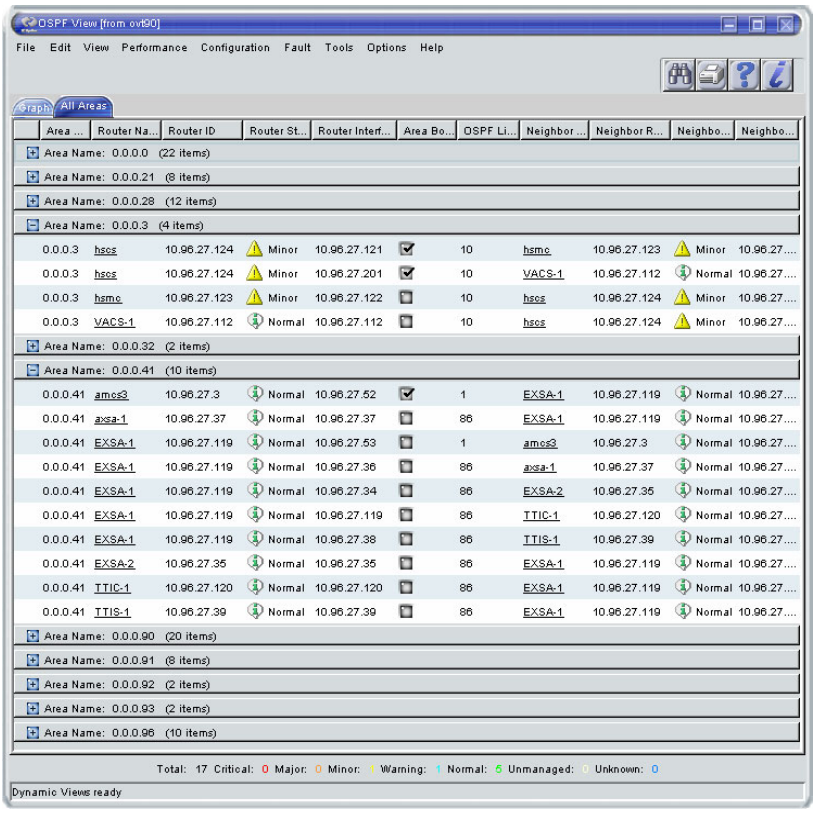
```
#File format
#nodeId sysname routerId abrvalue area ip nodeId sysname routerId abrvalue ip
#abrvalue is 0 for normal, 1 for area border router
#2 for autonomous border router, 3 for both 1 and 2
#
#area of seed 0.0.0.0
#seed is 10.96.27.32
#
524 AMCS-1 10.96.27.32 3 0.0.0.0 10.96.27.2 516 amcs3 10.96.27.3 3 10.96.27.3
524 AMCS-1 10.96.27.32 3 0.0.0.0 10.96.27.32 522 TTVNgk 10.96.27.24 0 10.96.27.24
…continues
=====================================================End
```

NOTE The `nodeId` reference is to the NNM object database (`ovobjprint -o nodeId`).

13. View the fully discovered OSPF topology using Home Base to launch the OSPF view.

    You should see the backbone area 0.0.0.0 with 10 area symbols as shown:





14. Save this view. Select `File:Save`. Name the view `lab1afullOSPF.xml`.
15. Navigate the views and drill into each area. Note on paper the Area Names (such as 0.0.0.4, 0.0.94, etc). Take note of the Area Border routers. There are 4: AMCS-1, amcs3, BLTE-1, and hscs.

    Verify this.

Summary

Significant points about the lab thus far:

1. This is the full `OSPF` lab network that has been simulated from an actual customer network.
2. The `$OV_CONF/nnmet/Ospf.cfg` file was configured for the proper SEED IP-ADDRESS. YOU MUST IDENTIFY THE PROPER OSPF SEED ADDRESS FOR YOUR NETWORK.
3. We omitted any discovery restrictions by NOT using either the `INCLUDE` OR the `EXCLUDE` statement.
4. OSPF discovered data is contained in the text file `$OV_DB/ospf/ospfdis.data`.

Lab A: Full OSPF Discovery Questions:

1. What is the significance of the OSPF BACKBONE Designation?
2. How is the Backbone Area Designated?
3. Area 0 contains seven routers (AMCS-1, amcs3, BLTE-1, exnp-1, hscs, TTVNgk, and wanBRI). Area 96 contains 5 routers: BLTE-1, SRDO-1, SDFL-1, VATE-1 and SDBA-1.

   What router(s) does SDBA-1 know exists?

What routers does BLTE-1 know exists?

What is the significance of “known routers” in this context?

4. List the routers that are (hint, read the `ospfdis.data` file):

   a. Area Border Routers:

   b. Autonomous Border Routers:

   c. Both Autonomous and Area Border Routers:

You may proceed to the next lab at your own pace.

Lab B: OSPF Limited Discovery

Objectives:

Gain hands-on experience with OSPF Limited discovery and views.

At the completion of this module you will be able to:

- Execute a limited OSPF discovery using INCLUDE and EXCLUDE statements.

Assumption:

- Student has previously successfully completed the Full OSPF Discovery Lab.

Directions

1. Edit the `$OV_CONF/nnmet/Ospfg.cfg` file and use either `INCLUDE` or `EXCLUDE` statement to limit the OSPF discovery to areas *0, 3, 90, 91, 92 and 96*:

   ```
   vi $OV_CONF/nnmet/Ospf.cfg
   ```

   For LAB B (Limited OSPF Discovery), ensure the `SEED IP_ADDRESS =10.96.27.32` and that you use ONE of the `INCLUDE` OR `EXCLUDE` statements. Save and exit the file editor.

   The `Ospf.cfg` file should look similar to:

   ```
   ====================================================Start
   #The next line will use the seed 10.96.27.32 for the start of the discovery.
   SEED  10.96.27.32
   #
   #The next line will discover areas 0,3,90,91,92,96
   INCLUDE 0.0.0.3 0.0.0.90 0.0.0.91 0.0.0.92 0.0.0.96
   #
   # The next line will exclude areas 99 and 100 from the discovery.
   # EXCLUDE 0.0.0.99 0.0.0.100
   ====================================================END
   ```

2. Save the current `ospfdis.data` file:

   ```
   cd $OV_DB/ospf
   ```

   Copy the current `ospfdis.data` file to `ospfdis.data.lab1`.

3. Execute the limited OSPF discovery:

   ```
   ospfdis.ovpl
   ```

   You will see the following:

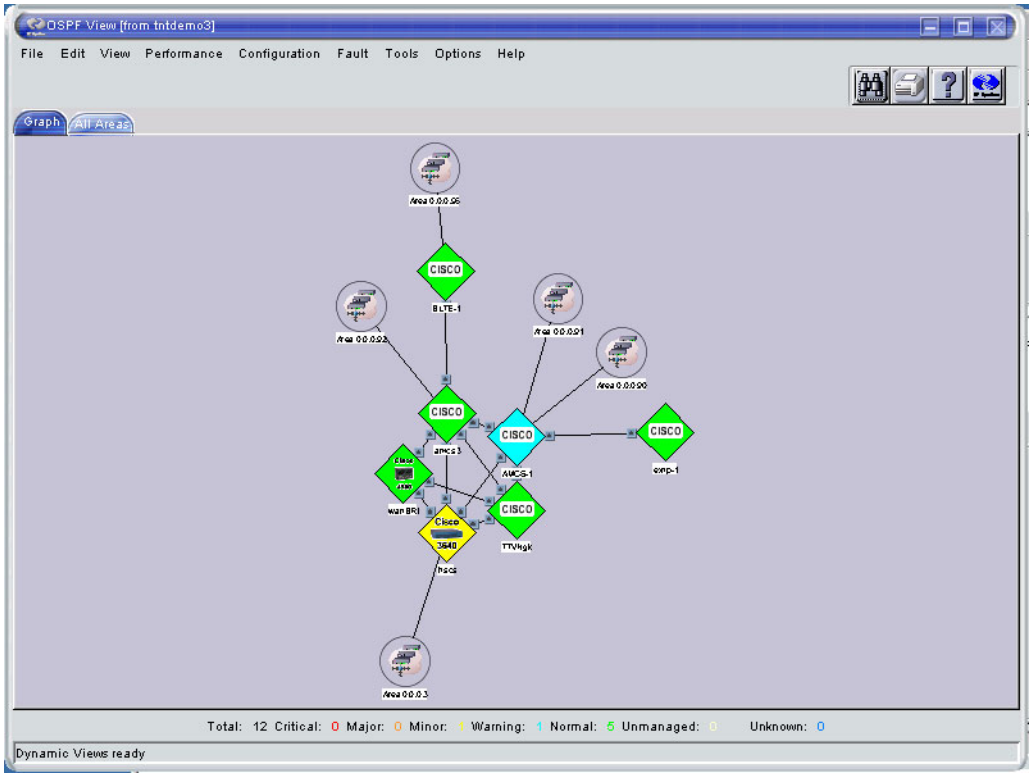
   ```
   #> ospfdis.ovpl
   SEED is 10.96.27.32
   INCLUDE list is 0.0.0.3 0.0.0.90 0.0.0.91 0.0.0.92 0.0.0.96
   # EXCLUDE list is
   OSPF discovery results, 33 paths found, 1 errors found.
   See /var/opt/OV/log/ospfdis.err for any errors reported.
   ```

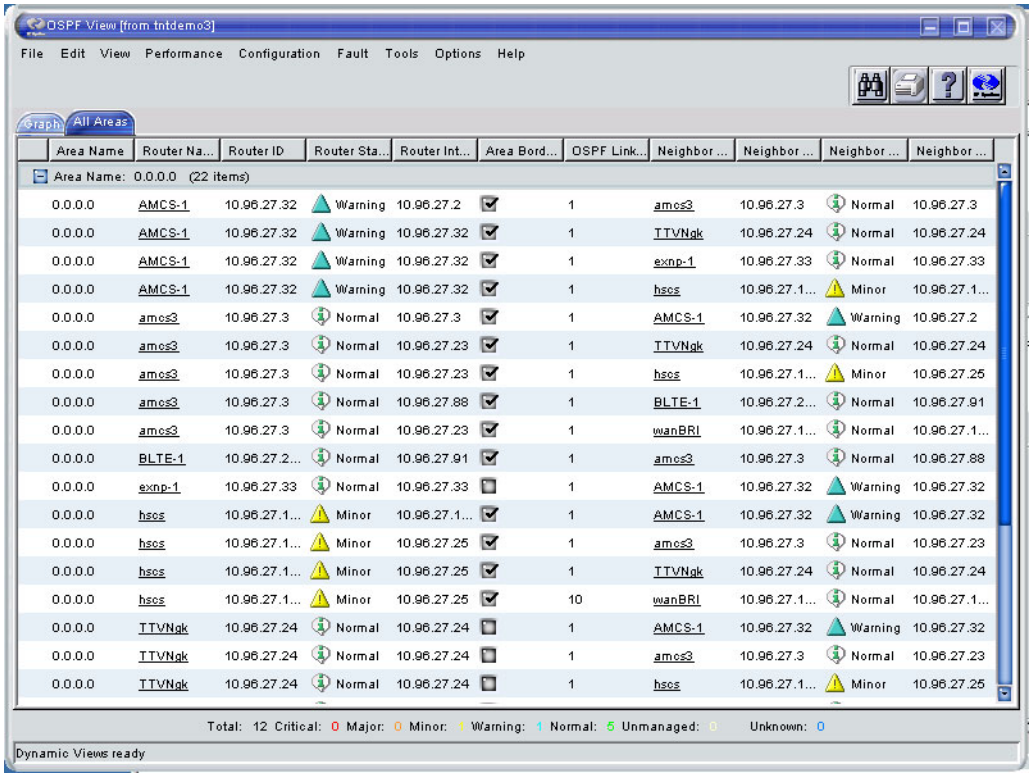
4. List the contents of the OSPF database directory:
   `ll $OV_DB/ospf`
   <=># ll $OV_DB/ospf
   total 24
   -rw-rw-r-- 1 root sys 4262 Mar 18 14:20 ospfdis.data
   -rw-rw-r-- 1 root sys 6572 Mar 18 09:51 ospfdis.old

   The most recent discovery is `ospfdis.data`, the one just prior to the current is the

`ospfdis.old`

5. Compare the results between the 1st and 2nd (3rd etc) OSPF discovery files in `$OV_DB/ospf`. BE CERTAIN TO COPY THE `ospfdis.data` files each time you run the OSPF discovery. (You will need these copies to compare the results) and label them Lab-1Aospfull-1, Lab-1Bospflimited1, etc…).
6. Run at least two separate OSPF discoveries using an `INCLUDE` statement then an `EXCLUDE` statement (save the results!). Execute the OSPF discovery, save the data file, examine the view and then reconfigure the `Ospfg.cfg` file for `EXCLUDE` and re-run `ospfdis.cfg`.
7. Compare the current `ospfdis.data` file with the one just prior (Limited Discovery compared to the Full Discovery).
8. The results of the limited discovery should show something similar to:





LAB B Limited OSPF Discovery Questions:

1. How can you limit OSPF Discovery?
2. When would you use the INCLUDE statement?
3. When would you use the EXCLUDE statement?
4. Must you INCLUDE area 0.0.0.0?
5. What are the results when you do not explicitly INCLUDE the backbone area?
6. What is the result when you EXPLICITLY EXCLUDE the backbone (area 0.0.0.0)?
7. Compare the results between the FULL OSPF discovery and the Limited (INCLUDE) OSPF Discovery.

   What differences are visible in the TABLE for OSPF Area 0 presented in the OSPF View? Explain:

STOP

Please wait for the instructor before proceeding.

(Optional) Lab C: OSPF Partial Discovery (incomplete SNMP access)

Objectives:

Explore the results of running the OSPF discovery when a device fails to respond due to improper SNMP community names and/or device is otherwise not available.

At the completion of this module you will be able to:

- Understand the importance and impact of SNMP Community Strings regarding OSPF discovery.

CAUTION Successful execution of this lab depends on following the directions given by your instructor AT THE TIME those instructions are issued.

Do not proceed until your Instructor has said to continue.

Assumptions:

- Student has previously successfully completed the Limited OSPF Discovery Lab.

Directions

1. Change your working directory to `$OV_CONTRIB/NNM7labs/Lab_ospf`:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_ospf
   ```

2. Replace the modified `Ospf.cfg` file with the original:

   ```
   cp –p $OV_CONTRIB/NNM7labs/Lab_ospf/configFiles/Ospf.cfg
   $OV_CONF/nnmet/Ospf.cfg
   ```

3. When the instructor has re-configured the MIMIC Simulation Server and says to continue, execute a new OSPF discovery:

   ```
   ospfdis.ovpl
   ```

   You will see the following:

   ```
   #> ospfdis.ovpl
   SEED is 10.96.27.32
   INCLUDE list is
   EXCLUDE list is
   OSPF discovery results, 20 paths found, 4 errors found.
   See $OV_PRIV_LOG/ospfdis.err for any errors reported.
   ```

   If you do not get these results, notify the instructor for assistance.

4. View error log file. What is the root cause of the error? HINT: run `etrestart.ovpl`.

5. View the resulting OSPF dynamic view. [Tip: if you kept the prior web browser active, just click reload | refresh].

   What is different?

   What device(s) and/or areas is(are) no longer available?

   What OSPF area(s) is(are) missing?

6. How can you tell by reading the `$OV_BIN/ospf/ospfdis.data` file?
   Hint: notice there are two files:

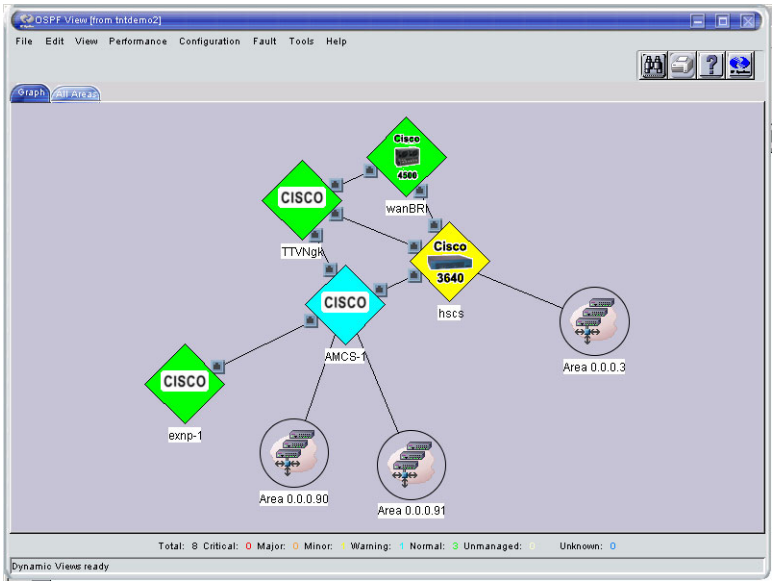
   ```
   #> ll $OV_DB/ospf
   ```

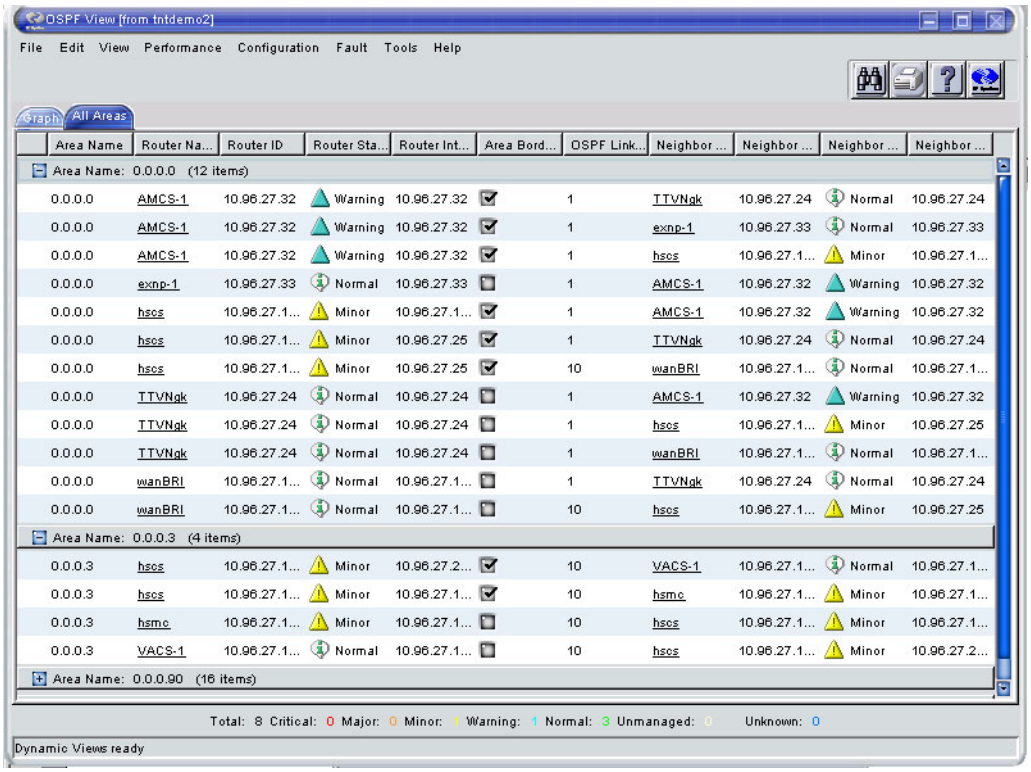
```
total 16
-rw-r--r--   1 root       sys            2899 Feb  6 16:14 ospfdis.data <<latest discovery
-rw-r--r--   1 root       sys            4260 Feb  6 15:51 ospfdis.old  <<previous discovery
```

Can you use the contents to help discover what the differences are?

7. The expected results for the OSPF view are shown:

OSPF Partial Discovery (snmp)





Summary

NNM Extended Topology OSPF discovery relies on a “well known” SEED Address and snmp access to neighboring devices within the OSPF environment for proper and complete discovery. When one or more devices fail to respond, errors are logged and the resulting view is likely incomplete.

STOP

Please wait for the instructor before proceeding.

(Optional) Lab D: OSPF Partial Discovery (Unmanaged Device)

Objectives:

Explore the results of running the OSPF discovery when a device, which should be included in the discovery process, is not managed by NNM or is otherwise not in topology.

At the completion of this module you will be able to:

- Understand the importance and impact of having all relevant devices in the NNM topology and in the "managed" state.

CAUTION Successful execution of this lab depends on following the directions given by your instructor AT THE TIME those instructions are issued.

Do not proceed until your Instructor has said to continue.

Assumptions:

- Student has previously successfully completed the Partial (no snmp access) OSPF Discovery Lab.

Directions

1. Change your working directory to `$OV_CONTRIB/NNM7labs/Lab_ospf`:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_ospf
   ```

2. Using either an `ovw` map or the Internet view, locate and UNMANAGE the device amcs3.
3. When the instructor has re-configured the MIMIC Simulation Server and given the "go-ahead", execute a new OSPF discovery:

   ```
   ospfdis.ovpl
   ```

   You will see the following:

   ```
   #> ospfdis.ovpl
   SEED is 10.96.27.32
   INCLUDE list is
   EXCLUDE list is
   OSPF discovery results, 20 paths found, 0 errors found.
   See $OV_PRIV_LOG/ospfdis.err for any errors reported.
   ```

   If you do not get these results, notify the instructor for assistance.
4. What does the error log show?
5. View the resulting OSPF dynamic view. [Tip: if you kept the prior web browser active, just click reload or refresh.]

   What is different?

   What device(s) and/or areas is(are) no longer available?

   What OSPF area(s) is(are) missing?
6. How can you tell by reading the `$OV_DB/ospf/ospfdis.data` file?

   Hint: notice there are two files:

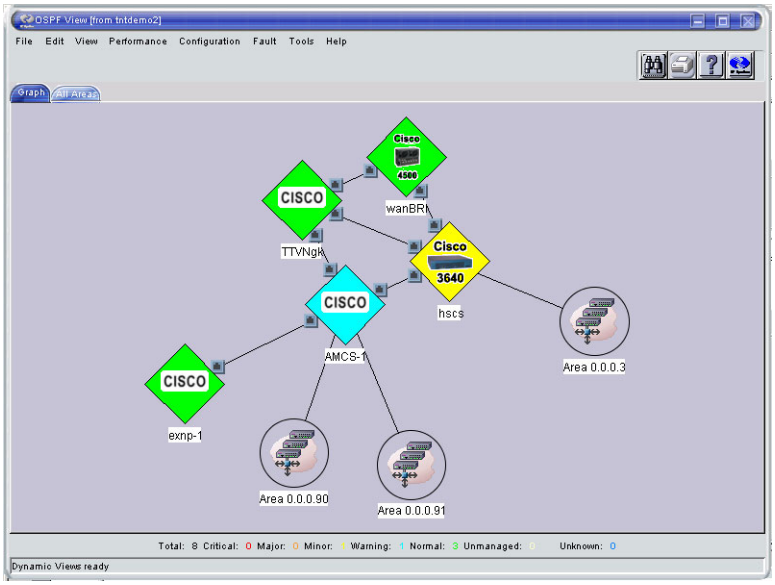
   ```
   #> ll $OV_DB/ospf
   total 16
   -rw-rw-r--   1 root       sys           3279 Mar 18 16:58 ospfdis.data
   ```

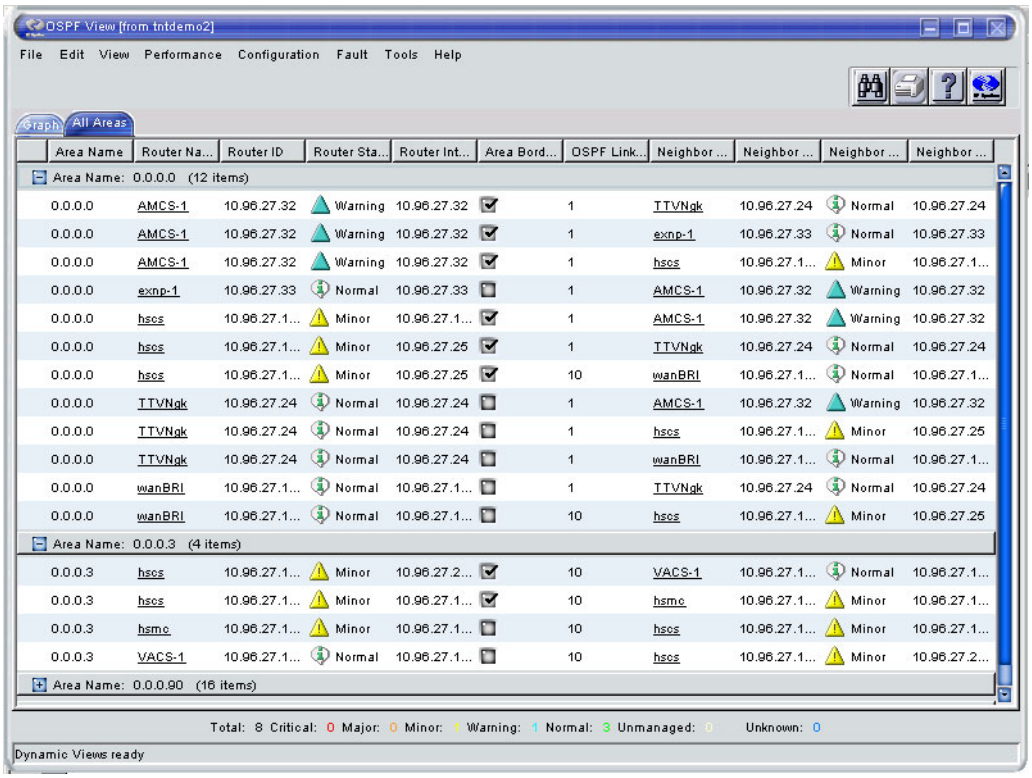
```
-rw-rw-r--   1 root       sys            2900 Mar 18 16:12 ospfdis.old
```

Can you use the contents to help discover what the differences are?

The expected results for the OSPF view are shown:

OSPF View unmanaged device



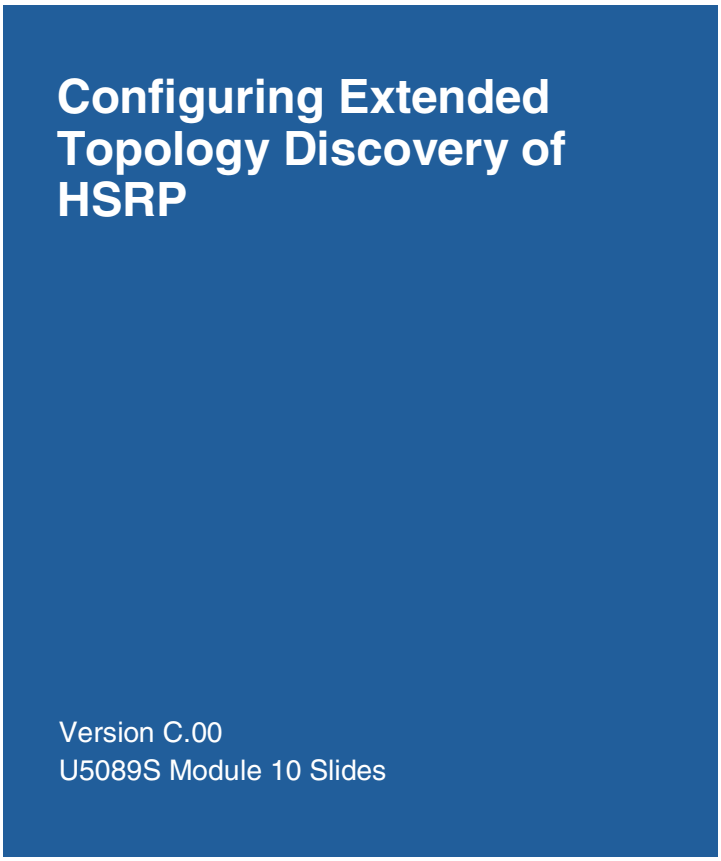


Summary

Devices that are not reachable (down, no snmp response, etc) yet managed in topology will result in discovery errors. Devices that are not managed (or not in topology) will result in OSPF discovery completing without errors but perhaps without the desired result.

10 Configuring Extended Topology Discovery of HSRP

Module Objectives



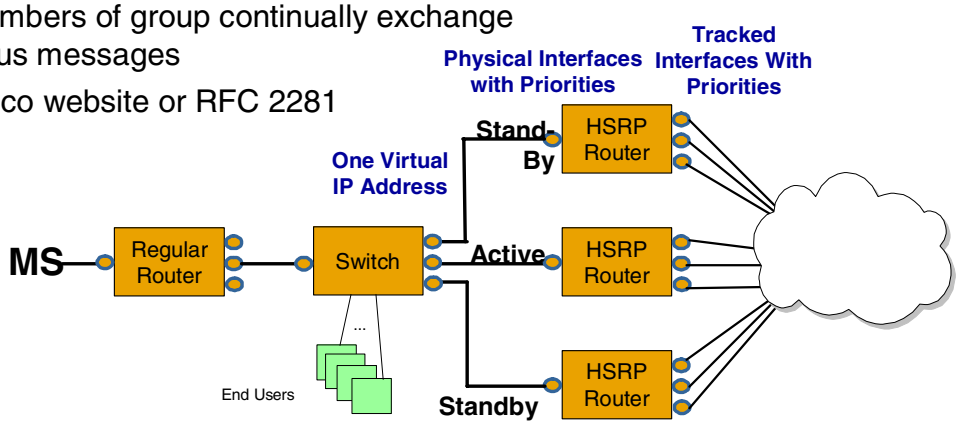
After completing this module, you will be able to:

- Describe the operation of Hot Standby Routing Protocol.
- Describe HSRP failover and the resulting events.
- Configure Extended Topology for HSRP monitoring.
- Display and interpret HSRP views.

HSRP Background Information

Hot Standby Router Protocol

- Network redundancy for IP networks
- User traffic immediately and transparently recovers from first hop failures in network edge devices and access circuits
- Two or more routers share an IP address and a MAC address to act as a single virtual router
- Members of group continually exchange status messages
- Cisco website or RFC 2281



U5089S ©2004 Hewlett-Packard Development Company, L.P. 2

One way to achieve near-100 percent network uptime is to use the Hot Standby Routing Protocol (HSRP), which provides network redundancy for IP networks, ensuring that user traffic immediately and transparently recovers from first hop failures in network edge devices or access circuits.

By sharing an IP address and a MAC (Layer 2) address, two or more routers can act as a single "virtual" router. The members of the virtual router group continually exchange status messages. This way, one router can assume the routing responsibility of another, should it go out of commission for either planned or unplanned reasons. Hosts continue to forward IP packets to a consistent IP and MAC address, and the changeover of devices doing the routing is transparent.

To learn more about HSRP, refer to documents on the Cisco website and RFC 2281.

HSRP is a protocol layered on top of the UDP layer in the network stack. The protocol is implemented in Cisco routers or route switch modules (RSM) to provide failover and/or load balancing for layer 3 routing.

Extended Topology allows you to monitor the HSRP statuses of various HSRP groups, and to track the active and standby (if any) interfaces in the various HSRP groups.

Definitions

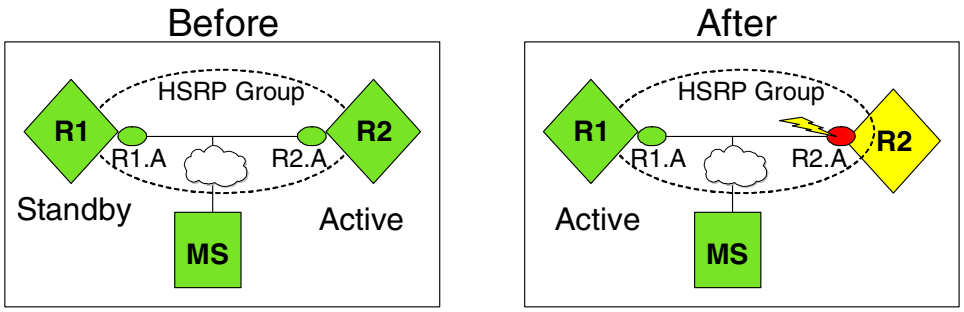
HSRP status or HSRP group status This refers to the status computed by Extended Topology based on the NNM statuses of the various interfaces in the group

HSRP state This refers to the HSRP state value in the HSRP finite state machine that can be assumed by an interface.

HSRP state transition This refers to an HSRP interface leaving one HSRP state and moving to another. Typically this does not happen in one interface but in more than one interface in the group.

HSRP Failover

HSRP Failover Scenarios



- OV_HSRP_No_Standby for the group
- OV_HSRP_FailOver for R1
- OV_APA_IF_Down for R2.A. Status propagates for node.
- Router R1 may generate a HSRP State transition trap

©2004 Hewlett-Packard Development Company, L.P.

Various HSRP topologies exist today. To complicate the matter, they can be created with either two or more than two routers. Also, an HSRP interface can participate in multiple groups for load balancing. The most typical HSRP situation involves only two routers connected via a variety of topological layouts. Each topological layout can have a number of fault conditions associated with it. Elaborating all failure conditions in all topological layouts is an extremely tedious task. Here we take two HSRP routers, and connect them in different topological layouts, and take one or two fault conditions in those layouts as possible scenarios. This is representative of almost all possible scenarios that need to be handled.

The HSRP enabled router interfaces are in the same LAN segment, or connected point to point without any intervening devices between them.

R2.A was active and R1.A was standby.

R2.A failed.

After R2.A failed R1.A is the active interface, and there is no standby. This scenario should generate an OV_HSRP_No_Standby event as the reported HSRP group has no standby interface.

HSRP group status is Minor or Marginal (yellow). The HSRP Group has an interface in the Active state, but there is no interface in the HSRP group which is in the Standby state. This HSRP group is providing routing functionality, but there is no standby router available.

APA generates `OV_HSRP_FailOver` for R1.A. To investigate this alarm, open an HSRP Group view

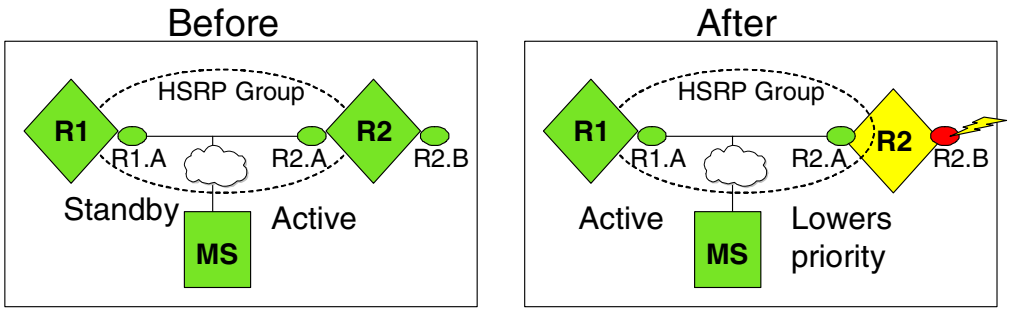
by selecting the `OV_HSRP_FailOver` alarm in the Alarm Browser and selecting the `Actions:Views-> HSRP View` menu. The HSRP Group Detail view shows you that interface R1.A is now active and interface R2.A is down. You need to troubleshoot interface R2.A.

APA generates `OV_APA_IF_Down` for R2.A. The status propagates for the node.

Router R1 may generate a HSRP State transition trap.

HSRP Failover

HSRP Failover Scenarios



- R2.A is tracking R2.B
- OV_HSRP_FailOver for R1
- OV_APA_IF_Down for R2.B. Status propagates for node.
- Router R1 may generate a HSRP State transition trap

©2004 Hewlett-Packard Development Company, L.P.

The HSRP enabled router interfaces are in the same LAN segment, or connected point to point without any intervening devices between them.

R2.A is tracking R2.B. APA monitors tracked interfaces as configured on the routers contained in the HSRP group.

R2.A was active and R1.A was standby.

R2.B failed.

After R2.B failed R2.A lowers its priority, and forces R1.A to become active.

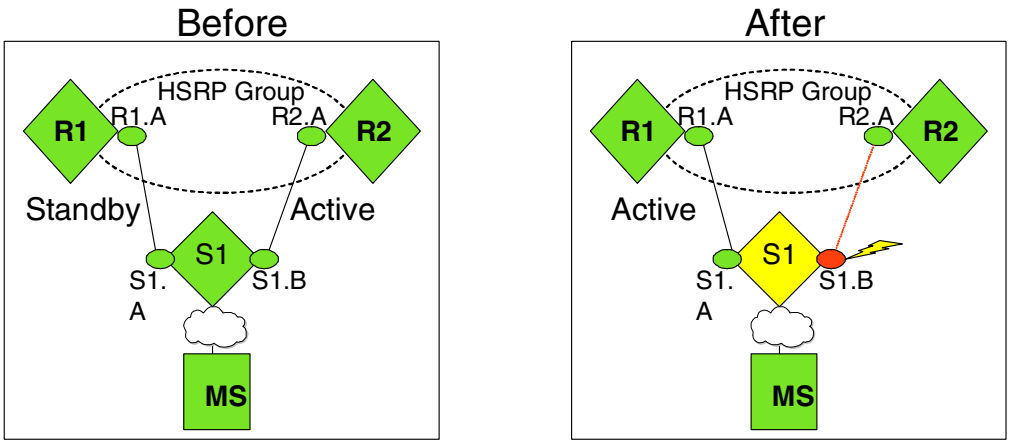
APA generates `OV_HSRP_FailOver` for R1. The HSRP view displays the new priority value and the interface state changes. It also shows the status of the tracked interface.

APA generates `OV_APA_IF_Down` for R2.B.

Router R1 may generate an HSRP State transition trap.

HSRP Failover

HSRP Failover Scenarios (Contd.)



- OV_HSRP_FailOver for R1
- OV_IF_Down for S1.B (from netmon)
- Router R1 may generate a HSRP State transition trap.

U5089S ©2004 Hewlett-Packard Development Company, L.P. 5

The HSRP enabled router interfaces are in the same broadcast domain but are physically connected via switch.

R2.A was the active and R1.A was the standby.

If S1.B or R2.A fails, HSRP fails over, and R1.A is the active.

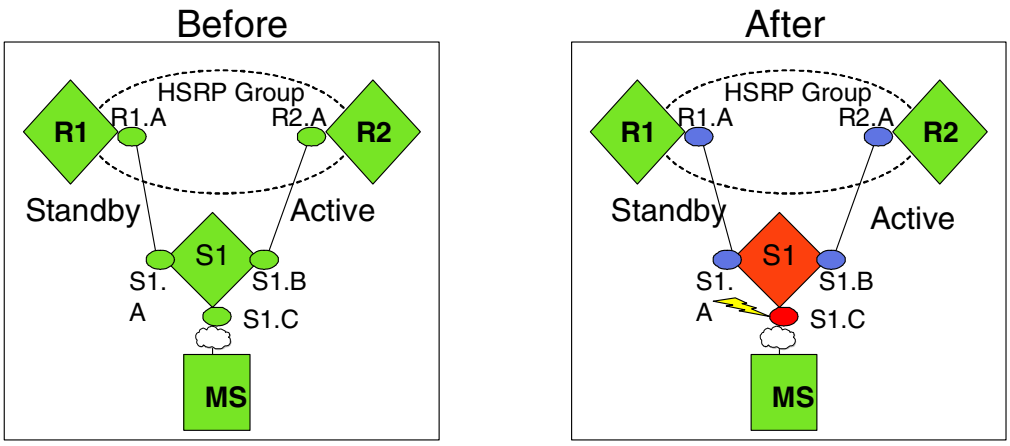
APA generates an `OV_HSRP_FailOver` event. APA generates an `OV_HSRP_No_Standby` event: The reported HSRP group has no standby interface.

`netmon` generates `OV_IF_Unknown` for R2.A and appropriate OV_Node events. `netmon` also causes `ovtopmd` to generate an `OV_IF_Down` for S1.B.

Router R1 may generate a HSRP State transition trap.

HSRP Failover

HSRP Failover Scenarios (Contd.)



- OV_IF_Unknown for R2.A, R1.A
- OV_IF_Down for S1.C
- This scenario highlights the necessity to handle OV_IF_Unknown events differently from the previous scenario

©2004 Hewlett-Packard Development Company, L.P.

The HSRP enabled router interfaces are in the same broadcast domain but are physically connected via switch.

R2.A was the active and R1.A was the standby.

If S1.C fails, HSRP is not impacted.

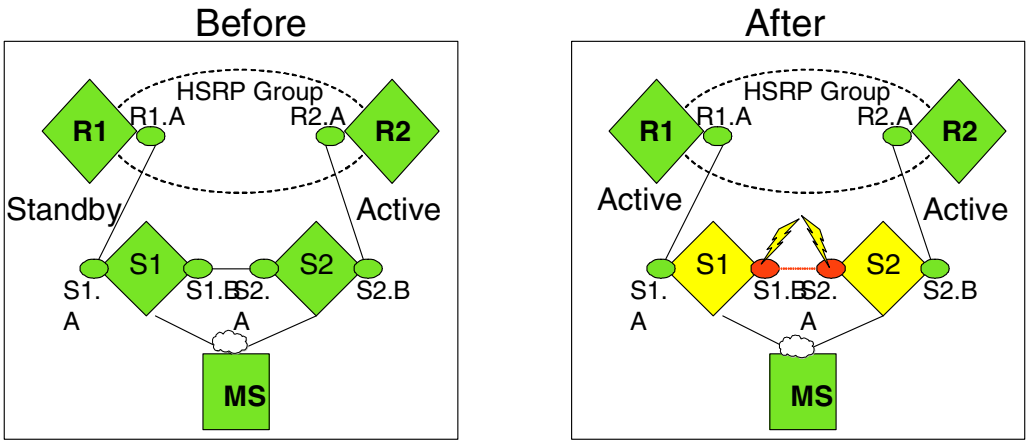
`netmon` generates `OV_IF_Unknown` for R2.A, R1.A, and appropriate OV_Node events for them. `netmon` also causes `ovtopmd` to generate an `OV_IF_Down` for S1.C.

APA has no access to the HSRP group and cannot evaluate its state.

This scenario highlights the necessity to handle `OV_IF_Unknown` events differently from the previous scenario.

HSRP Failover

HSRP Failover Scenarios (Contd.)



- If S1.A and/or S2.A fails, HSRP will be negatively impacted. Both R1.A and R2.A may become active simultaneously!
- OV_IF_Down for S1.A and S2.A

©2004 Hewlett-Packard Development Company, L.P.

The HSRP enabled router interfaces are in the same broadcast domain but are physically connected via two switches.

R2.A was the active and R1.A was the standby.

If S1.A and/or S2.A fails, HSRP is negatively impacted. Both R1.A and R2.A may become active simultaneously! This scenario generates an OV_HSRP_Multiple_Active: The reported HSRP group is in an abnormal condition as there are multiple active interfaces.

`netmon` generates OV_IF_Down for S1.A and S2.A.

Collect and Display HSRP Router Information

Collect and Display HSRP Router Info

- Extended Topology collects HSRP info from devices that support HSRP
  - cisco-hsrp MIB
  - cisco-hsrp-ext MIB
  - No discovery configuration required
- Info is stored in Extended Topology's database
- HSRP view displays HSRP groups
  - Access from Alarm Browser
  - Access from Home Base
- HSRP status monitored by APA and Composer

©2004 Hewlett-Packard Development Company, L.P.

Extended Topology notifies you of state and status changes to an HSRP device. The status change is an indicator of the 'health' of the HSRP virtual router, and state change is an indicator of the state of the HSRP state machine changes.

APA polls HSRP devices only in non-overlapping IP (normal) environments.

HSRP polling in the polling engine provides accurate polling of HSRP devices. The polling engine polls the real state of the HSRP interfaces via SNMP.

Polling Engine

The polling engine polls each HSRP Group defined in the topology. For each HSRP Group the topology stores a list of HSRP addresses that participate in the group. Each HSRP address has a state associated with it that corresponds to the state in the HSRP MIB, `cHsrpGrpStandbyState`.

For each HSRP address, the polling engine reads the `cHsrpGrpStandbyState` and compares it to the state in the topology.

If the state changes, the polling engine records the new state and sends the new state to the Status Analyzer. The polling of the HSRP addresses is done together so the status sent to the Status Analyzer contains all interfaces that have changed.

HSRP Status Analyzer

The Status Analyzer receives status from the polling engine, which is a list of interfaces in an HSRP group that have changed. The status analyzer initially looks at this list to see if any interfaces are in transient states. The transient states are Initial, Learn and Speak and indicate that the poller may have polled the HSRP group while an HSRP reconfiguration was occurring. If there is an interface in a transient state then the Status Analyzer sends a request to the polling engine to re-poll the HSRP group. The Status Analyzer then uses the new list of interface states from this request.

The HSRP address objects in the topology are updated with the new states from the polling engine. After the states are updated, the status analyzer sets the state of the HSRP Group object in the topology based on the new states of the HSRP address objects.

The status analyzer sends HSRP events after the appropriate state has been set.

HSRP State

HSRP state consists of the HSRP group state and the HSRP interface state (the state of the interface participating in the HSRP group).

HSRP Interface State

The NNM Advanced Polling engine reads the actual SNMP state from the HSRP MIB variable and updates the state of the HSRP interface in the database to be one of the following:

initial(1),

learn(2),

listen(3),

speak(4),

standby(5),

active(6)

HSRP Group State

The following list describes the states for an HSRP group that the SA will set.

1. Normal – All interfaces in the HSRP group are functional.
2. Critical – The HSRP group is not operational at all. This would indicate that all interfaces in the group are down.
3. Warning – One or more interfaces in the HSRP group are down but there is still a functional active and standby interface.
4. Marginal – The HSRP group still has an active interface but there is no functional standby interface.
5. Unknown – This indicates that the status of the HSRP group is unknown. This could occur

because the interfaces in the group are unreachable due to a network failure up stream.

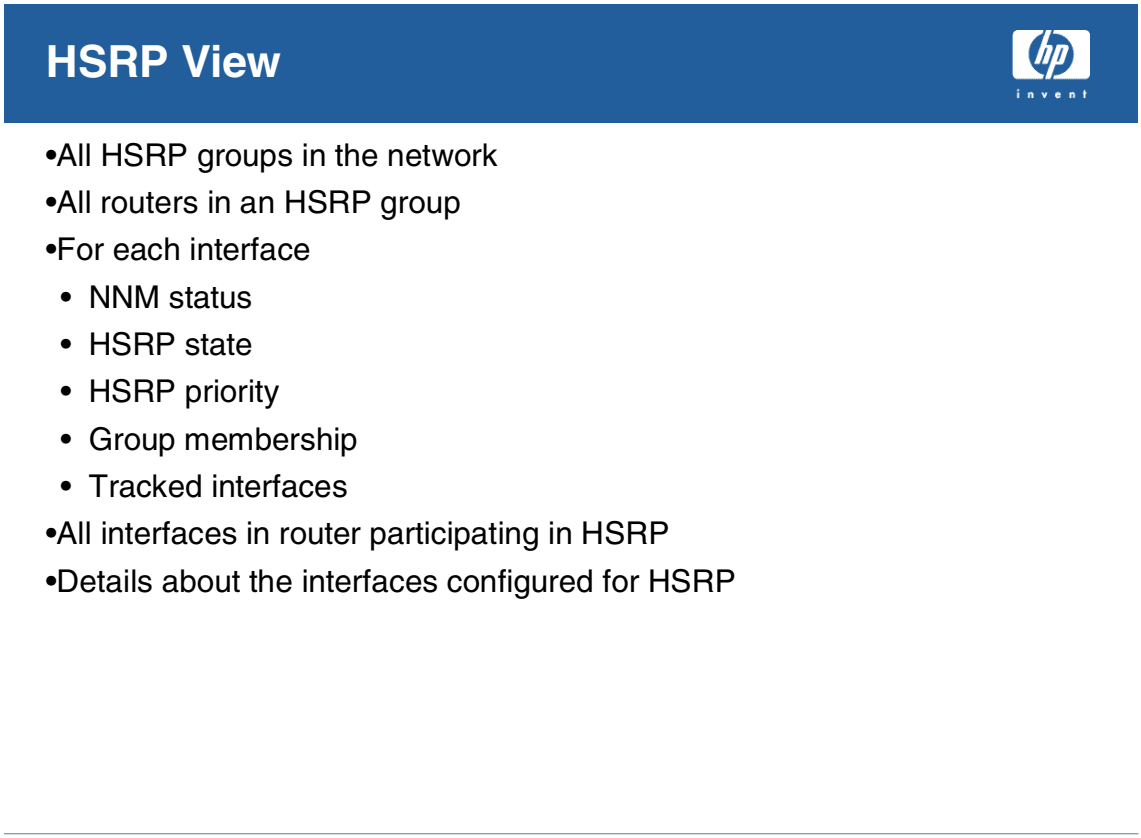
To Disable HSRP Monitoring

To disable HSRP monitoring, answer “no” to the HSRP question in `setupExtTopo.ovpl`. No HSRP objects will be in the topology and none of the correlators will fire.

To disable HSRP correlators, you must disable all 6 HSRP correlators in Composer. From the ECS GUI, select Composer and click [`Modify`]. For each of the following correlators, uncheck the Enabled box:

- OV_HSRP_StatusChange (contains complete information in its Description)
- OV_HSRP_NewNNMEtTopo
- OV_HSRP_ProcessIfEvent
- OV_HSRP_ProcessTrap
- OV_HSRP_Trap_DrillDown
- OV_HSRP_If

HSRP View Details



HSRP View is a dynamic view that presents the HSRP related information in the network.

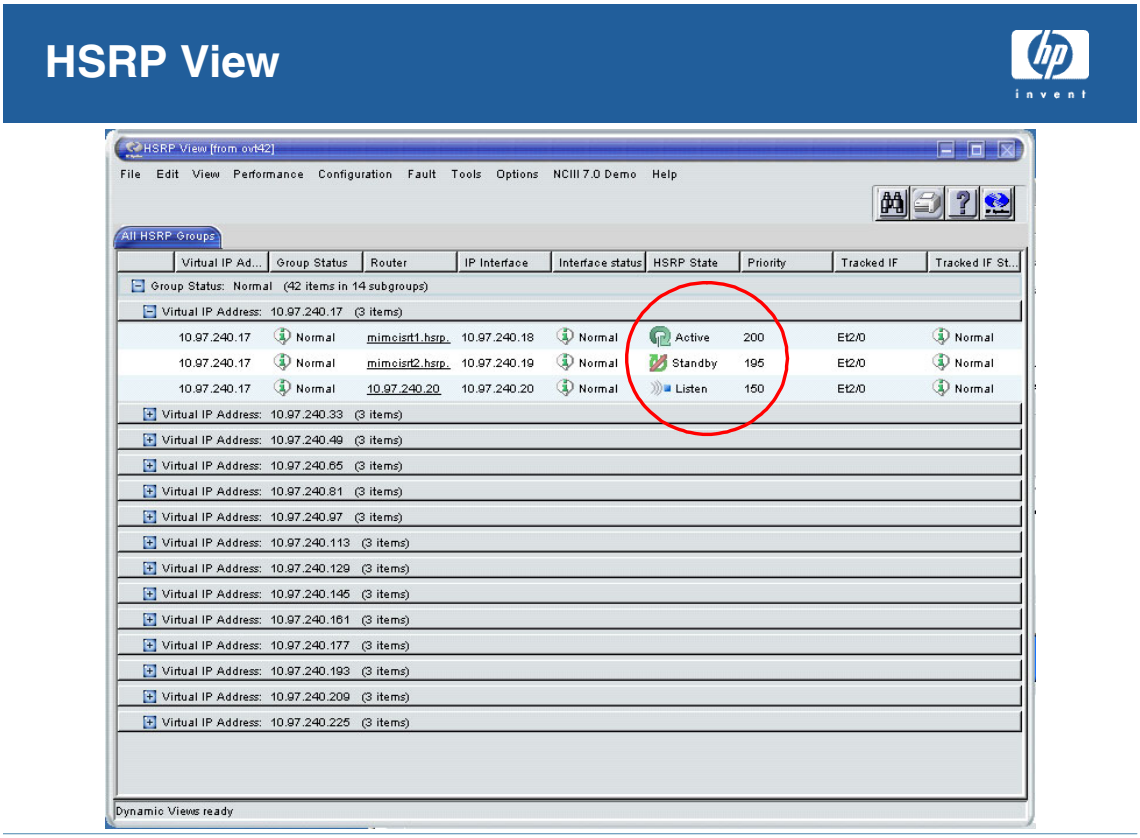
This feature gives you information about:

- All the HSRP groups in the network.
- All the routers in a group with their NNM statuses and HSRP states. All the interfaces in the group. For each interface, the NNM status, HSRP state, priority, group membership and tracked interfaces are specified. The view also shows the NNM status for each tracked interface.
- All the interfaces in a router that participate in the HSRP protocol.
- Details about the interfaces configured for HSRP.

Home Base has a HSRP View link. You can also get to this view from the Alarm Browser. You probably look at the view only for a short period of time to see the immediate effect of the event.

It can also be used as an inventory if you need to know all the routers that participate in a group. Double clicking on a router shows all the interfaces that belong to that router. This includes interfaces that have been configured for HSRP as well.

HSRP View



If HSRP monitoring is enabled, the view is dynamic for an HSRP state change event at the group view. The status update delays 10 seconds to allow the network to stabilize.

The help page describes the HSRP status that each of the colors represent.

Since a group is not a node, Neighbor View is grayed out in the top menu bar and popup menus when you select a group.

HSRP Operation

Extended Topology collects information from devices that support HSRP. The following data is collected from the `cisco-hsrp.mib`: `cHsrpVirtualGroup`, `cHsrpGrpStandbyState`, `cHsrpGrpPriority`, `cHsrpGrpNumber` along with IP interface and `Ifindex`. This additional information is collected from the `cisco-hsrp-ext.mib`: `cHsrpTrackedPriority`, `Ipaddress` and `Ifindex` of the tracked interface.

The Extended Topology solution is achieved by interactions between `ovtrapd`, `pmd`/ECS/Composer, `netmon`, `ovtopmd`, `ovwdb`, and Extended Topology topology API. The solution also does not support the load balancing scenario because of inconsistent data model between NNM and Extended Topology when one interface has many IP addresses.

Composer Correlators

Composer contains a few correlators to support this work. Correlators process OV_IF events and HSRP traps and trigger the status monitoring of HSRP objects. In the OV_HSRP_StatusChange correlator, you can configure NetmonGroupPollWaitTime to configure the wait time for netmon polls to finish.

HSRP related OV events

Events have been created under the Status Alarms category in NNM for informing the user about HSRP group status change conditions.

- `OV_HSRP_No_Active`: The reported HSRP group is completely inoperational.
- `OV_HSRP_Multiple_Active`: The reported HSRP group is in an abnormal condition as there are multiple active interfaces.
- `OV_HSRP_No_Standby`: The reported HSRP group has no standby interface.
- `OV_HSRP_Degraded`: The reported HSRP group has an interface that is not functioning properly
- `OV_HSRP_FailOver`: The reported HSRP group has changed its active interface.
- `OV_HSRP_Standby_Changed`: The reported HSRP group has changed its standby interface.
- `OV_HSRP_Normal`:The reported HSRP group is now functioning normally.

HSRP Interaction with netmon

- netmon automatically excludes HSRP virtual addresses from discovery
  - Identifies devices running HSRP
  - Adds virtual IP address to netmon.noDiscover
- You may need to clean out the topology database if automatic discovery was done before Extended Topology configuration.

©2004 Hewlett-Packard Development Company, L.P.

If you are using HSRP, you do *not* need to add the virtual IP address of your HSRP-configured routers to the `$OV_CONF/netmon.noDiscover` file (for NNM versions 7.5 and later). `netmon` automatically detmines if a router is participating in HSRP (`isHSRP`) and what the virtual IP address of the group is. If any interface on the router is associated with the virtual IP address, the interface is set to migratable and a different IP address is selected as the management address.

The `netmon` process sometimes discovers the HSRP virtual group IP address for a router that participates in HSRP. Since the virtual IP address keeps moving from one router to another, this address should not be used to represent the management address. This can be avoided with the help of the `migrateHsrpVirtualIP` flag. By setting this flag to TRUE (default), you can avoid manually adding a router's HSRP virtual group IP address to the `netmon.migratable` or `netmon.noDiscover` files.

`netmon` does not attempt to discover or manage the virtual IP address of an HSRP group and does not manage the actual IP address of router interfaces in the HSRP Group. This stops NNM from deleting and rediscovering the virtual IP address in a regular pattern, which can cause participating routers to be deleted from and re-added to your map.

Checking NNM for Correct Handling of HSRP Virtual IP Addresses

To make sure NNM is configured for correctly handling HSRP virtual addresses, use the following procedure:

1. Edit `$OV_LRF/netmon.lrf`.
2. Look for the following text within the file and verify that the bold text is present.

   ```
   OVs_YES_START:ovtopmd,pmd,ovwdb:-P -k segRedux=true -k migrateHsrpVirtualIP=true
   :OVs_WELL_BEHAVED:15:PAUSE
   ```

3. If the bold text is present, then NNM is handling the HSRP virtual IP address correctly.
4. If the bold text is not present, add the text and save the file. You also need to restart the `netmon` process as follows:

   a. `ovstop netmon`

   b. `ovaddobj netmon.lrf`

   c. `ovstart netmon`

HSRP Discovery with Pre-existing NNM Topology

If you already have an NNM topology database (for example, if you have installed NNM and proceeded with an automatic discovery of your network), and the contents of the `netmon.lrf` shows `migrateHsrpVirtualIP=false` or doesn't contain the `migrateHsrpVirtualIP` parameter, then the database is likely to contain the virtual IP addresses of your HSRP groups. This is especially true if you used NNM's auto-discovery feature.

You have two tasks:

1. Verify that NNM is correctly managing HSRP virtual IP addresses in your environment.
2. Remove from NNM any HSRP virtual IP addresses that NNM currently manages.

   If you are using an existing database discovered by an older version of NNM, and NNM has already discovered the virtual IP address of your HSRP-configured routers, use the following procedure to remove the virtual IP addresses:

   a. Execute `ovtopofix -r` *`a.b.c.d`* where *`a.b.c.d`* is the virtual IP address of your HSRP-configured routers. You may want to remove the node (using the NNM ID) rather than the IP address. The NNM ID can be obtained with the ovtopodump command.

      See the ovtopofix(1M) and ovtopodump(1M) reference pages in NNM's online help (or the UNIX manpages) for details on using these commands.

   b. Run `ovtopodump -lR` to make sure the virtual IP address is gone.

Validating Your Results

Validating Discovery

•After Extended Topology discovery:

- If the virtual IP address appears in IP interface column, remove the interface.
- If an HSRP group only has one router, check that the actual IP address is present, NNM has SNMP access, and the interface is up during discovery.
- Review HSRP status indications.

©2004 Hewlett-Packard Development Company, L.P.

Validating Your Results

To verify that your HSRP discovery is correct, run another Extended Topology discovery, using `etrestart.ovpl`.

When the discovery finishes, open the HSRP Group Detail screen for each group and check for the following:

- If the HSRP virtual IP address shows up in both the IP Interface column and the Group Membership column, NNM is most likely managing the virtual IP address. You need to remove this interface from NNM.
- You may see an HSRP Group with only one router, even though an HSRP Group must have at least two routers. This could have several causes. To resolve the problem, do the following:
  - — Ensure that the actual IP address of the missing HSRP router is in NNM by running `ovtopodump` *`actual_IP_address`*.
  - — Make sure that NNM has SNMP access to the missing router.
  - — Rerun discovery when all the HSRP router interfaces are up.

Note that if an HSRP router interface is down during discovery, the Advanced Routing SPI will discover HSRP groups incompletely. This is because the HSRP MIBs at the time of discovery reflect the actual state. This should be a transient problem that resolves when an Extended Topology discovery occurs after the interface is back up.

Potential Status Anomalies

Subtle conditions can lead Extended Topology to yield unexpected status for HSRP Groups.

- First, if there are only two router interfaces in an HSRP group, and one of them is down during Extended Topology discovery, the group is seen as having only one interface in it.

  In this case, the status is marked Unknown, because for HSRP to operate properly at least two interfaces are required. But one of the two HSRP interfaces was down during discovery, so the HSRP Group appears to Extended Topology to be invalid.

  If another Extended Topology discovery finds that the down interface is back up, the group is properly discovered, and its status is correctly computed.

- Second, if there are more than two router interfaces in an HSRP group, and one or more than one is down during an Extended Topology discovery, the HSRP group will be incompletely discovered and the status will be incorrectly maintained until a discovery happens which finds all the router interfaces of an HSRP group.

Lab Exercises

•Objectives:

Understand a “correct” HSRP discovery

Become familiar with HSRP-related database contents and support tools

©2004 Hewlett-Packard Development Company, L.P.

HSRP Lab A: Full HSRP Discovery

Objectives:

- Recognize and understand NNM Extended Topology HSRP discovery.

At the completion of this module you will be able to:

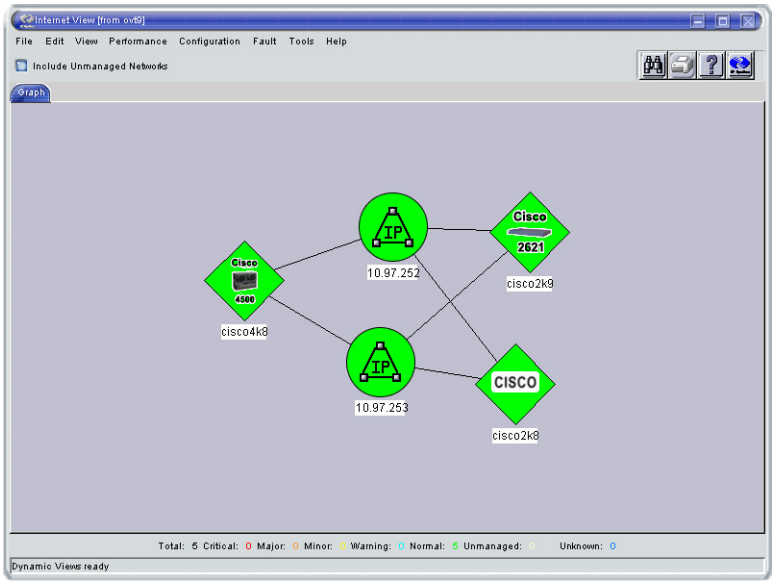
- Recognize a complete HSRP discovery.
- Observe what happens as HSRP states change.
- Increase familiarity with the NNM Extended Topology support tools.

CAUTION Successful execution of this lab depends on following the directions given by your instructor AT THE TIME those instructions are issued.

Directions

1. `cd $OV_CONTRIB/NNM7labs/Lab_hsrp`
2. When the instructor informs you to proceed, execute the setup script:

   `setupLab_hsrp.ovpl`
3. When the setup script completes start Home Base. Launch the Internet View.
4. The map should appear as shown.



5. Commence the Extended Topology discovery. Use the Home Base Discovery tab or execute:

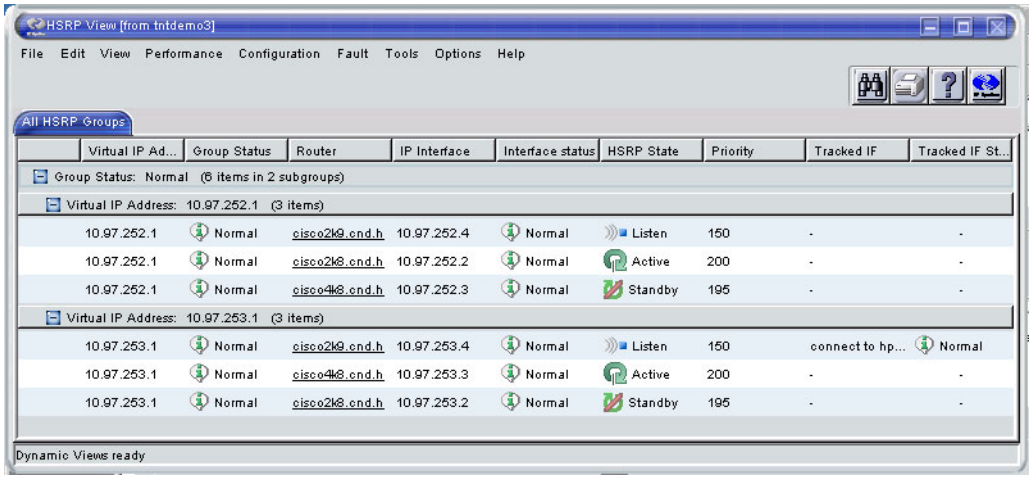
   `etrestart.ovpl`
6. Since this topology only contains three nodes, the discovery will complete quickly. In order to view the results of the support commands such as

   `dumpAgentProgress.ovpl` *{agentName}*

   you must act quickly while discovery is in progress.
7. View the `dumpDiscoStatus.ovpl` results to see what agents are included in the discovery process.
8. Use the appropriate command to view the agents. Run this HSRP base discovery several times so that you have a chance to view the status of the agents.
9. After discovery has completed (`ovstatus -v ovet_disco` shows Awaiting next discovery) dump and review the contents of the database using:

   `ovet_topodump.ovpl -info`

10. Launch the HSRP dynamic view.



11. These views collectively show an accurate and correctly discovered HSRP environment. Each HSRP group has at least two interfaces of which one is "`active`" and the other is "`stand-by`" (or other HSRP state).

12. Setup up your desktop environment so that you can view the All Alarms browser and the HSRP View for both groups.

    Now execute: `$OV_CONTRIB/NNM7labs/Lab_hsrp/bin/sendStateChangeEvents.ovpl`

    This simulates state changes on the devices. You should see the active devices go down and their HSRP states become unknown. Then the standby devices become active, whereas the listen devices become the new standby devices.

13. When you have completed running the HSRP discovery and have analyzed each view, inform the instructor that you are done with this portion of the HSRP lab.

CAUTION Do not proceed to the next phase of the HSRP lab until the Instructor tells you to do so. The Instructor must configure teh MIMIC server for the next phase of the lab.

LAB A2: Automatic HSRP state change

Objectives:

- Recognize and understand how the HSRP view follows real-world state changes.

At the completion of this module you will be able to:

- Recognize automatic HSRP state changes.

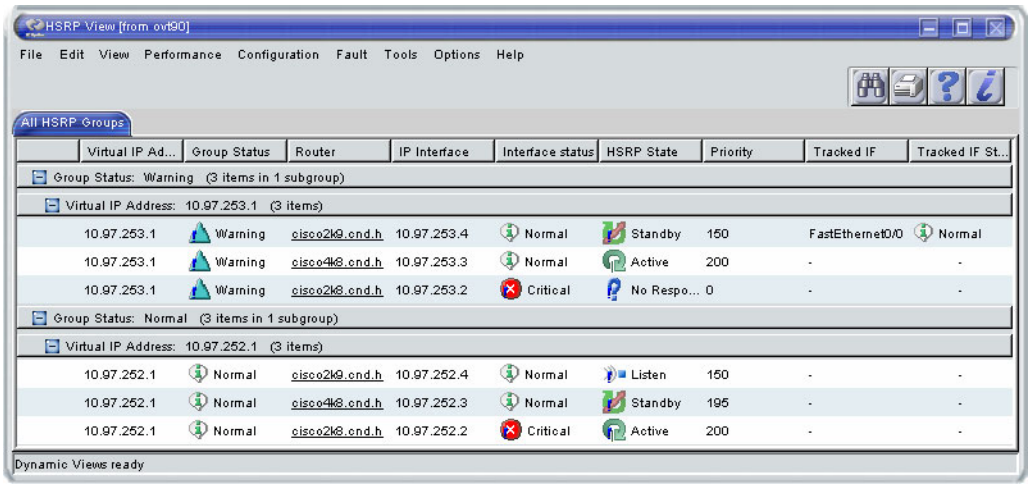
CAUTION Successful execution of this lab depends on following the directions given by your instructor AT THE TIME those instructions are issued.

Assumptions:

- Student has completed Lab A.

Directions

1. Open the HSRP view and expand to see the detail of all groups.
2. When your instructor has prepared the simulation server, do a demand poll of cisco2k8:
   a. Highlight the device cisco2k8 in the 10.97.252.1 group.
   b. Select `Fault:Network Connectivity:Poll Node`. Let this time out.
3. You should see devices 10.97.252.2 and 10.97.253.2 become Critical.
4. Within approximately 5 minutes, you should see the HSRP state changes similar to before:
   - The down device now becomes unknown.
   - The previous standby device is now active.
   - The previous listen device is now standby.
5. In addition, this time you should see the entire HSRP group change its overall status.



6. When all students have seen this, the instructor will simulate cisco2k8 returning to Normal.
7. Redo step 2 (repoll the critical node). It should now become normal.
8. Within approximately 5 minutes you should see the HSRP states return to their correct state.
9. Note the alarms in the All Alarms Browser. Select an alarm and see which views are available

under `Actions:Views`.

(Optional) Lab B: Partial HSRP Discovery

Objectives:

Recognize and understand a commonly occurring Extended Topology HSRP discovery issue.

At the completion of this module you will be able to:

- Recognize an incomplete HSRP discovery.
- Use the Extended Topology support tools to troubleshoot the HSRP discovery.

CAUTION Successful execution of this lab depends on following the directions given by your instructor AT THE TIME those instructions are issued.

Assumptions:

- Student has completed Lab A.

Directions

1. Change your working directory to `$OV_CONTRIB/NNM7labs/Lab_hsrp`.
2. WHEN THE INSTRUCTOR says to continue, commence the Extended Topology discovery. From Home Base, use the Discovery tab or execute:

   ```
   etrestart.ovpl -verbose
   ```

3. Since this lab only contains three nodes, the discovery will complete fairly quickly, though slower than the previous lab. In order to view the results of the support commands such as

   `dumpAgentProgress.ovpl` *`{agentName}`*

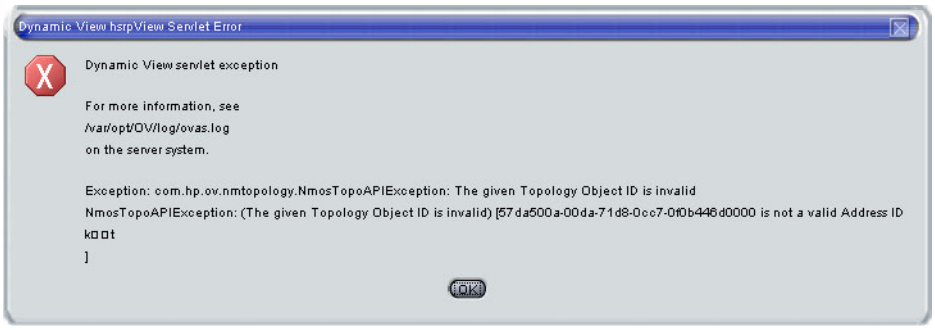
   you must act quickly while discovery is in progress.
4. View the `dumpDiscoStatus.ovpl` results to see what agents are included in the discovery process.
5. Use the appropriate command to view the agent progress. Run this HSRP discovery several times so that you have a chance to view the status of the various agents.
6. Can you determine which discovery agent(s) are having difficulty?
7. Which node(s) are causing the problem and why? Hint, from the Topology Summary, select `Doesn't Respond to SNMP`.
8. Dump and review the contents of the database using:

   ```
   $OV_BIN/ovet_topodump.ovpl -info
   ```

9. Tail the `$OV_PRIV_LOG/ovet_disco.log` during the HSRP discovery shows the following:

   ```
   In updateHsrpStatus
   In setHSRPStatus
   Cannot process HSRP group10.97.252.1 with fewer than 2 members
   ifOidListString is
   Could not process group 10.97.252.1 due to errors.
   In setHSRPStatus
   Cannot process HSRP group10.97.253.1 with fewer than 2 members
   ifOidListString is
   ```

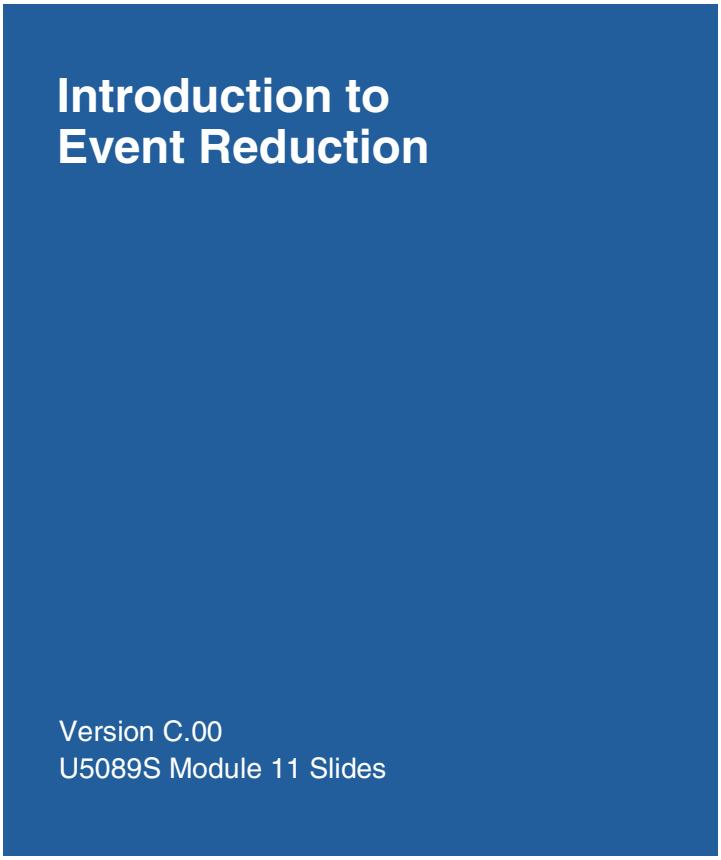
```
Could not process group 10.97.253.1 due to errors.
```



10. Attempt an HSRP View from Home Base.
11. These views collectively show a commonly occurring problem with HSRP discovery and one that you should be able to recognize. Without appropriate SNMP access (and/or all devices within the HSRP group have been discovered by NNM AND managed) to the routers configured in an HSRP group, NNM Extended Topology will show only the partial configuration.

11 Introduction to Event Reduction

Module Objectives

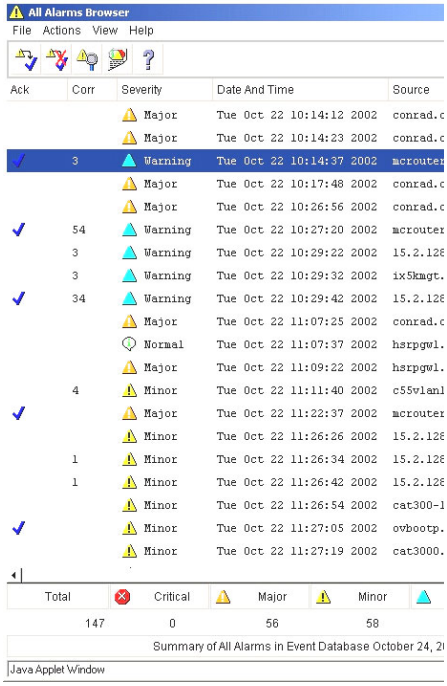


At the completion of this module the student will be able to:

- Describe the concept of ECS correlations and how NNM processes event correlation.
- List the forms of event reduction available in NNM.
- Compare operation of de-duplication and pattern deletion with ECS and Composer.
- Identify the best means to remove unwanted events from the Alarm Browser for a given situation.

Fundamental Objective: Event Reduction

Fundamental Objective: Event Reduction

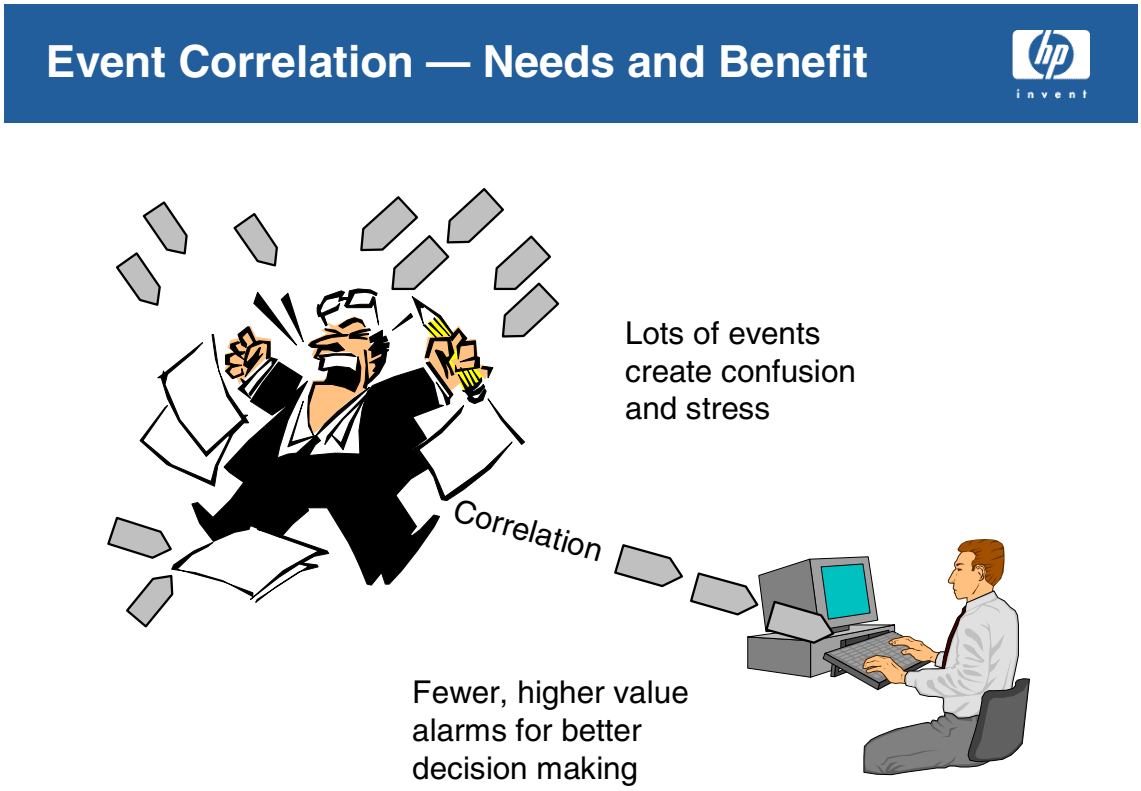


- Want the operator using the Alarm Browser to see
  - the fewest possible number of alarms
  - with the most useful information
  - to solve the root problem the fastest.
- The Alarm Browser should reflect the current state of the network and services.

©2004 Hewlett-Packard Development Company, L.P.

The NNM Alarm Browser should provide the Operator with a grasp of the current state of the network and its services. However, some devices emit spurious events that cause “noise” in the browser. Some events in the network get cleared up, yet their alarms could remain in the browser. Sometimes several alarms actually point to the same root cause. To maximize Operator efficiency and minimize the time to repair a network problem, the Alarm Browser needs to be tuned to focus on the highest priority alarms that point directly to the problems in the network. NNM provides several mechanisms to tune your Alarm Browser.

The Need for Event Correlation



What Is Event Correlation?

Event correlation is the process by which the relationship between events is identified. Once relationships are identified, the process can produce a smaller number of new events with the same or more meaningful information.

Event correlation simplifies the display of alarms by analyzing the stream of events to find multiple events stemming from the same root cause and displaying only the root cause. The additional (secondary) events are available for viewing through menu selections.

Benefits of Event Correlation

The relationships between input events can be used to:

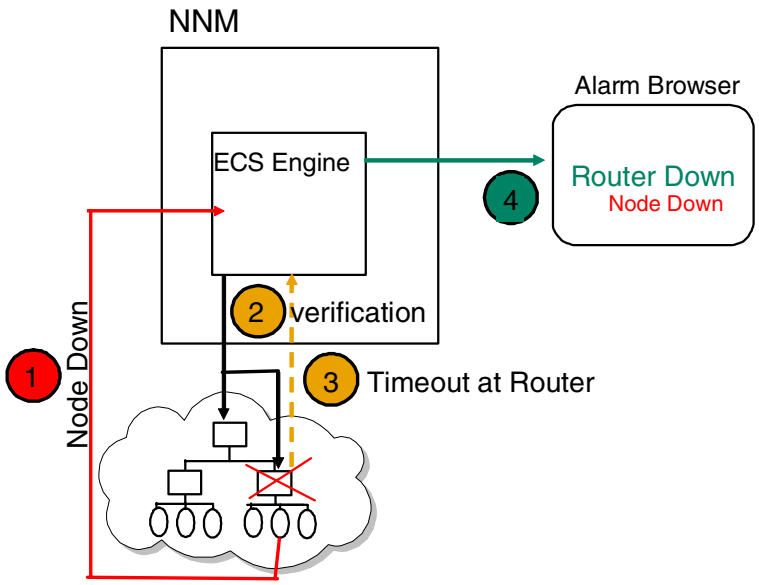
- Reduce the number of events. Events can be “allowed through” based on previous, current, or subsequent events. This is different from simple event filtering which does not compare events with other events.

- Detect problems by their signatures, where a signature typically consists of multiple, time-separated events, possibly delayed by the network or delivered out of order.
- Modify events to contain more information.
- Create new events. A problem may have several symptoms which generate events. By analyzing the relationships between events, the root cause or problem can be identified.
- Events can be sorted. For example, based upon latency in the network, some events may take longer to arrive than others.

Correlation Example

Correlation Example

Connector Down Correlation



U5089S ©2004 Hewlett-Packard Development Company, L.P. 4

The ConnectorDown correlation notices incoming Node Down events. When it sees one, it verifies the expected path to the node (based on the topology database) to see whether any intervening device is unreachable. When an intervening device does not respond, ConnectorDown sends out a new alarm that the connector is down and needs to be fixed. Information that the original node is down is still available, but is correlated under the Connector Down alarm as a secondary failure.

ECS and HP OpenView

ECS and HP OpenView

•HP OpenView Event Correlation Service (ECS) is a
- high-speed,
- real-time,
- multi-protocol

event correlator.

•Available with NNM and other OpenView products.

- Engines
- Integrated with NNM
- Integrated with OVO

- Protocols
- SNMP
- OVO messages

•The ECS Correlation Designer is a separate product.

U5089S ©2004 Hewlett-Packard Development Company, L.P. 5

The HP OpenView Event Correlation Service (ECS) has been integrated with NNM. The event flow paradigm has events flow through a correlation consisting of interconnected logic blocks, analogous to a current flowing through an electric circuit.

The ECS engine processes events through a set of logical blocks either to completion (suppressed or output) or to be held pending some future conditions being evaluated. Decisions are made based on the attributes of the individual events as they flow through the correlation nodes, together with the values of attributes of earlier events or events yet to arrive. Event attributes may be modified. Attributes of multiple events may be consolidated into a new event. Information from outside the engine may be retrieved to be used to make decisions or modify event attributes.

Event Correlation Service (ECS) can provide many benefits:

- Generic correlations can be tuned for a specific network environment.
- Continuous real-time operation under event storm conditions.
- Produces real-time event correlation, reduction, filtering, and suppression of transient, redundant, and implied events.
- Handles delayed events and events delivered out of creation-time order.
- Correlates multiple time-separated events, not just the filtering of discrete events.
- Can create new alarms with data consolidated from source events.

- Can combine information from external sources with data from events to suppress alarms or create new alarms with higher value.
- Monitors for the absence of events.

ECS is delivered as two distinct products:

- The “ECS Engine” is a high performance, real time processor of multi-protocol event streams.
- The “ECS Designer” is a graphical development environment to create and test event correlation logic.

The ECS correlation engine is a general-purpose software module that can handle several different types of events: SNMP, CMIP, ASCII and OVO messages. The engine is available for several environments:

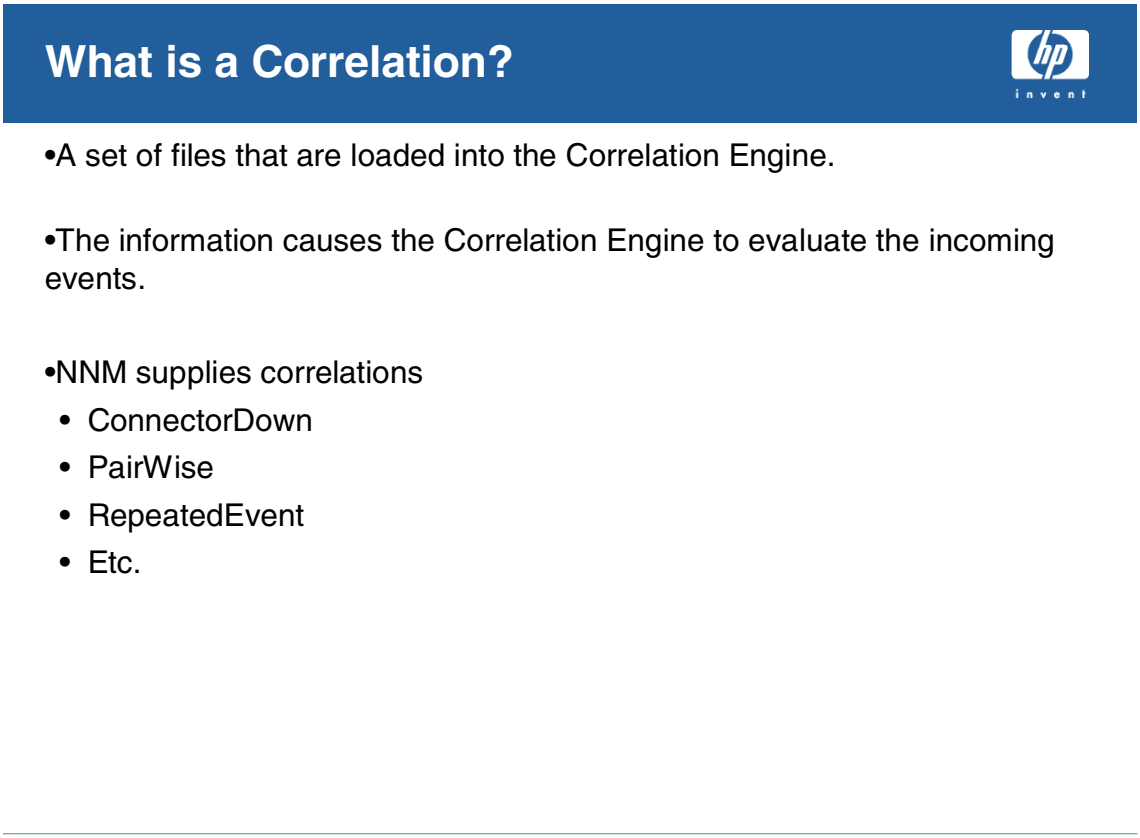
- Stand-alone ECS engine. As designed, the ECS engine module could be incorporated into other applications. A third party might create a flow of ASCII-based events from which the engine would correlate meaningful output stream. Such an instance would not share any data with NNM.
- NNM integrated engine. The ECS engine is linked as a sub-stack of `pmd`. The NNM supplied correlations only process SNMP events.

  In this course we address HP OpenView NNM and the user interface to its configuration and management utility.
- OVO integrated engine. The ECS engine is integrated with the OVO agent (HP-UX) and management server processes. The OVO ECS engine processes events after they have entered the OVO message stream through the OVO Message Stream Interface (MSI). NNM events from the “Raw” flow come into the OVO process, where they are reformatted into OVO events. Once they are OVO events, the OVO ECS engine handles them in its stream with other OVO events. This is installed separately from NNM and only receives NNM events after they have been processed.
- Extensible ECS for NNM.

  Extensible ECS for NNM has the ability to add an ECS correlation logic that uses an 8-bit message as the source of an event.

Since NNM correlates only SNMP events from the Raw flow and OVO correlates only OVO events, their operation is conceptually independent. None of the supplied correlations take a stream of mixed events and correlates them into a meaningful output stream.

What is a Correlation?



A correlation is a set of rules to control the flow of events. The correlation rules consist of interconnected logic blocks known as “correlation nodes.” Each node has attributes and values that can be defined to control its behavior on how the events are handled. Each correlation:

- Has a unique name.
- Causes the Correlation Engine to perform a task on incoming events such as analyzing them, filtering them or recognizing patterns in the stream of incoming events. Such information can signify faults in the network or human activity such as planned maintenance.
- Can usually be modified to alter the behavior of the Correlation Engine. This is done by assigning different values to the correlation parameters.
- Always contains a description explaining when and how the correlation should be used. If parameters can be modified, there is usually extra information which defines the allowable limits and values that can be assigned to the parameters. Observe this information when deploying or modifying the correlation.

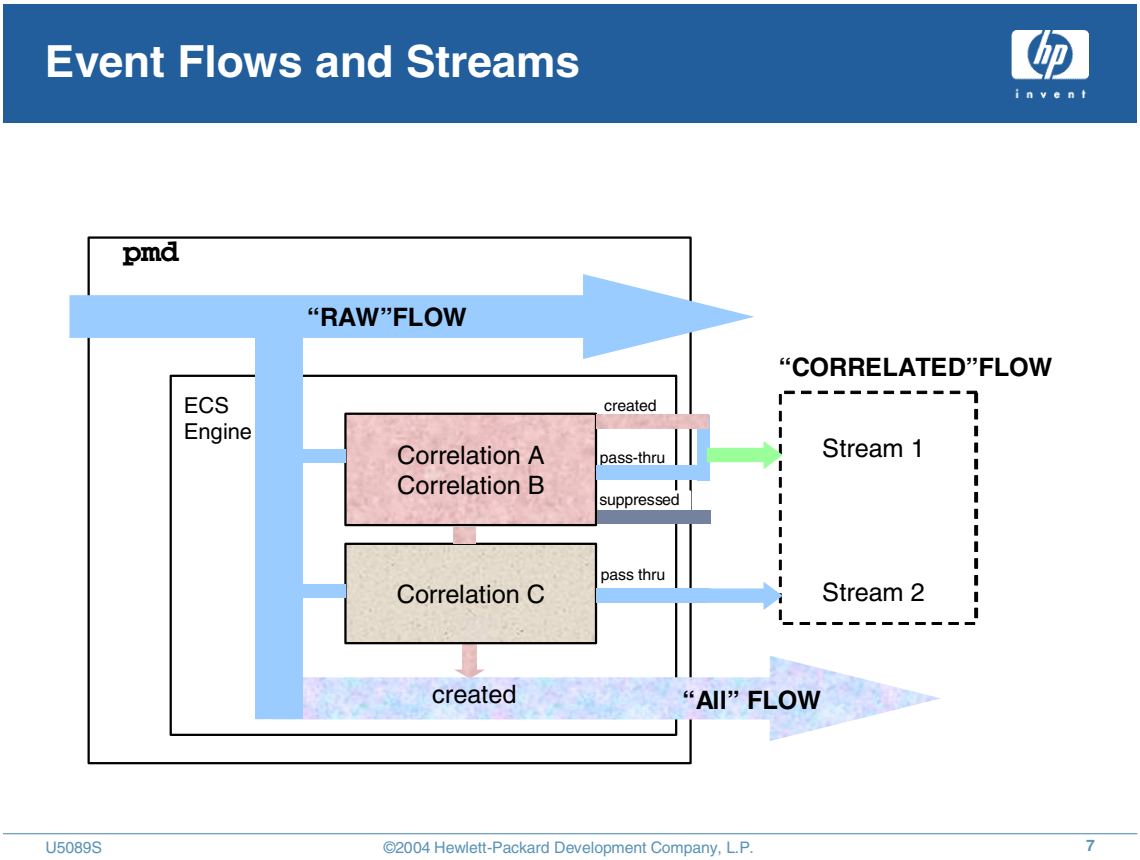
- Can reference external data sources if supplemental information is needed to make determination of what to do with the event.
- Can call an external program known as an “Annotation server” if additional information is needed.

Correlation and Suppression

Correlation is the act of associating a child event with a parent event. Child events can be viewed in the Alarm Browser by selecting `Actions:Show Correlated Events`.

When an event that has entered `pmd` is prevented from leaving `pmd` via a particular stream, the event is said to be **suppressed** in that stream. A suppressed event is **not displayed** in the main window of the Alarm Browser, but may be displayed in the `Show Correlated Events` dialog.

Event Flows and Streams



Events pass through the NNM system in flows and streams. NNM operates with three event flows:

- Raw: The set of all events that come into NNM.
- Correlated: The set of events coming out of all correlations.
- All: The set of all Raw events plus all events potentially created by correlations.

The Correlated event flow is further divided into streams. This is a conceptual flow only. Processes may listen to the Raw flow, the All flow, or to a specific correlated stream. Flows and streams are the same in content, events, and differ only in their sources. The term "streams" applies to those groups of events produced from correlations. A stream is a distinct, logically independent flow of events through an ECS engine.

Initially there is only one stream, called "default." The ECS engine supports multiple independent streams. Each stream contains one or more correlations. Individual correlations can be configured on one or more streams. All streams physically reside in the same ECS engine and receive the same input events. Each stream can be subscribed to by different applications.

An event may be present in more than one stream. All correlations receive the same input events. Each stream is correlated independently, which means that the fate of an event in one stream does not influence the fate of the same event in another stream.

There is only one correlation, no matter how many streams it may be in. It processes events only once. A correlation can be enabled in one stream and disabled in another stream.

All correlations supplied with NNM operate with the stream called "default." The Alarm Browsers and `ovactiond` listen to the default stream.

A third party application might create a different group of correlations for a stream. For example, Cisco might create a stream which applies a correlation they create to select only events relevant to Cisco routers. The output would be far too restricted for general NNM applications, so they could not add their correlation into the Default stream.

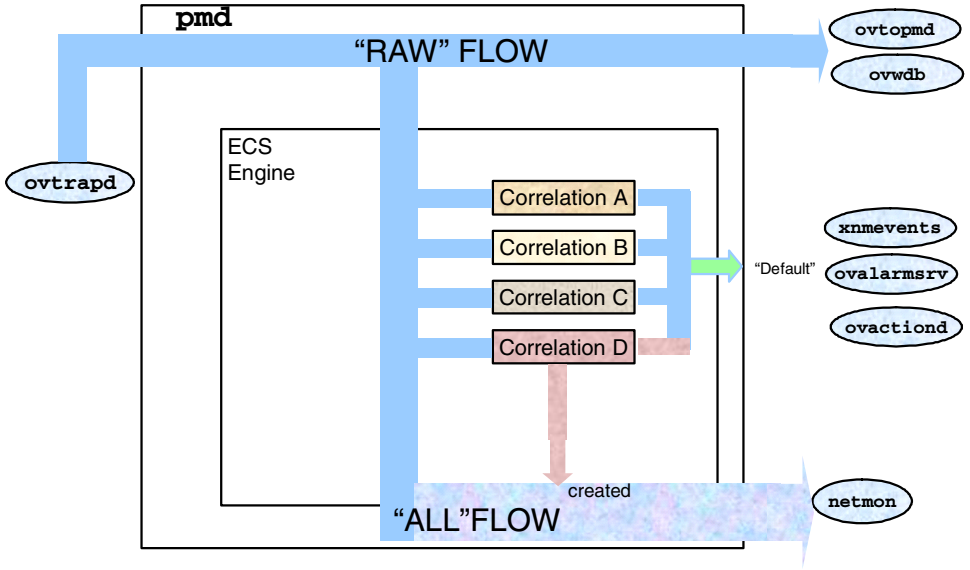
NNM Stream Policy Style Guides for Default Stream

Each stream has its own output "policy" which determines the final output of the stream. Once all correlations enabled for a stream have processed the data, final logic determines the result of their combined output. For the default stream, the logic imitates circuits running in parallel. That is,

- Output the event unless discarded by a correlation in the stream.
- Output the event only when all correlations in the stream output the event.
- Any correlation can suppress the event.

Architecture of ECS in NNM

Architecture of ECS in NNM



©2004 Hewlett-Packard Development Company, L.P.

NNM events are routed through the correlation engine (ECS run time) by the `pmd` service.

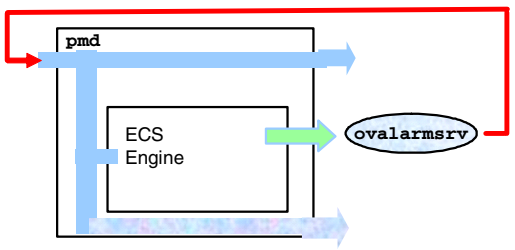
Subscribing applications receive the stream of correlated events to which they subscribe. Applications such as `ovactiond` and the Alarm Browser receive the default stream which may have been correlated by the correlation engine.

Operation of the correlation engine (ECS run time) is controlled through the GUI for the configuration and management utility known as the ECS Event Configuration GUI.

Improving on What Correlations Do

Improving on What Correlations Do

- Correlations monitor event relations for a configured time window.
- Related events may appear later or conditions may repeat.
- Alarm Browser should only reflect current network state, not all history.
- Clean up the history in the Alarm Browser
  - Event de-duplication
  - Pattern deletion
- Feed information on clean up back to **pmd** for correlation.



U5089S ©2004 Hewlett-Packard Development Company, L.P. 9

PairWise and RepeatedEvents correlations monitor events for your defined time window. Events that arrive outside the time window do not get correlated and remain in the Alarm Browser, showing history rather than the current state of the network.

Increasing the time window to capture all such events causes `pmd` to consume more memory and invalidates the relationship between the monitoring and your Service Level Agreements.

Therefore, NNM provides additional mechanisms in `ovalarmsrv` to clean up the browser to reflect the current state of the network. These include event de-duplication, which extends the reach of RepeatedEvents correlation, and pattern deletion, which extends the PairWise correlation.

What If I Have Unique Needs?

What If I Have Unique Needs?

- Purchase third party correlations

- ECS Designer
  - Graphical editor for developing correlations
    - Design
    - Simulate
  - Electrical circuit paradigm
  - HP-UX, Solaris, and Windows
  - Developers' license only; No runtime license required

  - Initial investment in software
  - Requires extensive training

U5089S ©2004 Hewlett-Packard Development Company, L.P. 10

Third Party Correlations

You can add third party correlations without buying the Extensible ECS for NNM. HP will provide the ECS Designer to any premier partner.

Contributed Edition correlations are provided by HP OpenView Partners. Check the HP OpenView web page for links. HP does not test or support contributed correlations. They are tested and supported by the partner.

Extended Edition correlations can be obtained from system integrators, such as HP Consulting Services and others. They are tested and supported by the supplier.

ECS Designer

The ECS Designer is available as a separate product, allowing you to create custom correlations for your environment. No runtime license is required for the correlations you create.

The user interface for ECS Designer is an easy to use graphical editor with selectable icons for

over a dozen primitives. By combining the primitives in a topology which reflects your desired logic, you can create correlations of arbitrary complexity. Each primitive may also contain a function written in Event Correlation Description Language (ECDL) to further focus the results.

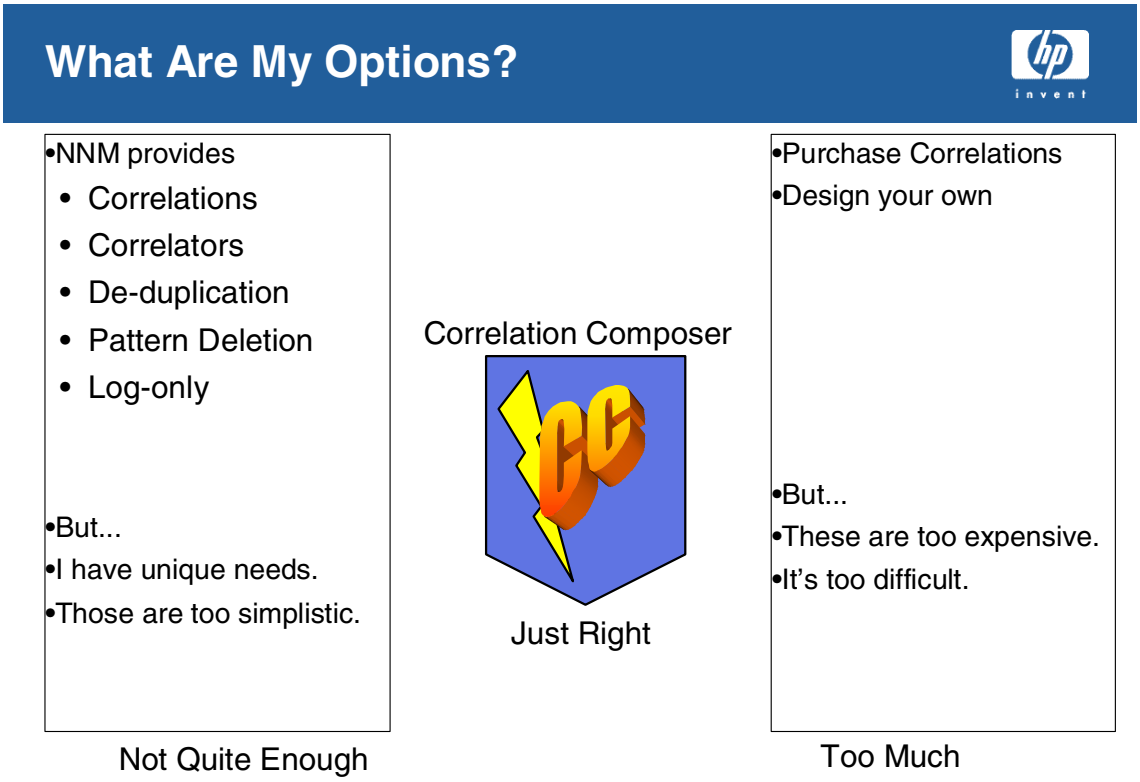
Once you have developed your correlation in ECS Designer, you can actually simulate its operation to test it before putting it into production.

Each correlation uses an electrical circuit model, which means that events enter from a source, navigate the logic, and possibly exit at the sink. The logic consists of arbitrary connections of primitives, each of which contains state information and an ECDL program.

ECS works with many kinds of event types to integrate with multiple products. You can even combine multiple event types in one correlation.

For example, you could create a correlation that correlates both SNMP and ASCII events, and outputs only SNMP events (since other NNM components only support SNMP). Or, in the simplest case, an ECS correlation accepts ASCII events and converts them to SNMP so all NNM applications can then take advantage of those events.

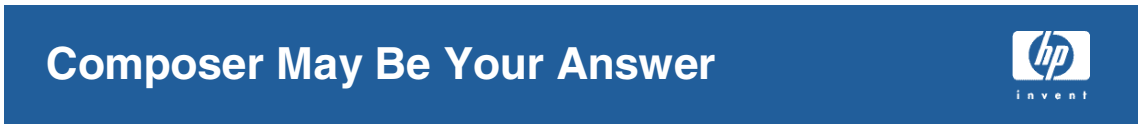
What Are My Options?



For those situations where the NNM-provided correlators and correlations don't address your particular event stream, you can develop your own solution. However, ECS Designer may be more than you need to invest.

For most of your correlation needs, Composer provides a simplified way to develop your own solution.

Composer May Be Your Answer



- For moderately complex event reduction tasks
- that involve relating events to one another
- and can be reduced to the most common objectives:
  - Gather more information before deciding to suppress display of "noise" events
  - Add more data to an event to help operators
  - Count how many times an event occurs
  - Count how many times a transient event occurs
  - Match symptom events with a root cause event
  - Gather symptom events to indicate a root cause event

U5089S ©2004 Hewlett-Packard Development Company, L.P. 12

Composer provides a simplified interface to the work that ECS Designer does by "pre-packaging" the most commonly desired tasks into 6 templates for you to use. By basing your design on a Suppress Template, for example, you can discard events that are unique to your environment.

Composer development may be as simple as identifying the event to the interface and clicking a `Discard` button. Or you may create your own custom functions to be called when certain events arrive to capture unique information and access external data sources to further quantify the true nature of the network problem.

What Does It Take?

What Does It Take?

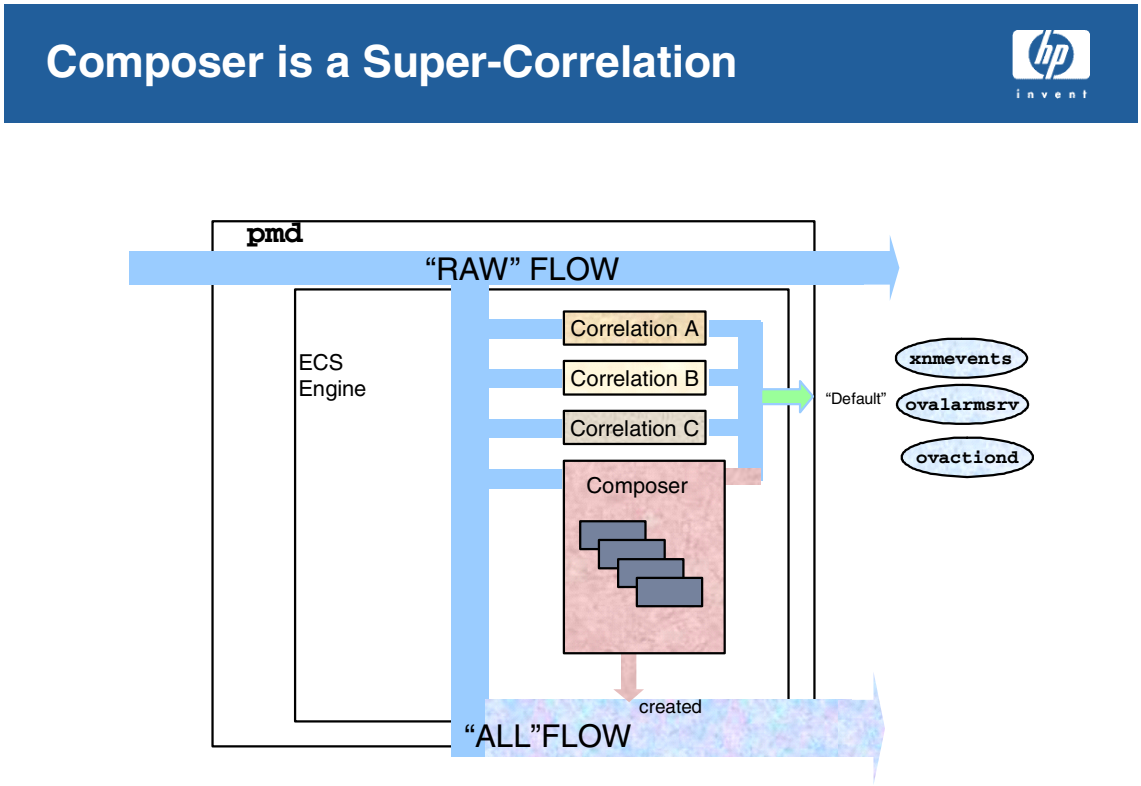
- If you can specify
  - how to detect the incoming events from the right sources
  - how to match up events from the same source
  - how long to watch for them
  - how many you care about
  - what to do when they arrive
    - what you want to happen to the incoming event (discard, keep, update)
    - what other event you may want to create with more data
- You can create a correlator to do the work!

©2004 Hewlett-Packard Development Company, L.P.

Composer's interface allows you to designate events to process based on event header information, such as the specific trap and the agent-address and based on event contents in varbinds. To be able to design a correlator, you must be able to articulate which events characterize your situation.

Furthermore, for successful correlator definition, you need to be able to tell Composer what actions to take when the event arrives, such as discarding the event, modifying it, or creating a new event.

Composer Is a Super-Correlation



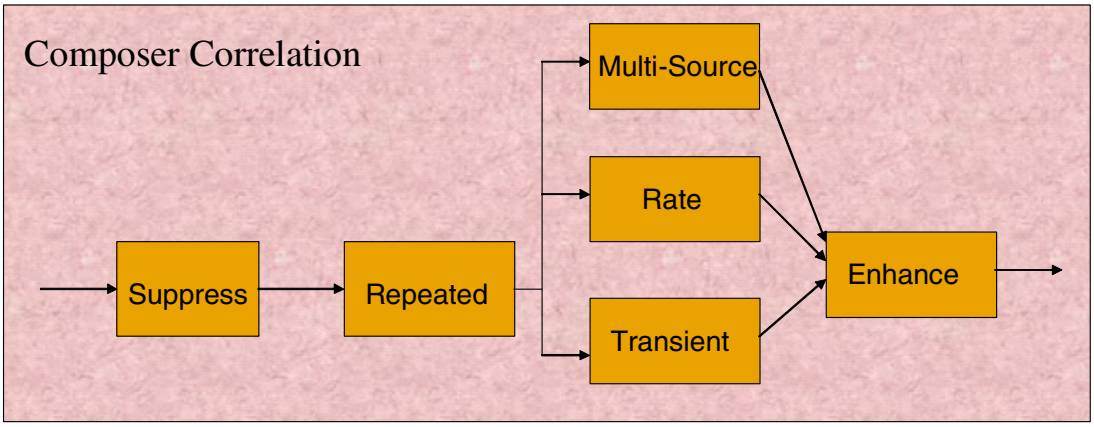
The ECS stack is the correlation engine that performs the correlation logic defined by the correlations and the Composer correlators. The following details the event flow through the ECS engine.

1. Events are first evaluated to see if they match the input signature for any of the active correlations or correlators.
2. Events that don't match any signature are returned immediately to OVEvent. Events that do match are held and, in the case of Composer, are evaluated against the Advanced filters.
3. Composer events that pass the Advanced Filter then have the logic of the correlator executed. All actions from all correlators for the matching event are executed.
4. After processing the event is either held, released or discarded depending on what the correlator has specified.
5. If multiple correlators have the event held then the holding period becomes the longest such period specified by the correlators.
6. Once the event is released, the callback actions are performed and the events are returned back to OVEvent.

Correlator Evaluation

Correlator Evaluation

- Correlator types processed in a specific order
- Event proceeds in evaluation only if no correlator discards it



U5089S ©2004 Hewlett-Packard Development Company, L.P. 15

Each Correlator is implemented by a discrete decision-making mechanism based on the Correlator Template used. If the filters of two Correlators are defined such that they admit the same event, then both the Correlators are applied to the event. When an event participates in multiple Correlations, the following rules are applied to determine the outcome:

- The order of Correlator evaluation is Suppress followed by Repeated, all other correlation rules (in parallel) and finally Enhance correlation.
- If the Suppress and Repeated Correlators choose to discard the event, the user can optionally choose to allow the event to participate in other Correlators before it is discarded.
- An event is output if and only if no other Correlator has discarded it.
- The Enhance Correlation is run last. The event is enhanced if and only if no other Correlator decides to discard that event.

Choosing an Event Reduction Mechanism

Comparing Event Reduction Mechanisms

- Single, independent event by type
  - Log-only or Ignore event configuration
  - De-duplication in Alarm Browser
- Pattern of events
  - Composer correlator
  - ECS correlation
- Combine correlators with de-duplication of the root cause

©2004 Hewlett-Packard Development Company, L.P.

Before attempting to develop a correlation or de-duplication, consider:

- What the operator really wants to see
- Level of complexity in the mechanisms

The following is an ordered list of mechanisms for developing event reduction, starting with the simplest.

1. Log Only or Ignore
2. De-duplicate
3. Composer correlator
4. ECS correlation

Log only and de-duplication are mechanisms that operate on a single event type, independent of other events. Composer correlators and ECS correlations are more powerful in that they can be designed and developed to identify a pattern of events and reduce that pattern to a single root cause. The rationale for having this range of mechanisms is to provide some scale of effort to developing reductions (that is, simple things should be simple to do).

If the event being considered for reduction is independent and of no use to the operators in real time, then the simplest and most efficient mechanism is to configure that event to be LOGONLY. A good example of this is in NNM is `SNMP_Authen_Failure`. This trap is configured as a

LOGONLY trap and a report can be scheduled to run at various intervals to produce a list of hosts and frequencies of an authentication failure for security monitoring.

If the event being considered for reduction is frequent but the operators do occasionally require real time to access to the event data, as with `OV_NodeAdded`, then de-duplication is the most appropriate. De-duplication leaves only the most recent occurrence of the alarm at the top level in the browser with all duplicates correlated underneath. This mechanism also provides a better organization to the alarms in the Alarm Browsers as the duplicate alarms are collected under one top-level alarm as opposed to appearing throughout the browser.

If the event(s) being considered for reduction are not independent and are symptomatic of a more fundamental problem, then a correlator is the most appropriate choice. The point at which ECS Correlations are more appropriate over Composer correlators is harder to define. In general, ECS correlations will continue to be a part of complex solutions like managing FrameRelay or MPLS. This is mostly due to it being more general. Complex solutions require that generality even at the expense of more time to develop.

Correlation Composer is expected to be adopted by a wider audience of users as compared to that of ECS Designer. Also the logic of a correlator being developed should fit well into one or a combination of the Composer templates. The Composer templates have encapsulated the common logic usage cases such as transient, rate, etc. If the correlation requires significant logic and state beyond the Composer templates, then it is more of a candidate for an ECS correlation.

Combining Methods

Practical experience in developing event reductions shows a valuable design pattern for any correlator is to combine de-duplication with the correlator. The nature of a correlator is to hold onto an event(s) for some period, do an analysis and then release the events correlated under some root cause. Often the result of using just a correlator produces a repeated pattern of root cause events in the browser, all basically indicating the same problem. Extending the window of time in the correlator can reduce the frequency of these patterns, but this can also slow down the event system by holding onto events.

The better solution in this case is to have the suppressor event (root cause) be de-duplicated. This allows the correlator to release the correlations more frequently and the browser is kept free from noise by having all occurrences of the root cause de-duplicated under the most recent. This type of solution also reduces the net amount of processing required by `pmd` and `ovalarmsrv`. An example of using this technique is with `OV_IF_Intermittent`. This is the root cause event of the `OV_Connector_IntermittentStatus` correlator and it is also de-duplicated.

Available Manuals

Available Manuals

•For more advanced management of ECS Correlations and the Correlation Engine, refer to the:
HP OpenView Event Correlation Services Administrator's Guide

•For information on developing Composer Correlators, refer to:
HP OpenView Correlation Composer's Guide

•For information on configuring existing correlations and correlators as well as de-duplication, refer to:
Managing Your Network with NNM, Event Reduction Chapter

©2004 Hewlett-Packard Development Company, L.P.

The *HP OpenView Event Correlation Services Administrator's Guide* provides complete coverage of the utilities, procedures and issues you need to understand before attempting more advanced management of the ECS environment. This manual is **not** shipped with NNM. It is shipped with the ECS designer kit.

The *HP OpenView Correlation Composer's Guide* describes the operation of the Composer development interface and syntax for built-in function calls. It is available through NNM's online help menu.

Lab Exercises

•Review concepts of event correlation.

 ©2004 Hewlett-Packard Development Company, L.P.

Objective:

Describe the concepts of event correlation and ECS.

Review Questions:

1. What is event correlation?

2. What are the benefits of event correlation?

3. What is ECS?

4. What is an ECS correlation?

5. What are "Flows" and "Streams"?

6. How can you improve on the results from event correlation?

7. How does Composer relate to ECS?

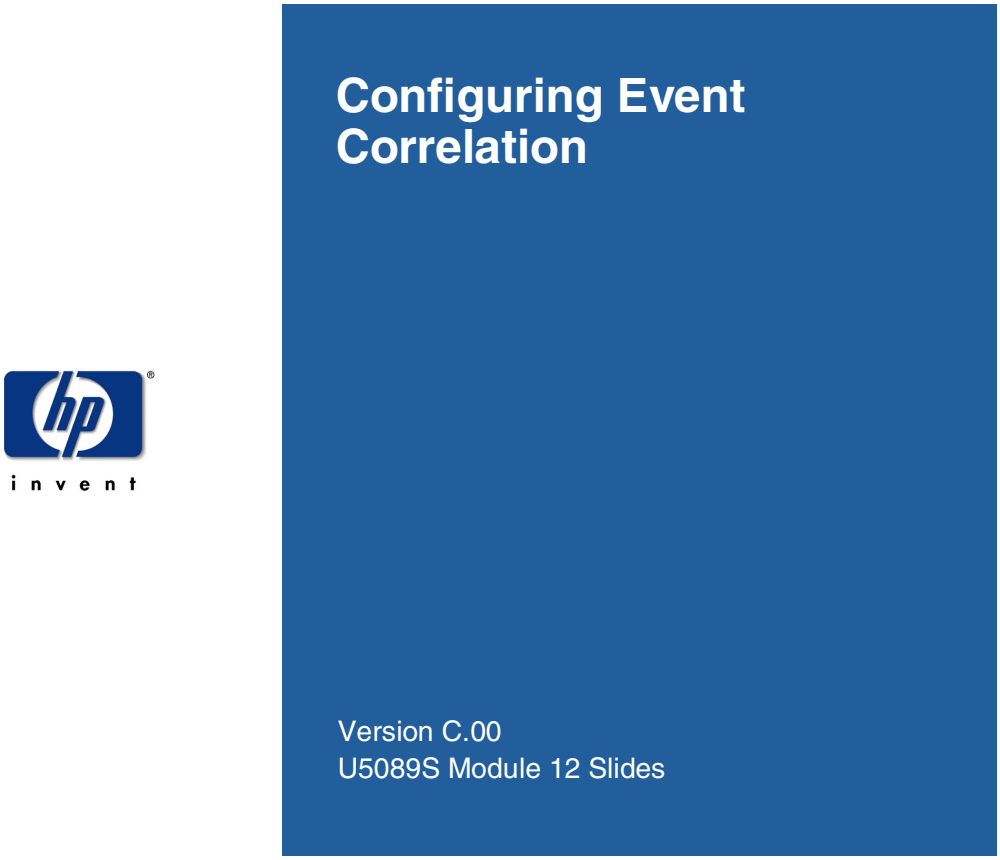
8. How many times does Composer evaluate each incoming event?

9. What qualifying questions would you ask to decide whether a problem was amenable to a Composer solution?

10. List the event reduction mechanism in order of least load imposed on `pmd`.

12 Configuring Event Correlation

Module Objectives



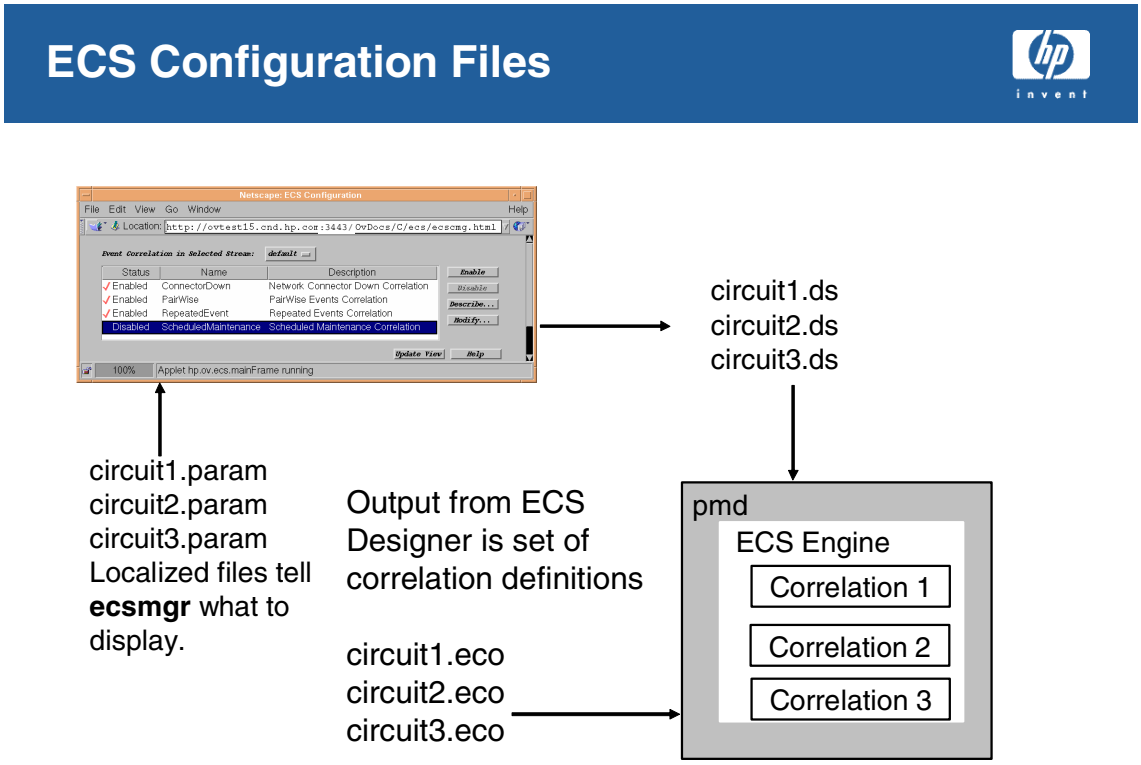
At the completion of this module the student will be able to:

- Describe the supplied correlations and their usefulness.
- Describe the operation of the internal correlators.
- Describe and configure event de-duplication.
- Contrast PairWise correlation with event pattern deletion.
- Contrast event de-duplication with RepeatedEvent correlation.
- Manage, through the ECS Event Configuration GUI, the standard ECS Correlations supplied with HP OpenView NNM.
- Modify correlation parameters to adjust their operation.
- Configure correlators shipped with NNM.

References

You may find additional information in *A Guide to Scaling and Distribution for Network Node Manager*, and in the online manual for ECS in `/opt/OV/docs/ecs`.

ECS Configuration Files

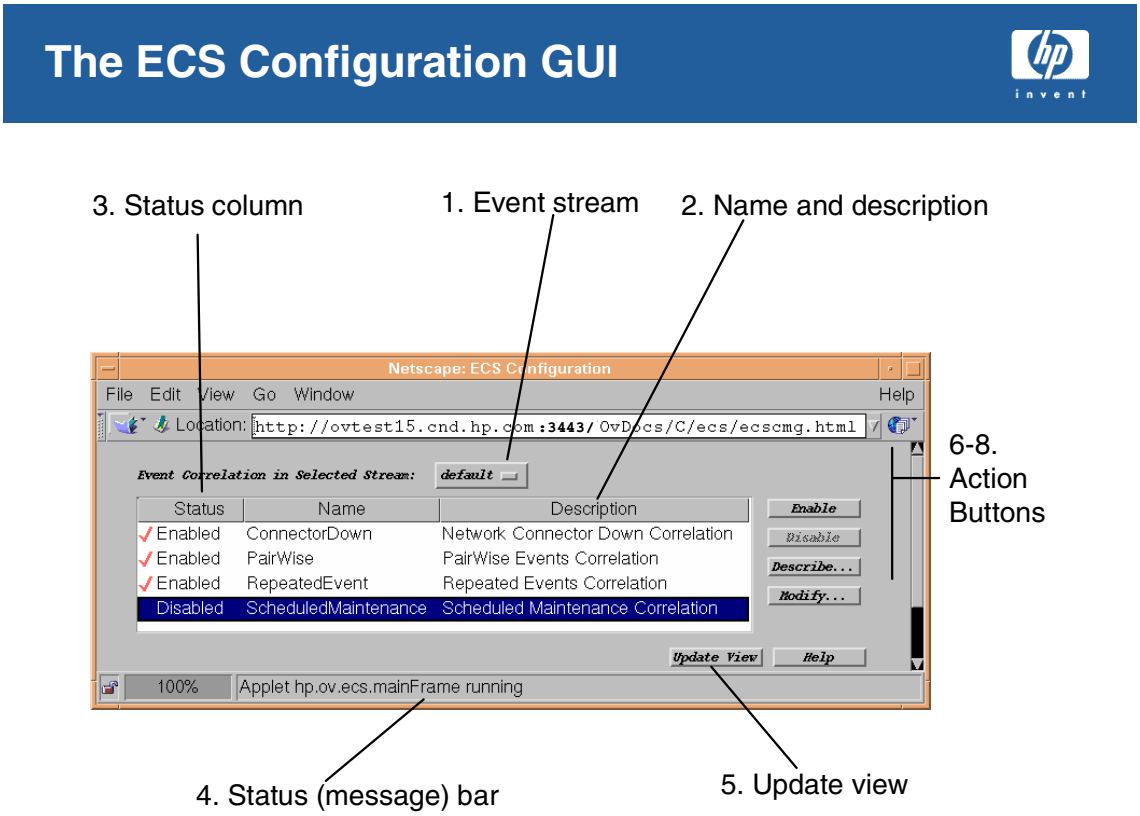


 ©2004 Hewlett-Packard Development Company, L.P.

You configure ECS correlation behavior by setting values in the ECS Configuration GUI. These values are written to data store (.ds) files, which are read by the ECS Engine during operation.

The values in the data store files are localization-neutral, meaning they consist of integers, booleans, or English strings. Localization of the ECS Configuration GUI is done through the parameter description (`.param`) files.

The ECS Event Configuration GUI



The ECS Event Configuration GUI is a Java applet that runs inside an internet browser application. It displays all installed correlations in the local NNM environment. Initiate the GUI from the menu command `Options:Event Configuration`, followed by `Edit:Event Correlation`, or from the tools tab in the Launcher.

All basic event correlation management activities are performed through this interface. The ECS correlation engine manager, `ecsmgr`, is used to alter the correlation's configuration when it is running.

After installation, one stream and several correlations are available.

Screen Description

1. Event stream.

   In the standard HP OpenView NNM installation, there is only one stream, "Default". If other streams have been implemented at your NNM installation, you must select the stream of interest using this button before you proceed.

   The ECS Event Configuration GUI shows all of the correlations that have been installed regardless of the stream that might be selected.

2. Name and Description.

   The Name of the correlation is derived from its installed file name. You cannot change the name. Correlation files are located in:

   Windows: `%OV_CONF%\ecs\circuits\`
   UNIX: `$OV_CONF/ecs/circuits/`

   The Description field usually shows a short description of the correlation. A more complete description should be displayed in the Correlation Description Window when you click the [`Describe`] button.

3. Status column

   Shows a check mark and the word 'Enabled' if the correlation is enabled. Correlations that are not enabled show the word 'Disabled'.

4. Status (message) bar

   Displays messages indicating the success or failure of a configuration attempt. The Status Bar is cleared on the next mouse click or keystroke.

5. Update view.

   If the correlation you want is not displayed, click [`Update View`] to refresh the list. Recent updates made by other users, and updates made at the command line, are not displayed until you refresh the list.

6. Enable/Disable.

   These buttons enable and disable a selected correlation within the context of the selected event stream.

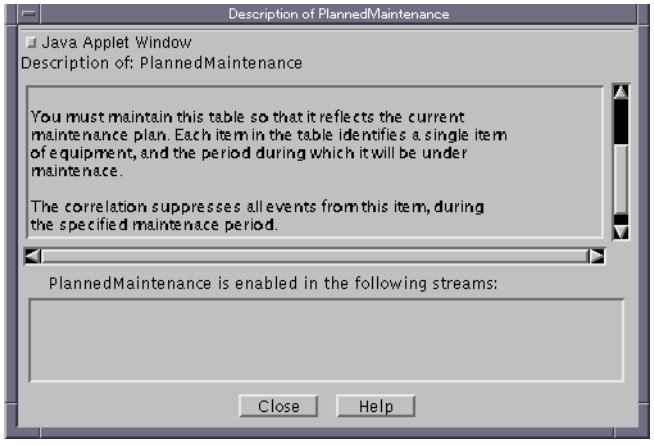
7. Describe

   Click on this button to display a more comprehensive description of the correlation.

8. Modify

   Leads to facilities for modifying a selected correlation.

The Correlation Description Window

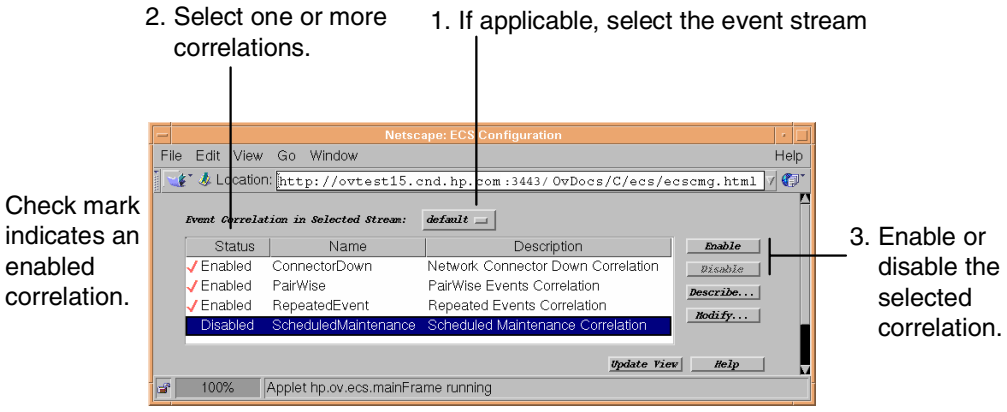


The correlation description window serves as the “reference page” for specific correlations. This window displays:

- The name of the selected correlation at the top of the window.
- A detailed description of the correlation.
- A list of all the event streams in which the correlation is enabled.

Enabling and Disabling Correlations

Enabling and Disabling Correlations



 ©2004 Hewlett-Packard Development Company, L.P.

Each of the correlations can be enabled or disabled. Disabling a correlation may save some processing resources, but loses the advantages of higher information content in the Alarm Browser.

If the correlation you want is not displayed, click [`Update View`] to refresh the list.

Enabling

If the correlation cannot be enabled, an error message will appear in the Status Bar, at the bottom of the window.

As network conditions change, you can modify the parameters of enabled correlations. However, modifying some parameters will disable the correlation and re-enable it when the change is applied. This disrupts the event flow, not just for the brief interval while the correlation is disabled, but also for the time the correlation requires to return to normal operation after it is enabled.

Many correlations work by comparing an incoming event with a selected history of earlier events. When a correlation is enabled it has no history and takes time to build one. Meanwhile, events that would otherwise be suppressed may be passed through, and events that would normally be modified or generated may not be. This is referred to as the settling time.

A given correlation can be enabled only once on any given event stream.

More than one distinct correlation can be enabled on any given event stream.

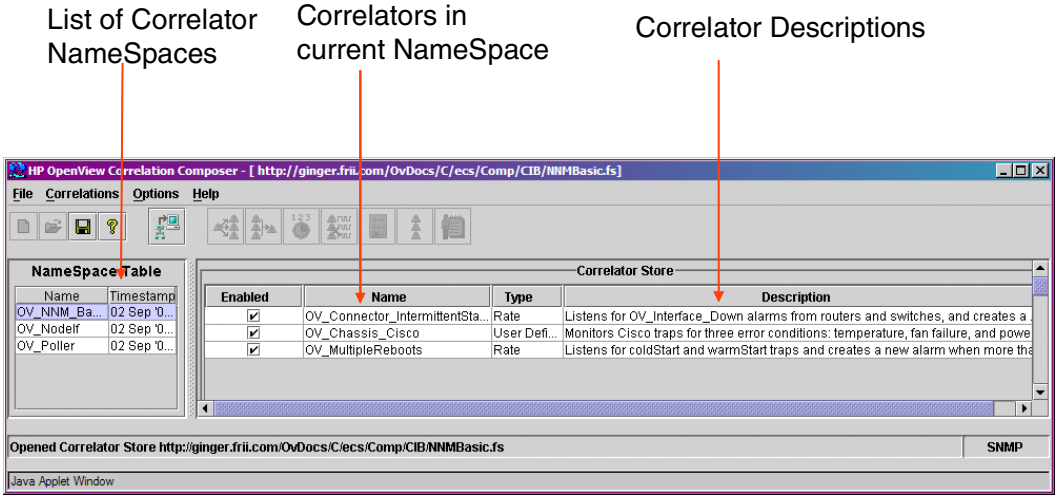
The same correlation can be enabled on more than one event stream.

When the same correlation is enabled on more than one event stream, the same parameter values apply to all instances of the correlation. Also, any changes you make to the parameters for the correlation apply to all associated event streams.

For example, if you have two streams 'A' and 'B' and the ScheduledMaintenance correlation is enabled on both, then if maintenance moves from 8PM to 9PM and you apply this modification to the correlation in stream 'A', then the modification is also applied to the correlation in stream 'B' as well.

Composer: Configuring NNM-Shipped Correlators

Composer: Configuring NNM-Shipped Correlators



Select **`Options:Event Configuration`**.
Select **`Edit:Event Correlation`**.
Select the Composer correlation and click [**`Modify`**].

 ©2004 Hewlett-Packard Development Company, L.P.

Rules which operate inside the Composer correlation are called “correlators” to distinguish them from correlations developed with ECS Designer.

To modify a Correlator:

1. Select the Correlator name from the table in the Correlator Store window.
2. Right click the mouse button and select Modify from the menu or double-click on the correlator. This opens the Correlator window.
3. Make the required changes to parameters. Review the description tab help to see which parameters are configurable. ONLY modify supported parameters.
4. Deploy the changes. Click the deploy icon or select `Correlations:Deploy` to deploy the modified changes.
5. Then select `File:Close` to close the Correlator window and return control to the Correlator Store window.

Internal Event Correlators: NodeIF

Internal Event Correlators: NodeIf

•Internal correlators are used by NNM and have no customer-configurable parameters.

•Displays interface status events

•Node status events are log only

•Displays catastrophic node status events

U5089S ©2004 Hewlett-Packard Development Company, L.P. 6

Internal Correlators are displayed in Composer, but are not configurable. These correlations should function properly out-of-the-box. For detailed information about any of the Internal ECS correlations, see the description in their Composer definition.

- NodeIF-This group of correlators extends the functionality of the ConnectorDown correlation and supports a different event model. In this new event model, the NodeIF correlator displays interface events instead of node events for most interface failures. The NodeIF correlator may display Node events for connector devices during catastrophic situations. In the case of catastrophic situations, a node's interface events are suppressed.
- NNM displays node status events in the Alarm Browser only in catastrophic situations. If one interface of a node goes down, all of the other interfaces associated with that node having an up status are polled immediately to quickly determine if a catastrophic failure exists.
- In cases where you need to disable the NodeIF correlator and configure NNM to display node status events in the Alarm Browser and log interface status events, you should follow a detailed procedure located in the *Managing your Network with Network Node Manager* manual.

For a procedure detailing how to return NNM back to monitoring node status events instead of interface status events, see the *Managing your Network with NNM* manual.

Internal Event Correlator: IntermittentStatus

Internal Event Correlator: IntermittentStatus

- Identifies device interfaces reporting intermittent status
- Works closely with the PairWise correlation
- Generates a OV_Intermittent event when an interface cycles down and up 4 times in 30 minutes

©2004 Hewlett-Packard Development Company, L.P.

If an interface is continuously going down and then going back up again, the PairWise correlation will normally cancel the `OV_IF_Down` event with the `OV_IF_Up` event. You may never see this situation occurring. The `IntermittentStatus` correlator detects this situation and generates an `OV_Intermittent` event when a network interface goes down and comes back up more than 4 times in 30 minutes. This correlator works closely with the PairWise correlation, which allows an `OV_IF_Up` event to cancel an `OV_IF_Down` event.

Internal Event Correlator: Chassis

- Displays Cisco chassis traps as NNM events
  - Fan fault
  - Power supply fault
  - Excessive temperature fault

©2004 Hewlett-Packard Development Company, L.P.

The Chassis Internal Event Correlator converts the Cisco chassis traps, described in the CISCO-STACK-MIB.my MIB, into NNM events which indicate one of the following faults: fan fault, power supply fault, or excessive temperature.

Multiple Reboots Correlator

•Problem
- The Operator does not know if device reboots because coldStart and warmStart traps are LOGONLY.
- Multiple failures in a short time may indicate a hardware problem on the device.

•Solution
- Correlator to create a new OV_Multiple_Reboots event if more than N cold/warmStart traps are received within M minutes
  - You can configure how many and the window period

©2004 Hewlett-Packard Development Company, L.P.

NNM tracks multiple reboots of a network device, which may indicate a problem. For example, a device can reboot if a fan fails or if one of the internal processes aborts. A single reboot is not necessarily a problem. However, if a device fails a couple of times within a short period of time, it may indicate an issue with the device.

Currently, the operator has no way of knowing that a device rebooted because the `coldStart` and `warmStart` traps are suppressed by default. Some device vendors add additional varbinds to the trap. When a MIB with such enhancements is loaded, the trap usually appears in the Alarm Browser.

NNM blocks display of all `coldStart` and `warmStart` traps in the Alarm Browser with a correlator, including. the vendor-specific traps. For multiple reboots, ECS creates a new event if more than N cold/warmStart traps have been received in M minutes (for example, N=4, M=5min).

Once the `MultipleReboot` event has been sent, all further traps for this interval are suppressed.

Configuring the MultipleReboots Correlator

You can modify the number (count) of reboots and the window period for correlator in the ECS Composer GUI.

NNM Supplied Correlations

•NNM provides several correlations:

- Network Connector Down Correlation
- Pairwise Events Correlation
- Repeated Events Correlation
- Scheduled Maintenance Correlation
- ManageX Server Down Correlation
- Composer Correlation

 ©2004 Hewlett-Packard Development Company, L.P.

NNM provides several correlations: Connector Down, Pair Wise, Repeated Events, Scheduled Maintenance, and ManageX Server Down correlations.

Connector Down: Use of this correlation can prevent event storms when a router or other connective device goes down. A secondary failure is a node which cannot be reached, not because it may be down, but rather because the route to the node contains a down link or node. Without the correlation for example, if a router is down, all nodes beyond that router would show as being unreachable, and thus down, when, in fact, their status is unknown.

Pairwise Correlation: This correlation matches an event to one or more previous events that it 'cancels out'. For example, if a NodeDown event is followed by a NodeUp event from the same source, within a given period of time (see the PairedTimeWindow parameter), then the NodeUp event 'cancels' the NodeDown event.

Repeated Events: This correlation allows multiple alarms which are associated with a single physical event to be bundled together and replaced by a single alarm in the Alarm Browser.

Scheduled Maintenance: Computer nodes and network segments are occasionally scheduled for maintenance. Such an occurrence typically results in a number of SNMP traps or events being generated. The Scheduled-Maintenance correlation prevents unneccessary alarms during maintenance.

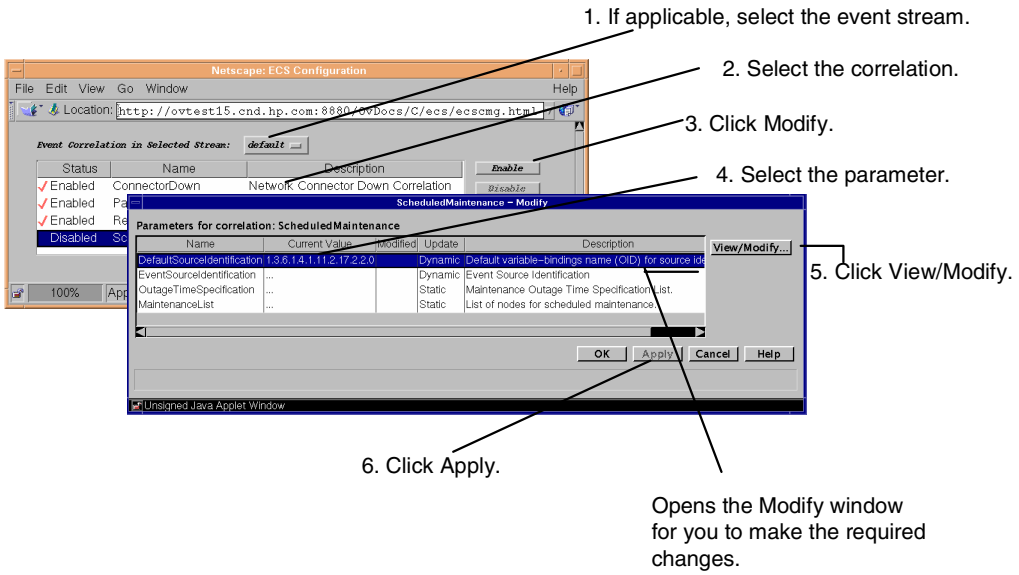
The **ManageX Server Down** (MgXServerDown) correlation provides useful information about devices that are managed by HP OpenView ManageX and about network connectivity. You would

only enable this in a ManageX environment.

This correlation builds on the NNM Connector Down correlation by using root cause analysis to determine the primary reason for network connectivity problems relating to devices that are managed by ManageX. You can use this correlation determine when a ManageX server is down. In addition, you can distinguish whether the root cause of a network failure is a device that is managed by ManageX or a network connector device (for example, a router or a switch).

Modifying Event Correlations

Modifying Event Correlations



 ©2004 Hewlett-Packard Development Company, L.P.

You use the ECS Configuration GUI to set parameter values for the correlations to read from their datastore (.ds) files. These parameters include things like which fields of the events to match on, how long to continue looking, and how to present the results in the Alarm Browser.

For each correlation you want to configure, step through the parameter fields listed as rows in the table. For each parameter, the GUI presents a data entry table specific to that data type.

ECS defines some parameters as **dynamic** and accepts changes immediately, whereas others are defined as **static** and can only be incorporated when ECS unloads and reloads the correlation and builds a new history. Unloading and reloading a correlation can be disruptive to processing. You can use the “Update” column to determine how your changes will be handled.

Parameters	**Enabled**	**Disabled**
Static	Event correlation is disrupted. Correlation: • is reloaded with new parameter values. • may need to reload information from disk. • may need to build new history of events.	Changes accepted immediately.
Dynamic	Changes are accepted and put to immediate use.	Changes accepted immediately.

There is no log file which logs user changes to the correlations.

In general:

- Some correlations have no parameters.
- You cannot change the data type of a parameter or table cell.
- Some parameters have a default value. The description may show the default value and the correct syntax to be used.
- Many parameters have limits to ensure you set reasonable values.
- All parameters are checked for correct syntax for their data type before acceptance.
- For information on individual correlations, see the ECS online help.
- Fields that cannot be modified are grayed out.

Recommendations for Modifying Correlations

Correlation parameters can be changed at any time, even while the correlation is enabled. Observing these recommendations will minimize disruption to the correlation service and any impact on other network administrators.

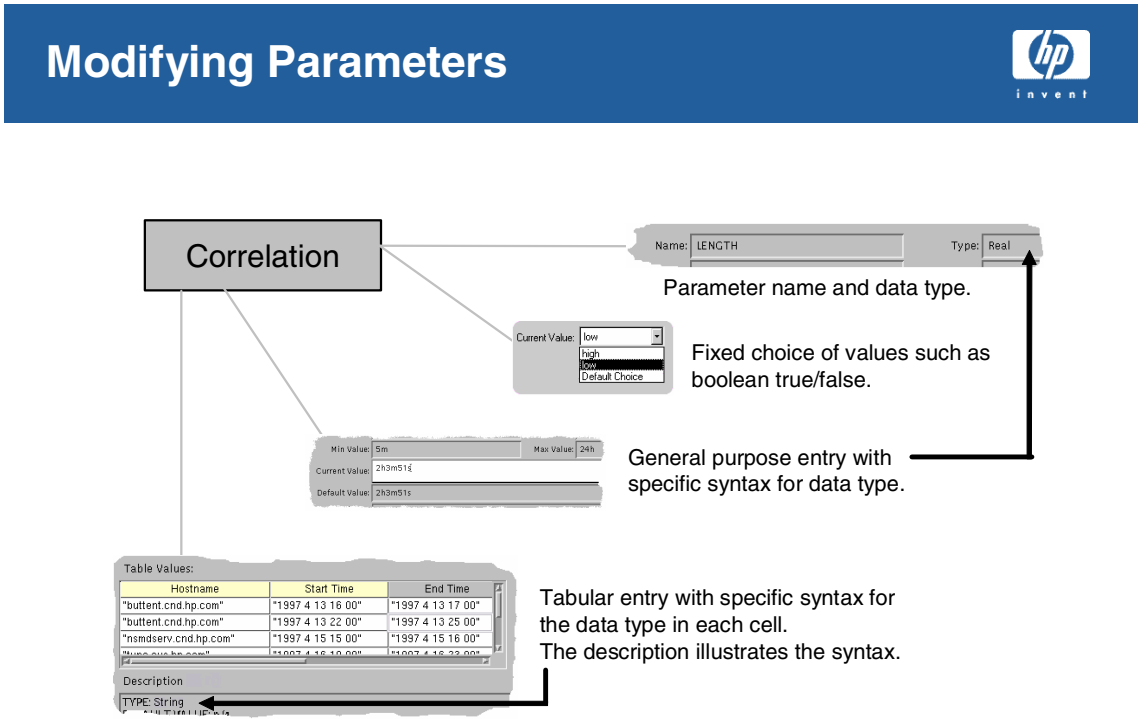
In the parameter modification window of the ECS Configuration GUI, the parameter is indicated as static or dynamic.

- Plan changes ahead of time.
- Choose a time when network activity is low.
- Do not make ad hoc changes to static parameters.
- Apply consistent parameter modifications together in one batch.
- Avoid changing static parameters in the middle of an event storm.
- Be aware of the effect of enabling, disabling or modifying a correlation beforehand.
- Ensure that all network administrators know which correlations are in effect.

NOTE Do not apply modifications individually. Make all the required modifications first and then apply them as a “batch”. This avoids possible undesirable consequences due to the correlations operating with mismatched parameter values.

Do not modify the parameters for any correlation that is currently enabled in a “production” stream without carefully assessing the possible consequences of your actions. If your site has implemented multiple streams, any change you make to a correlation in one stream will affect all other streams in which the correlation is enabled.

Modifying Parameters



The three variants of the *Modify* window have similar parameter name and type fields and different styles of text entry area, depending on the parameter type.

The entry area styles are:

1. Fixed choice.

   Choose from a drop down list of predetermined values.

2. General purpose entry.

   Enter a string of characters to define the value and be certain to use the correct character syntax for the associated data type.

3. Tabular entry.

   The data type for the parameter in this case is always “dictionary” but the data type for values in individual table cells will come from a range of data types. When entering values into cells, use the correct syntax for the associated data type.

Tables have the following characteristics:

- Columns are predefined for a given table.
- One or more columns will be data “keys” for the row.
- Rows can be deleted or added at the foot of the table.

To change a value in a table, click in the table cell, click in the data entry box and enter your new value, then click back in the table cell.

Simple Data Types

ECS correlations use several data types for data representation. This slide discusses only simple data types although there are others not covered. Some of these data types can be more complex than the representation here.

Integers

- Are whole numbers.
- Are represented as a string of digits. Numbers without a leading + sign are assumed to be positive. Commas are not allowed.
- For example: 21 +47 -4759

Reals

- Are expressed with a decimal point, and may include an exponent.
- They range from -2.2250738585072014E308 to 1.7976931348623157E308 and the smallest exponent is -308.
- For example: 21.0 -629.4 2.34E-7

Boolean

- Are either true or false. No other value is allowed. The GUI converts any input to lower case.

Duration

- Is used to store relative times. A relative time is the elapsed time between two absolute time points as represented by the Time data type.
- Is a signed data type with a resolution of 1 microsecond and a range of approximately +/- 596,563 hours (68 years).
- Is expressed in hours, minutes and/or seconds. It can consist of any or all of these parts. Each part is followed by one of the letters **h**, **m**, or **s**, as appropriate. There must not any spaces in the value.
- For example: 13.5h 13h30m 44s

Time

- Is used to represent the absolute time of day on a particular date.
- Has a resolution of 1 microsecond and can represent any time from 00h 00m 00s, 1 January 1970 (UTC) up to the year 2038.
- Values are represented in this format: *yyyymmddhhiiss.uuuuuuZ*
- For example, the date/time value January 3, 1997 at 1:59:59.123456 PM (UTC) would be: 19970103135959.123456Z

OID

- Is used to represent Object Identifiers.
- Is a sequence of integers separated by periods.
- For example: 1.43.67.52
- An OID value must contain at least two dots. If a value has only one, it will be interpreted as a real.

String

- Is an array of multi-byte characters. Each character is a value that represents a character in the current character encoding.
- Is entered as a sequence of characters enclosed by double quotes.
- For example:“notificationIdentifier”
  “A newline starts after this \n”

  Two quotes typed together (““) represent an empty string.

Token

- Is a set of discrete alternatives. A token is one value from the set.

netmon Accelerated Polling

- If one interface on a connector goes down or up, move any other interfaces to the front of the polling queue.
- If an interface goes from Up to Unknown (secondary failure), reschedule for “soon” (“soon” = “now”+30 seconds)
- The move to the front of queue is inhibited if the interface is flapping over an extended period.

©2004 Hewlett-Packard Development Company, L.P.

Ordinarily interfaces on a router or switch are scheduled to be polled throughout the configured polling cycle for the device. However, if the router fails, `netmon` might discover one non-responding interface immediately, but on the average would not discover the full set until the entire polling interval had expired. The failure of the entire device would not be reported until the last interface had failed to respond.

NNM avoids this situation scheduling all the OTHER (up) interfaces on the node where one interface has gone down, to be polled immediately. As a result, the status of all the interfaces on a failing connector device are determined and the final node status event is emitted very quickly. Typically this could occur in 30 seconds instead of 15 or 20 minutes. This polling procedure also work for Down to Up transitions, which are scheduled for repolling in 2 minutes.

In addition, when an interface goes from Up to Unknown (which happens for secondary failures), `netmon` reschedules the other interfaces Soon (now+30 seconds) rather than Now. The time slot of Now is used for Primary Failures. If an interface is scheduled to be polled anyway sooner than now+30 seconds, then the poll time of the interface is not adjusted.

Polling Performance Improvement for Flapping Interfaces

If an interface is toggling from up to down repeatedly due to some sort of hardware problem, this

toggling situation can be detected quickly. This facilitates the operation of the IntermittentStatus correlation logic and allows it to successfully report these situations quickly and reliably.

False Downs: Interfaces which are operational and accessible to certain protocols may be reported down by `netmon` due to overloaded connector devices that have been configured to treat ICMP or SNMP as a low priority. In situations of high load (for example, through the router) the connector device will drop ICMP packets, which causes `netmon` to time-out waiting for the ICMP response. When the network administrator tries to test the connection using some other protocol (for example, http) or using ping a short time later, the interface appears up. This is confusing if NNM does not show the interface up until the next polling cycle, perhaps 15 minutes later. In the meantime, an event may have been displayed on the Alarm Browser.

When an interface goes from up to down at time T0, the next two polls of this interface are scheduled for 2 and 4 minutes from T0 respectively. If the overload condition is temporary, then the quick re-poll may catch the interface up and bring the status back to up in time for the ECS PairWise correlation to catch the pair of events and cancel them. In addition, this partially decouples the status polling interval configuration from the PairWise correlation's PairedTimeWindow parameter and allows the overall system to function properly with small timing configuration imperfections.

NNM inhibits this behavior if flapping continues over an extended period. `netmon` tracks the number of status changes internally and if an internal threshold is exceeded, disables the “repoll this interface in 2 minutes” feature for approximately 24 hours. To change the default values, use the following `netmon` options:

- `-kcomputeAccessPort=true`

  By default `netmon` tries to determine which switch ports are connected only to end nodes (so-called access ports), and sets interface flags accordingly. If you set the `computeAccessPort` keyword to false, you disable this behavior.

- `-kuseAccessPort=true`

  By default, `netmon` uses switch access port information for enhanced identification of primary failures. Setting `useAccessPort` to false disables this behavior. When this feature is enabled, `netmon` can better identify end-nodes as the primary failure causing a switch port to go down. If you assign a value of false to either the `computeAccessPort` keyword or the `useAccessPort` keyword, `netmon` does not use access port information to compute primary failures.

- `-kscheduleChassisIfsImmediate=true`

  If true, when an interface goes down on a switch or router, all of the remaining interfaces which are not down or unknown are scheduled for an immediate poll. Similarly, if an interface goes up, all interfaces on the connector device that are not already up, are rescheduled for an immediate poll.

- `-kshortPollUpTime=120`

- `-kshortPollTime=120`

  If the `-kshortPollUpCount` or `-kshortPollDownCount` options are utilized (non-zero), then a poll of the interface occurs at time <*shortPollTime*> seconds or the regularly scheduled poll time, whichever time occurs soonest. This option is used to accelerate when the next poll will occur for interfaces that are changing state. Defaults to 120 seconds.

- `-kshortPollUpCount=1`

  If an interface goes up it may be due to a flapping or intermittent network interface. So repoll it ahead of schedule <*shortPollUpCount*> number of times to help detect if the interface has gone back down. Defaults to 1.

- `-kshortPollDownCount=1`

  If an interface goes down it may be due to network congestion. So repoll it ahead of schedule <*shortPollDownCount*> number of times to see if it's up. Defaults to 2.

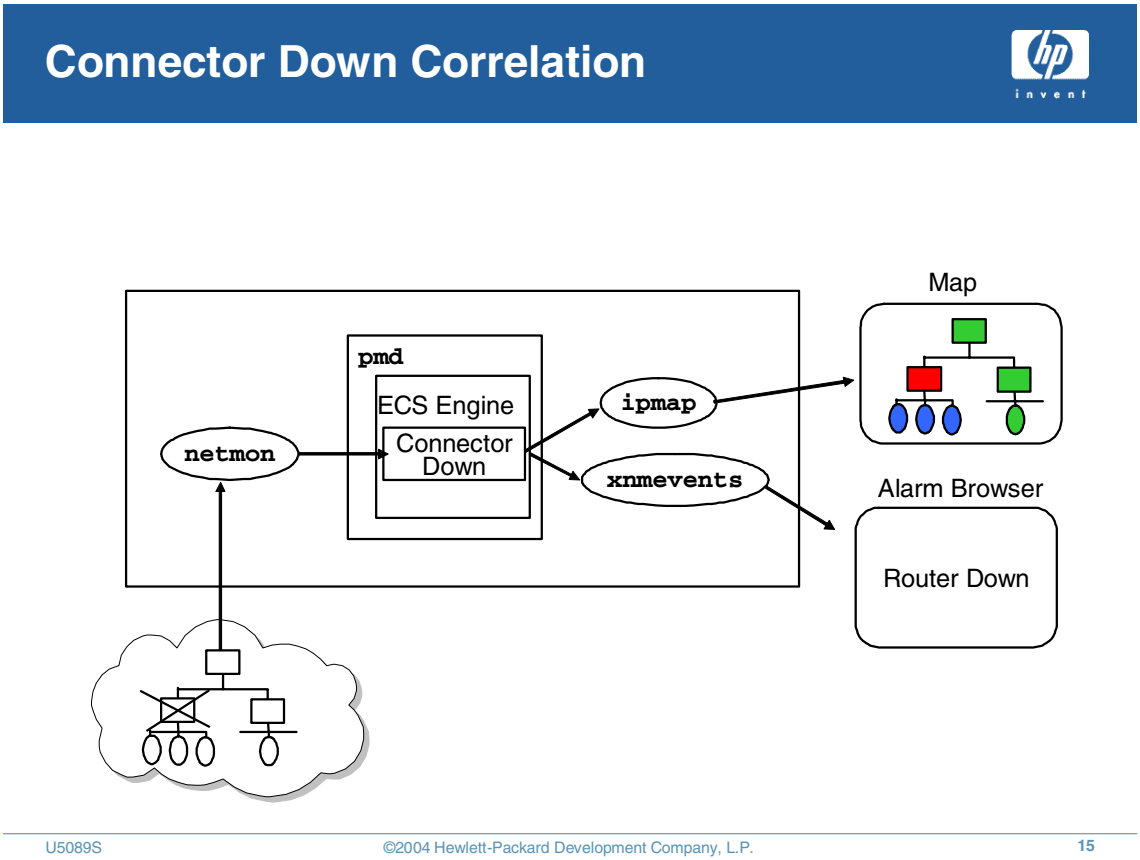
- `-kflapTriggerCount=40`

- `-kflapRearmCount=300`

  If `-kshortPollDownTime` option is utilized, then flapping or intermittent interfaces can consume a lot of `netmon`'s bandwidth. This is necessary to facilitate detection of this condition in the network. However, after netmon has detected the situation, performance can be preserved by inhibiting the special attention provided by the short poll facility.

  After an interface changes state <*flapTriggerCount*> in consecutive polls, the short poll feature is disabled for <*flapRearmCount*> polls.

Connector Down Correlation



Use of this correlation can prevent event storms when a router or other connective device goes down. In the event of a network failure, NNM works in conjunction with the Event Correlation Services to automatically determine:

- Which element is malfunctioning. NNM automatically determines the route to a node that is down and checks each connective device along the path to that node to determine the **primary** malfunction.
- Which other network elements are impacted by this failure. That is, which functional network elements are now inaccessible over the network because of the failing device.
- Which inaccessible network elements are important to the productivity of the organization and thus should be given high priority.

Only the error message from the primary device is logged to the Alarm Browser, making it easy for you to determine the root cause of the problem. For a list of secondary nodes which are unreachable, double click on the parent alarm in the Alarm Browser.

A **secondary failure** is a node which cannot be reached, not because it may be down, but rather because the route to the node contains a down link or node. Without the correlation for example, if a router is down, all nodes beyond that router would show as being unreachable, and thus down, when, in fact, their status is unknown.

NNM allows you to indicate what NNM should do in the case of secondary failure.

Common Use Models

This correlation is commonly used in one of the following configurations:

1. **As shipped:** The object which has failed is noted as Critical. Nodes beyond the failed connector are marked Unknown.
   - Failed Connector: Red on map, alarm appears with “Cor” checked.
   - Important Nodes: None defined.
   - Secondary Failures: Blue on map, alarms suppressed from Alarm Browser and correlated under the primary failure.
2. **Important Node:** Identical to the shipped configuration, with the addition of one or more “important” nodes, identified by a filter, which are also noted as Critical.
   - Failed Connector: Red on map, alarm appears with “Cor” checked.
   - Important Nodes: Red on map, alarm appears. The important node(s) also appear in the secondary list when you “drill down” from the correlated alarm.
   - Secondary Failures: Blue on map, alarms suppressed from Alarm Browser and correlated under the primary failure.
3. **High Performance:** Rather than update all the NNM databases with the information that the objects beyond the failed connector are “unknown,” just leave them “Unchanged” on the map. Internally, NNM does not get messages to change every node’s records. `netmon` does use the reduction multiplier for secondary failure nodes.
   - Failed Connector: Red on map, alarm appears with “Cor” checked.
   - Important Nodes: Blue on map, alarm appears. They also appear when you “drill down” from the correlated alarm.
   - Secondary Failures: Green on map, no alarms. They do not appear in the secondary list when you show children of the correlated alarm.

Configuring Secondary Failures

Secondary Failure Polling Options

- For cases where “in-between” devices interrupt polling:
- Use Secondary Status polling options
- Set multipliers and suppression of alarms
- Define the list of important nodes
- Define what to do about other nodes

©2004 Hewlett-Packard Development Company, L.P.

NOTE Since this correlation is so tightly tied to `netmon`, its configuration is done through `Options:Network Polling Configuration` rather than `Options:Event Configuration`. Do not attempt to configure ConnectorDown or secondary failures through the ECS Configuration GUI interface.

To set the parameters:

1. Select the NNM `Options:Network Polling Configuration:IP` menu, rather than the ECS Configuration GUI.
2. On Windows, select the `Secondary Failures` tab. On UNIX, click the [`Configuration Area`] button and select `Secondary Failures`.

- The Secondary failures polling options checkbox enables or disables operation of the correlation.
- Status polling of secondary failure nodes can be reduced by a multiplier. The default multiplier is 2. If the normal status polling interval were 10 minutes, this multiplier would make the polling interval 20 minutes instead.

  To maintain status information, `netmon` polls each node in the topology at the polling interval. The response time for a successful poll is approximately 1 millisecond. However, `netmon` may have to wait 10 seconds to determine that a node is not responding. This increased wait time

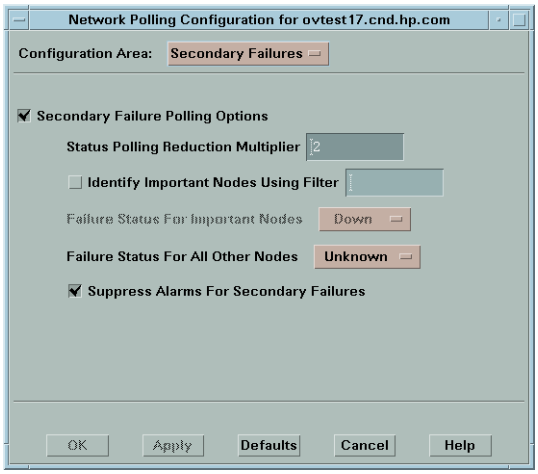
can put `netmon` behind in its polling cycle for nodes that *are* reachable. By reducing the polling for nodes known to be unreachable, you keep netmon polling the rest of the network in a timely manner.

- You can identify important nodes using a filter. This filter is used in conjunction with the status box.

  You might want some nodes to be specified as down because of their importance, such as a file server or database server, while other nodes can take on the status of unknown, such as individual workstations. By identifying these nodes, you can tell NNM to leave them red among all the blue nodes on the map and to display the node down message in the Alarm Browser rather than suppress it.

- You can specify that secondary failures should not generate alarms. Since the status of the nodes beyond the primary failure is unknown, there is generally not a critical need to have alarms on all the secondary failures. This keeps your Alarm Browser display clean so that the real problem shows clearly.

Configuring Secondary Failures on UNIX



Configuring Secondary Failures from the Command Line

`xnmpolling` is the command line version of the configuration application.

All configuration options are available for purposes such as changing options via automated scripts, at installation time, etc.

The parameters are described in the online help reference page (or manpage on UNIX). If no parameters are specified, the command displays the dialog box. You may use the command line instead of using the `Options:Network Polling Configuration:IP` menu selection.

Command Line Example

To reduce the amount of status polling for secondary failures by a factor of 10:

```
xnmpolling -secFailPollScale 10
```

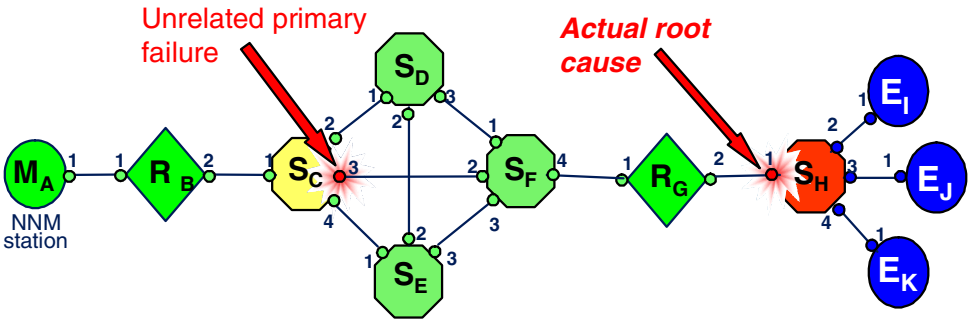
In this example, when an interface is identified as having a secondary failure, if the scaling factor is 10 and the configured status polling interval for the interface is 5 minutes, the scaled status polling interval would be 50 minutes.

Critical Routes

The `-c critRouteSeedFile` option in `netmon.lrf` causes `netmon` to load the specified file as authoritative entry points into networks in the topology for purposes of determining the critical route to nodes and their interfaces. The critical route is used to distinguish between primary failures (due to nodes and interfaces themselves being unavailable) and secondary failures (nodes and interfaces not being accessible as a result of primary failures) in the ConnectorDown ECS correlation. You can use the contributed program `makeCrtiRouteSeed.ovpl` to assist in creating the seed file for remote networks.

ConnectorDown Correlation with NNM Extended Topology

Connector-Down Correlation with Extended Topology



Path M_A to E_I calculated by NNM *without* Extended Topology:

A.1 B.1 B.2 C.1 **C.3** **F.2** F.4 G.1 G.2 H.1 H.2 I.1

Path M_A to E_I calculated by NNM *with* Extended Topology:

A.1 B.1 B.2 C.1

MESH(C.2 D.1 C.3 F.2 C.4 E.1 D.2 E.2 E.2 D.2 D.3 F.1 E.3 F.3)

F.4 G.1 **G.2** **H.1** H.2 I.1

©2004 Hewlett-Packard Development Company, L.P.

Many switched environments have redundancy, or meshing.

NNM Alone: First Alarm is Cause

In this diagram, there are interfaces down on switches “C” and “H” as marked. As a result, the NNM management station “A” has a node-down alarm for switch “H”, and secondary failure on end-nodes “I”, “J”, and “K”.

Note that while interface “C.3” is down, it is not the cause of the connectivity failure. That's because of the path redundancy in the mesh formed by switches “C”, “D”, “E”, and “F”.

The actual root cause of the connectivity failure with switch “H” and all nodes beyond it is the failure of the interface “H.1”. That is the problem, not the failure of interface “C.3”.

However, NNM without Extended Topology is *unaware of the mesh*. The path it calculates could incorrectly include interface “C.3”, and it could consequently derive an incorrect Root Cause for the problem.

With Extended Topology: Root Cause

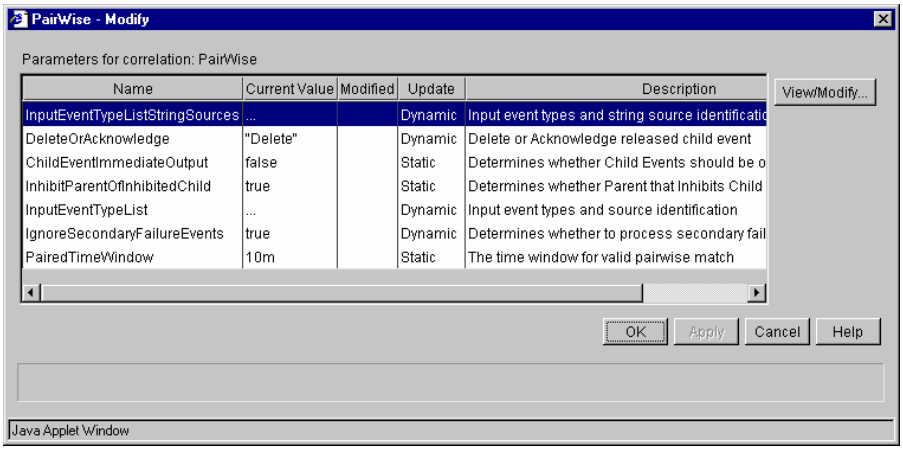
NNM with Extended Topology, on the other hand, understands the mesh, and correctly sees that the root cause is the interface failure on switch “H”.

The alarm for C.3 has no other events correlated under it. The events for the end nodes being down are correlated under H.1.

PairWise Event Correlation

PairWise Event Correlation

- Watch for a matching event that “cancels out” the first one.
- From the same source, within your specified time window.
- Configure types of events, how long to watch, whether to delete the children or just acknowledge them.



 ©2004 Hewlett-Packard Development Company, L.P.

This correlation matches an event to one or more previous events that it “cancels out”. For example, if a NodeDown event is followed by a NodeUp event from the same source, within a given period of time (see the PairedTimeWindow parameter), then the NodeUp event “cancels” the NodeDown event.

In the PairWise correlation, the event that “cancels” the previous event or events is termed the “parent” event (NodeUp). The event that has been “canceled” is termed the “child” event (NodeDown).

A parent event may “cancel” multiple child events of either the same or different pairwise correlations. See the InputEventTypeList parameter for more details.

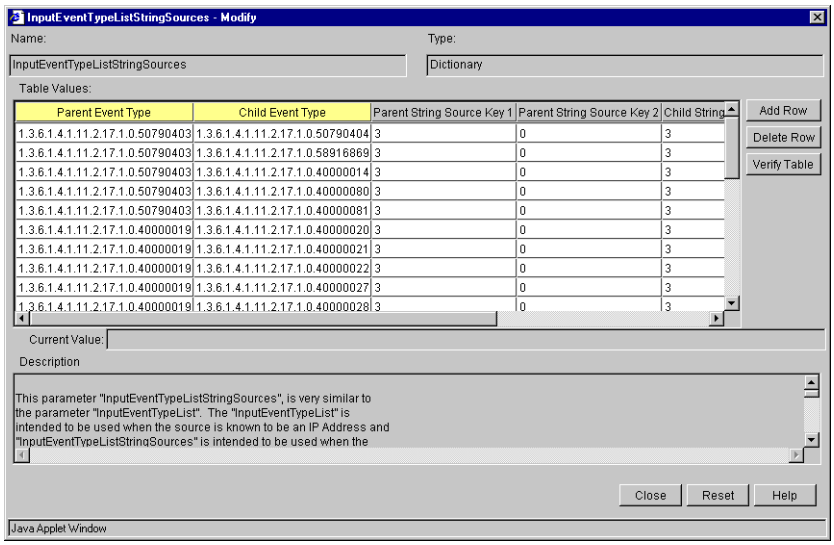
Events that have been configured in the PairWise correlation as possible child events are output immediately. If a matching parent event occurs from the same source, within the appropriate amount of time, it “cancels” the child event. To “cancel” means if the child event is visible in the Alarm Browser, the PairWise correlation either deletes or acknowledges the alarm (see the DeleteOrAcknowledge parameter).

Any child events that are “canceled” by a parent event are correlated under that parent event in the Alarm Browser.

Although this correlation is enabled by default, if you find that none of the parameter settings provide a useful service within your particular network environment, you may disable it to avoid unnecessary system overhead.

Configurable aspects of this correlation are:

- `InputEventTypeListStringSources` determines what types of events and from what sources to attempt to match. NNM ships with many pairs preconfigured for you to use.



  The Parent and Child event types may be identified as either an OID (for specific traps) or as a string (for generic traps).

  NNM looks at the Event ID to determine whether the event is one to attempt to correlate. For a list of event types, see the Event Configuration list in `ovw`. You can sort the OpenView Enterprise Event Identification list by event name to find the events of interest to you.

  The Parent and Child Source Identification columns define which variable binding (varbind) within the event contains the source of the event. The source can be identified by:

  — OID: the correlation searches the event's varbinds until it finds the varbind whose OID matches. The value of that varbind is the event source.

  — Integer: the correlation uses the value of the varbind at this position. The integer represents the first varbind.

  — "agent-addr": the correlation uses the IP address portion of the "agent-addr" attribute in the trap.

  You can concatenate two fields (Column 3 and Column 4, then Column 5 and Column 6) to create a unique source identifier.

  The `Accept` column allows you to turn a defined set of input events on (true) or off (false) without deleting or reentering rows in the table.

  The `Description` field in the Modify box for `InputEventTypeListStringSources` is very helpful.

- `DeleteOrAcknowledge` determines whether a child event (that has already been output) that has been successfully matched against a parent event, is Deleted or Acknowledged from the Alarm Browser.

  A value of "Delete" will remove the event from the browser, a value of "Acknowledge" will not remove it, but simply acknowledge it. In either case, the child event will be correlated under the parent event.

- `ChildEventImmediateOutput` determines whether potential child events should be output to the Alarm Browser immediately, and later acknowledged or deleted if a matching parent

arrives, or if potential child events should be held for the duration of the `PairedTimeWindow` to see if a matching parent event arrives to suppress the child event.

NOTE The default value of the `ChildEventImmediateOutput` parameter (`true`) allows the child events to be output as they occur. Choosing `false` for this parameter causes a delay in the display of child events in the Alarm Browser.

- `InhibitParentofInhibitedChild` determines whether parent events that have successfully inhibited child events should also be suppressed. Enabling this only works if `ChildEventImmediateOutput` is false.
- `InputEventTypeList` is similar to `InputEventTypeListStringSources`, and can be used when the source is known to be an IP address.
- `IgnoreSecondaryFailureEvents` determines whether configured input events which ConnectorDown determines to be secondary failures should be processed by this correlation.
- `PairedTimeWindow` is the maximum duration after the child event that a parent event can "cancel" the child event. For example, the default value of 10m means that the parent event can "cancel" the child event up to 10 minutes after the child event occurred.

Pattern Delete

- Provides a way to delete all stale child events for a given parent event type
- Changes to PairWise correlation
  - Generate pattern delete request when a parent is received
- Changes to ovalarmsrv
  - Receive pattern delete request
  - Find matching events in the alarm database
  - Tells ECS to correlate the events
- Alarm browsers update their displays

©2004 Hewlett-Packard Development Company, L.P.

The PairWise correlation clears the Alarm Browsers of transient situations if the matching event arrives within a certain time window. However, if the matching event arrives later, the Alarm Browser still shows the original alarm. For example, if a node goes down and comes up 4 hours later, the browser still shows the NodeDown event. The operator is left with questions as to the current status of the network -- is the node still down, or has it been repaired in the intervening time?

To enable the Alarm Browsers to reflect the current state of the network, NNM removes from the browser all alarms obsoleted by a new event. For example, a NodeUp for a system causes NNM to remove all previous NodeDown/Warning/Minor/Major for that system. This leaves the Alarm Browser scrubbed clean of all misleading alarms.

The PairWise correlation retains time window values consistent with your Service Level Agreements for repairing outages. Increasing the time windows would have caused ECS to hold all the events longer, taking additional RAM and CPU resources, and would have ruined PairWise's representation of your SLAs. You can review the correlated events to see the timestamps to determine how long the outage lasted.

You do not need to configure parameters for Pattern Delete. The event and system identifiers are taken from the PairWise configuration in the ECS GUI automatically.

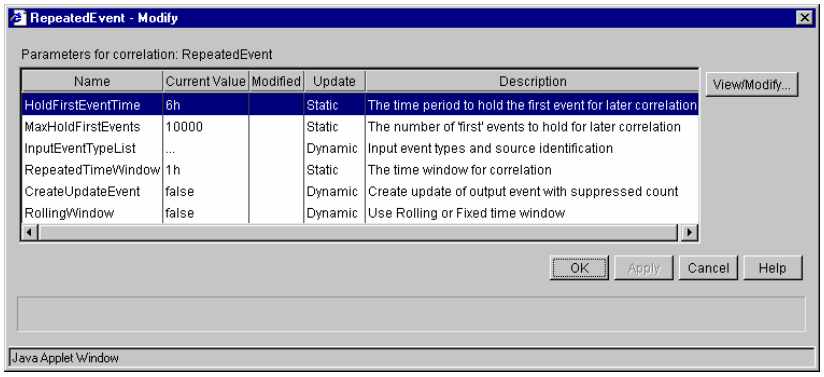
Feature Operation and Design

When `ovalarmsrv` receives an event, it compares it with the list of event signatures for PairWise. If the event would have cancelled out one or more previous alarms, `ovalarmsrv` sends notification to `pmd` of the specific alarms that would have been cancelled by this event. `pmd` logs the correlation in the Binary Event Store and cycles the notification to `ovalarmsrv` and `xnmevents`. The browsers turn on the correlation count indicator and remove the obsolete alarms from view.

Repeated Events Correlation

Repeated Events Correlation

- Multiple instances of the same type of event appear as one alarm.
- Works for events from the same address or multiple addresses.
- Which events to correlate
- Repeated time window
- Whether or not to delete the original alarm from the Alarm Browser window when the time window is exceeded.



 ©2004 Hewlett-Packard Development Company, L.P.

This correlation allows multiple alarms which are associated with a single physical event to be bundled together and replaced by a single alarm in the Alarm Browser. For example, when a collection station synchronizes with a management station, the management station may have thousands of Node Added events. These can all be correlated under the first one received so that only one alarm appears in the Alarm Browser.

NNM determines the event type for each event that is received. In the Alarm Browser, all events with the same event type are correlated under the first trap received for the specified time period.

1. To set the parameters, use the `Options:Event Configuration` menu or run `xnmtrap` from the command line.
2. Select `Edit:Event Correlation` from the menu bar. This starts your configured browser to run the ECS manager Java applet.
3. Select `RepeatedEvent` and click [`Modify...`].
4. Select the area to modify and click [`View/Modify...`].

   The parameters you can set are:

   - `HoldFirstEventTime` determines how long additional rolling events are correlated under the first instance when the Rolling Time Window is true. Increasing this number may adversely affect `pmd` performance. Refer to the Description field for the parameter for more information.

- `MaxHoldFirstEvents` determines the maximum number of simultaneous, unique source keys correlated upon. Increasing this number may adversely affect `pmd` performance. Refer to the Description field for more information.
- `InputEventTypeList`: What types of events to analyze for repetition.

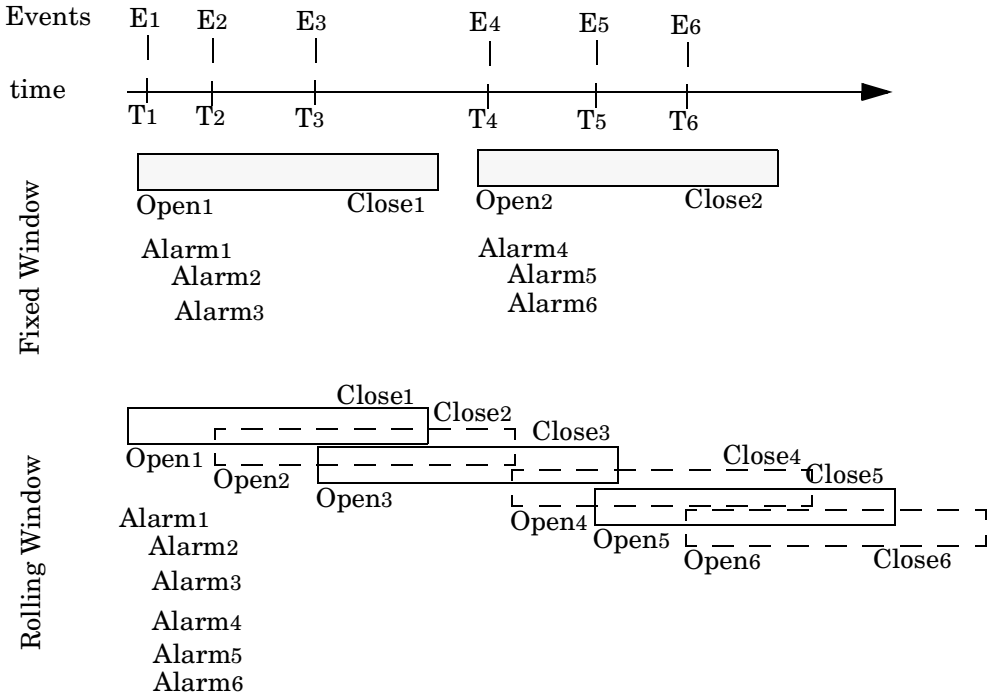
  This works similarly to the PairWise correlation configuration, except that it is a single event, not a pair. It allows you to tell NNM which types of events to attempt to match, based on the value of the specified varbind within the event record.
- `RepeatedTimeWindow`: Amount of time to continue watching for repetitions.
- `CreateUpdateEvent`: Create a new parent event with repeated events as children.

  You can either allow NNM to use the first event as the parent, or tell it to create a new event to be the parent in the Alarm Browser.

  Setting this parameter to "true" causes NNM to create an update summary event and delete the original event from the Alarm Browser. The original event is correlated under the summary event along with the suppressed repeated events.
- `RollingWindow`: Whether to restart time window with each added repetition.

  For example, if a synchronization with a management station takes more than an hour and your RepeatedTimeWindow is one hour, the first Node Added appears in your Alarm Browser. Then each Node Added that occurs within an hour from that first one is correlated under it. The first Node Added event to occur after the hour has elapsed appears in the Alarm Browser as a new parent, with the next hour's worth correlated under it.

  If your time window is rolling, then the first Node Added appears in the Alarm Browser, with the second Node Added correlated under it. The second event restarts the one hour window, so that all the Node Added events appear under the same parent.



5. Click [`Verify`] to validate your entries.

Event De-Duplication

Event De-Duplication

- Like Repeated Event, but infinite time window
- Correlate all duplicate events under the most recent
- List the events to monitor in **`$OV_CONF/dedup.conf`**
  - Each line is a rule
  - Comma separated list of fields that must match for the events to be considered duplicate
    - First field is event OID
    - Rest are varbind definitions
  - Read by **`ovalarmsrv`**

©2004 Hewlett-Packard Development Company, L.P.

In the interest of making the Alarm Browser more usable as a reflection of the current status of the network and less of a historical record, “obsolete” alarms and other noise in the Alarm Browser are cleaned up automatically. By only showing the latest alarms, the Alarm Browser is as clean as possible.

There are categories of events that operators don’t want to lose but the single alarms tend to create a lot of noise in the Alarm Browser (for example, all Node Added events correlated under the most recent Node Added). The existing correlations don’t address this type of correlation and don’t have the ability to provide the infinite time window in which to correlate.

NNM allows you to identify duplicate events and correlate all duplicate events under the most recent. The correlated events are not nested and are sorted in time order. This removes a lot of noise from the Alarm Browser without losing data, allowing you to see the most recent event.

De-duplication provides a better organization of similar events. Duplicate events are consolidated under a single most recent parent, rather than being scattered throughout the history of the browser.

Configuring De-Duplication

You configure the events to be de-duplicated in the file `$OV_CONF/dedup.conf`. Each line in the

file describes a de-duplication rule. Each rule is a comma-separated list of fields that must match for the events to be considered duplicate. The first field in the rule is the Event ID that identifies the event to be de-duplicated; all remaining fields use the $ notation to identify the varbinds to be used to compare for equality. An example line is:

```
<1.2.3.4, $r, $2>
```

This says that for all events with ID '1.2.3.4', de-duplicate those that are from the same source node and have the same value in varbind 2.

Feature Operation & Design

`ovalarmsrv` is the process that reads the de-duplication configuration and detects duplicate events. When `ovalarmsrv` starts, it reads `dedup.conf`. Then it loads events from the Binary Event Store. If it finds any events that match a de-duplication signature, it displays the latest alarm and notifies `pmd` which specific child events to correlate under the most recent. `pmd` logs the correlation to the Binary Event Store and cycles the correlation notification back to `ovalarmsrv` and `xnmevents`. The Alarm Browsers turn on the correlation count indicator for the most recent alarm and delete from view the child alarms listed in the notification. As new alarms arrive from `pmd`, `ovalarmsrv` compares them to the de-duplication signatures and notifies `pmd` if de-duplication is required.

Scheduled Maintenance Correlation

Scheduled Maintenance Correlation

- Use this correlation to avoid unnecessary alarms during scheduled maintenance.
- List the affected network elements by address or name.
- The time window can be a single event or recurring.
- Set the beginning data, time, and the duration.

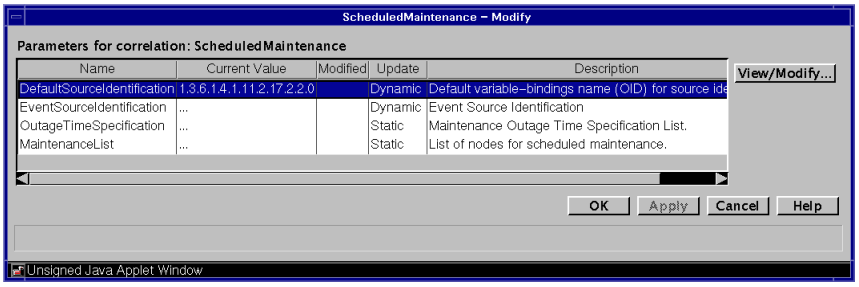
©2004 Hewlett-Packard Development Company, L.P.

Computer nodes and network segments are occasionally scheduled for maintenance. Such an occurrence typically results in a number of SNMP traps or events being generated. The Scheduled-Maintenance correlation prevents unneccessary alarms during maintenance.

When the Scheduled-Maintenance correlation detects the first event from a specified node or network segment within the specified time range, an alarm is generated and posted in the NNM Alarm Browser with the message: `Scheduled Maintenance`.

1. To set the parameters, use the `Options:Event Configuration` menu.
2. Select `Edit:Event Correlation` from the menu bar. This starts your configured browser to run the ECS manager Java applet.
3. Select `ScheduledMaintenance` and click [`Modify...`].

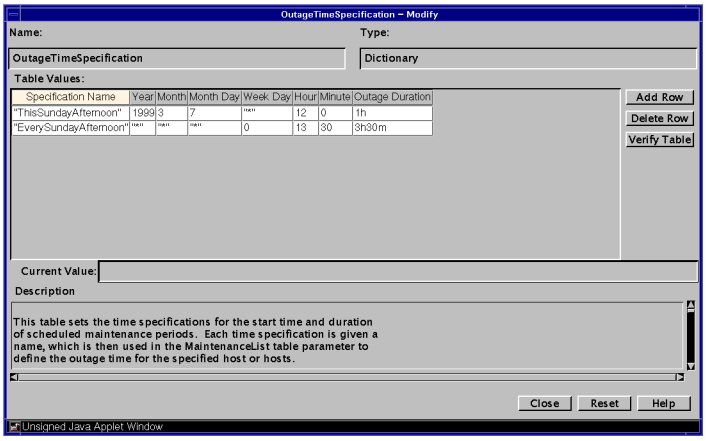
4. Select the area to modify and click [View/Modify...].



The parameters you can set are:

- `DefaultSourceIdentification`: Identifies the default source for specific traps if the event type is not defined in the `EventSourceIdentification` parameter.
- `EventSourceIdentification`: Identifies the list of event OIDs that describe specific trap events that have a source identification different from the default setting.
- `OutageTimeSpecification`: Sets the time specifications for the start time and duration of scheduled maintenance periods. Each time specification is given a name, which is referenced in the `MaintenanceList` table to define the outage time for the specified host or hosts.

  An "*" can be used to wildcard the Year and Month. An "*" in the Month Day or Week Day field means ignore this field. At least one of the Month Day or Week Day fields must have a valid integer day specification. The Hour and Minute fields cannot be wildcarded.



- `MaintenanceList`: specifies the hosts which have a scheduled maintenance period.

  The Host Specification can be a fully qualified hostname, a single IP address or range of nodes specified by using IP wildcards.

  The Outage Specification Name refers to an entry in the Outage Time Specification table.

5. Click [Verify] to validate your entries.

Copying a Correlation

Copying a Correlation

•Allows you to have one correlation operating on one set of parameters, such as time window, and another correlation of the same type operating on a different set of parameters.

•1. Copy files from the command line.
•2. Configure the new correlation from the ECS GUI.

©2004 Hewlett-Packard Development Company, L.P.

In some cases you may want to have two copies of a correlation in effect so that you can have different parameter values. For example, you may want to correlate some repeated events on a 10 minute window, and others on a 1 hour window.

In order to do this, you must manually make a copy of the original correlation from the command line, then adjust the parameters of the new correlation from within the ECS GUI.

To copy the files that comprise a correlation:

1. **`cd $OV_CONF/ecs/circuits`** on UNIX or

   `cd %OV_CONF%\ecs\circuits` on Windows.

2. Identify the correlation that you wish to copy. It has 3 files associated with it, a `".eco"`, a `".ds"`, and under the `./C` subdirectory, a `".param"` file.

3. Make a copy of each of these files using a basename that reflects your new correlation, such as

   copy `repeatedEvent.eco` to `repeatedEvent1Hour.eco`

   copy `repeatedEvent.ds` to `repeatedEvent1Hour.ds`

   copy `C/repeatedEvent.param` to `C/repeatedEvent1Hour.param`

4. Run the ECS configuration GUI from `ovw` and modify the parameters for your new correlation.

5. Enable your new correlation.

WARNING **If you copy a correlation and HP updates the definition of the original, your copy will not receive the update.**

Configuring ECS from the Command Line

Configuring ECS from the Command Line

- **ecsmgr [-instance <instance>][-stream <sname>] option**

- **ecsmgr** can be used to alter the correlation engine's configuration when it is running.
- **ecsmgr** can be used to create a stream.
- Only the superuser can use the **ecsmgr** command.
- Useful options include -info and -reset.

U5089S ©2004 Hewlett-Packard Development Company, L.P. 24

CAUTION This command is for advanced interaction by administrators. It may be useful if you have purchased ECS Designer and are doing your own integrations.

You must have `root` access on UNIX or `Administrator` rights on Windows in order to use the `ecsmgr` command. You use `ecsmgr` to alter the correlation engine's configuration when it is running.

ecsmgr [**-instance** <***instance***>][**-stream** <***sname***>] ***option***

-instance <***instance***> Identifies a particular engine instance when there are multiple instances on a host system. The default is `-instance` 1. The `pmd`-linked correlation engine is always instance 1.

-stream <***sname***> can be used with the `-policy`, `-enable` and `-disable` options to affect the `sname` stream. The stream must have been previously created with the `-create_stream` option.

Only one of the options may be specified on the command line. A complete description of the options can be found in the reference page in NNM online help (manpage on UNIX). They include:

- -info
- Load or unload a correlation ("circuit" in the reference page)
- Load or unload data
- Control logging
- `-reset` to reset the ECS system as though the system had just come up. A new settling time is required to rebuild histories.

Troubleshooting ECS

- Correlation not displayed
- Correlation has incorrect status
- ECS Configuration GUI hangs
- Debugging, testing, tracing or logging

©2004 Hewlett-Packard Development Company, L.P.

General

If the correlation you want is not displayed:

- Click [`Update View`] to refresh the list display. Updates made by other users, and updates made from the command line are not reflected in the list of correlations until you click [`Update View`].
- The correlation may not have been installed.

If the status of the correlation appears to be incorrect:

- Select the appropriate stream. The ECS Event Configuration GUI shows the status of correlations in the selected stream only. A correlation may be enabled in one stream and disabled in another.
- Click [`Update View`] to refresh the list display.

If the ECS Event Configuration GUI appears to hang when you click [`Verify Table`] or [`OK`] after modifying a parameter value:

Wait until the web server responds. Verification is performed on the web server and can take some

time if the web server is very busy.

- Check that the connection with the web server has not been severed.
- Run `ecsmgr -reset` to reinitialize the ECS runtime engine.

CAUTION	Owing to the way dynamically linked libraries are implemented in some operating systems, using this option may cause the `pmd` image in memory to grow. Avoid repeated use of this command in normal operations.

Command line control

The ECS Event Configuration GUI provides all the facilities you will need to enable, disable and control correlations. However, if you need to create a new stream to debug or test a correlation or enable tracing or logging for support purposes then you must use the `ecsmgr` command line utility. For further assistance, see the *HP OpenView Event Correlation Services Administrator's Guide*, the `ecsmgr` manpage (UNIX) or online documentation (Windows).

Lab Exercises: Enabling ECS

Lab Exercises: Enabling ECS

- Configure
  - ConnectorDown
  - PairWise
  - Pattern deletion
  - RepeatedEvent
  - De-duplication
  - ScheduledMaintenance correlations.

©2004 Hewlett-Packard Development Company, L.P.

Objective:

Manage the standard ECS correlations supplied with HP OpenView NNM.

Tip: You may choose to manually generate SNMP events for some exercises or you can use the lab scripts provided. This can easily be done from the command line, using either `snmptrap` or `ovevent`, combined with the following information:

Generic Trap Name	Number	Event ID
Cold Start	0	.1.3.6.1.6.3.1.1.5.1
Warm Start	1	.1.3.6.1.6.3.1.1.5.2
Link Down	2	.1.3.6.1.6.3.1.1.5.3
Link Up	3	.1.3.6.1.6.3.1.1.5.4

Generic Trap Name	Number	Event ID
Authentication Failure	4	.1.3.6.1.6.3.1.1.5.5
EGP Neighbor Loss	5	.1.3.6.1.6.3.1.1.5.6

For example, to send a "Warm Start" to the local management station, use either:

```
snmptrap "" ".1.3.6.1.6.3.1.1.5" "" 1 0 0
```

or

```
ovevent "" .1.3.6.1.6.3.1.1.5.2
```

Examining the ConnectorDown Correlation

1. Examine the configurable items in this correlation. What are the parameters used by this correlation?

2. Which events are processed by this correlation?

3. Which events are treated as primary events for interface cards?

Using and Configuring the PairWise correlation

1. Examine the default configuration of the PairWise correlation's parameters.

   a. Which event, "linkUp" or "linkDown", would be suppressed by this correlation? Which would be expected to occur first?

   b. Which six events are "cancelled out" by Interface Up?

2. Observe the action of the default configuration.

a. Use `Options->Event Configuration` to modify the default configurations of the linkUp and linkDown SNMP events so that a message will be displayed in the Status Alarms category and display a popup message.

b. Open the Status Alarms browser. Send a linkUp-LinkDown-linkUp sequence of events. Watch the Status Alarms browser carefully between each. You may use the tool `$OV_CONTRIB/OVTraining/NNM3/linkupdown.ovpl` or send the events from the command line. For example

```
snmptrap "" ".1.3.6.1.6.3.1.1.5" "" 3 0 0 or
ovevent "" .1.3.6.1.6.3.1.1.5.4

snmptrap "" ".1.3.6.1.6.3.1.1.5" "" 2 0 0 or
ovevent "" .1.3.6.1.6.3.1.1.5.3

snmptrap "" ".1.3.6.1.6.3.1.1.5" "" 3 0 0 or
ovevent "" .1.3.6.1.6.3.1.1.5.4
```

What did you expect to happen? What happened?

Delete the alarms when you are done to keep the next labs clear.

c. Open the ECS Configuration interface (`Options:Event Configuration, Edit:Event Correlation`). Examine the parameters for the PairWise Correlation by selecting PairWise and choosing [`Modify`].

What is the current value of the "`DeleteOrAcknowledge`" parameter?

d. Change `ChildEventImmediateOutput` to false. Repeat the link up and link down messages. What happens? Return the value to true when you finish. Delete the alarms when you are finished.

e. Change the `DeleteOrAcknowledge` parameter so that the child event will get acknowledged. Remember to [`Apply`] your changes. Test this configuration by sending another sequence of linkUp-LinkDown-linkUp events while observing the Status Alarms browser between each event.

What happened? Delete the events when you are done. Return `DeleteOrAcknowledge` to its default.

Observing Pattern Deletion

The Pattern Delete function helps handle pairs of events that would ordinarily be ignored by the Pair-Wise correlation. In this exercise, you will set up and use this function.

For this exercise, you may also use the standard SNMP Link Down (Event ID: .1.3.6.1.6.3.1.1.3) and Link Up (Event ID: .1.3.6.1.6.3.1.1.4) events. You will need to change their logging as well as the ECS definition for these events. As these events may occur in a normal setting, you may want to reset them after this lab.

1. Using the `Options:Event Configuration` GUI, configure a new pair of events under the OpenView enterprise.

   For example: `ChildTestEvent .1.3.6.1.4.1.11.2.17.1.0.2004`

   `ParentTestEvent .1.3.6.1.4.1.11.2.17.1.0.2005`

   Make sure that the event is displayed in one of the alarm categories. You may want to have a pop-up message and a specific severity.

   Issue each event using `ovevent` to be sure of its operation.

2. Using the ECS Configuration GUI, select the Pair-Wise correlation and make the following adjustments for the lab:

   a. Modify the time window to 1 minute.

   b. Add the child and parent events to the table of InputEventTypeListStringSources.

   c. You can use “agent-addr” for Key 1 and 0 for Key 2 for each event.

   d. Be sure to set the line in the Accept column to true.

   e. Verify the table and select Close and then OK.

3. Issue the events within the time window for the Pair-Wise correlation. This should result in the normal, or usual, operation of the correlation.

4. Now issue the child event followed by the parent event outside the time window specified in

the correlation. Note the results in the browser.

5. Finally, issue multiple occurrences of these events both within and outside the time window in the correlation. Note the impact in the browser.
6. Be sure to reset the time window in the Pair-Wise correlation after this lab.

Using and Configuring the RepeatedEvent correlation

1. Configure the SNMP_EGP_Down event so that you will recognize it when it is received.
   a. Configure the event so that it will be displayed in the Status Alarms category and a pop-up message will be generated.
   b. Test your configuration by generating by executing:

      ```
      snmptrap "" ".1.3.6.1.6.3.1.1.5" "" 5 0 0
      ```

      or

      ```
      ovevent "" .1.3.6.1.6.3.1.1.5.6
      ```

2. Examine the existing, default, configuration of the RepeatedEvent correlation.
   a. Which SNMP protocol events are configured for processing by this correlation? Which specific OpenView events?

   b. Will any events be placed in the Alarm Browser, and then removed later?

   c. What is the longest period of time during which reported events will be suppressed by this correlation? Is it possible to have a longer period of suppression without changing the `RepeatedTimeWindow`?

3. Modify the window for this correlation to 1 minute.

   a. Select the `RepeatedTimeWindow` and change the current value to 1 minute.

   b. Apply the change. What happens?

   c. Send 4 EGP_Neighbor_Loss events, 25 seconds apart. You may use the script `$OV_CONTRIB/OVTraining/NNM3/egpLoss.ovpl`. Which events are observed? Explain.

4. Observe the use of this correlation.

   a. In the `InputEventTypeList`, change the Enable Event Type to `true` for the EGP Neighbor Loss event.

   b. Close the `InputEventTypeList` window and [`Apply`] the change. Was the correlation restarted? Explain.

   c. Send 4 EGP_Neighbor_Loss events, 25 seconds apart.
      - When does the number occur in the Correlation column of the Alarms browser?
      - How many popups are generated?
      - Look at the correlated events. How many are listed?

   d. Change `CreateUpdateEvent` from `false` (the default) to `true`. Send 4 EGP_Neighbor_Loss events, 25 seconds apart.
      - When does the number occur in the Correlation column of the Alarms browser?
      - How many popups are generated?
      - Look at the correlated events. How many are listed?

5. Change `CreateUpdateEvent` back to the default: `false`. Change the `RollingWindow` parameter from `false` to `true` and apply the changes.

   a. Send 4 EGP_Neighbor_Loss events, 25 seconds apart, as before.

      - How many popups did you observe? Why?
      - What is different about the correlated events?
      - Summarize the effect of the RollingWindow.

   b. Change `CreateUpdateEvent` back to `true`. Send 4 EGP_Neighbor_Loss events, 25 seconds apart, again.

      - How many popups did you observe? Why?
      - What is different about the correlated events?
      - Summarize the effect of the RollingWindow.

6. Return the values to their defaults: RollingWindow, false; CreateUpdateEvent: false; RepeatedTimeWindow: 10m.

De-Duplicating Events

Event de-duplication is preconfigured in NNM for a pre-defined set of events. In this exercise, you will expand that list to include a user-defined event.

1. (Optional) Using the Event Configuration GUI, create a new Alarm Category for this set of exercises.

   While this step is optional, it helps keep the focus of attention on the desired exercise.

2. Using the Event Configuration GUI, create a new event under the OpenView Enterprise. For example: `DeDupTestEvent .1.3.6.1.4.1.11.2.17.1.0.2003`. Direct the event to the new Alarm Category, and optionally add a pop-up message and whatever severity you desire.
3. Use `ovevent` to send a few of these events to your browser.

4. Modify the `$OV_CONF/dedup.conf` file and add this event definition to the list of events to be de-duplicated.

5. Stop and then re-start the `ovalarmsrv` service. This re-reads the `dedup.conf` file.
6. Re-issue a series of the test events. Note the effect in the Alarm Browser.

Using and Configuring the Scheduled Maintenance correlation

1. Define a Scheduled Maintenance correlation for the systems in your classroom.
   a. Add a row to the `OutageTimeSpecification` parameter table and create a time period starting roughly 5 minutes from now and lasting for 10 minutes.

   b. Modify the `MaintenanceList` parameter table to specify that the classroom hosts have a scheduled maintenance planned during the defined `OutageTimeSpecification`.

   c. Apply the changes. Was the correlation restarted? Explain.

   d. Enable the ScheduledMaintenance correlation. You may want to reconfigure the Status Polling cycle of your management station to be about 30s.

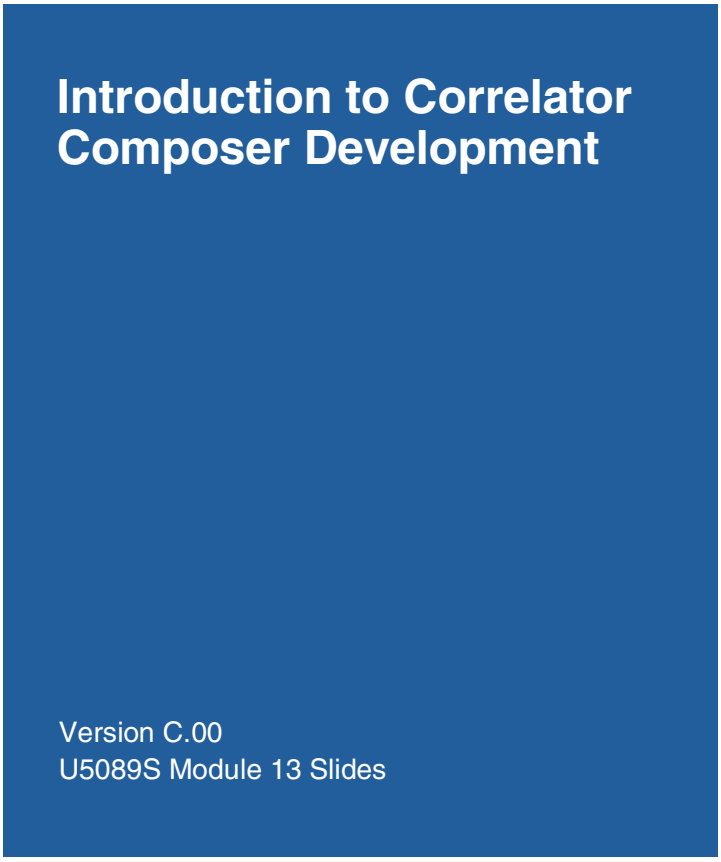
      - Your instructor will take one system offline.

      - Experiment with generating events. You may also use the script `$OV_CONTRIB/OVTraining/NNM3/maint.ovpl`.

      - Look at the correlated events for the scheduled maintenance

13 Introduction to Composer Development

Module Objectives



At the completion of this module, you will be able to:

- List the concerns with using Composer.
- Develop accurate and complete correlator requirement specifications.
- Translate a correlator requirement into a design.
- Start the Composer user interface.
- Select the Composer template appropriate to a problem description.
- Create and organize Correlator Store files.
- Configure Operator access to correlators.
- Deploy a correlator.

Correlation Composer Overview

Correlation Composer Overview

- Correlation Composer enables you to:
  - Simplify the development of customer specific/user defined correlators
  - Specify correlation logic for simple to complex problems including root cause analysis
  - Use six pre-packaged Correlator templates for commonly occurring correlation logic
  - Call Perl or C functions, if there is coding required.

- Correlation Composer has 2 components:
  - Graphical user interface to specify the correlation requirement
  - Runtime Component to implement the correlation

©2004 Hewlett-Packard Development Company, L.P.

The HP OpenView Correlation Composer is a combination of a pre-packaged ECS correlation and the graphic user interface used to parametrize and define correlators to perform Event Correlation.

The HP OpenView Correlation Composer has two Graphical User Interfaces (GUIs) that enable users to tailor the event correlation behavior for correlators that are shipped with OpenView products, and simplify the development of customer developed correlators. Correlators can be used out of the box or can be easily fine tuned to fit your environment like a glove without any programming knowledge.

The Composer comes prepackaged with six Correlator Templates that ease the creation of correlation solutions by providing correlation models for the most common correlation tasks. The Composer with the Correlator Templates form the basis for simple correlation tuning and development.

With Composer, you can take the knowledge your network administration experts use and automate it for correlation before the Operator sees an alarm in the browser. This includes problem diagnoses, collapsing several related events into one event for brevity, enriching events with external data (like topology or customer information) giving operators the additional information necessary within the event.

NOTE All de-duplication configurations, correlators defined within Composer, and all correlations defined within ECS must work well together in your NNM

environment. Before you begin, it is critical that you print out and read the following white paper:*Developing_NNM_Event_Reduction.pdf*

Windows: *install_dir*\Doc\WhitePapers\

UNIX: $OV_DOC/WhitePapers/

This information explains how to ensure that you do not break existing implementations.

The event logic or flow aspects of these correlators can be generalized and so what remains to implement a correlator is to configure one of the templates into a specific instance. An example from the correlators provided with NNM is Multiple Reboots. Managed devices may be rebooted several times by an administrator within a period of time. The only relevant operator information is if the device continues to reboot and/or stays down. Multiple Reboots is a Composer correlator instance of the Rate template that is configured to receive coldstart and warmstart traps. If 4 such events come within a 5 minute period then a new reboot trap is issued; otherwise the coldstart and warmstart traps are ignored. The instance data in this case are the incoming event signatures and the time interval and event count that trigger the new event to be sent.

Planning the Correlation

•Get requirements
•Design a solution
•Implement and test on a test machine
  • Create a sample log of events
•Deploy to production environment and verify

•Understand the network traffic patterns
•Know the NNM architecture
•Identify external data requirements
•Know which other correlations utilize those events

 ©2004 Hewlett-Packard Development Company, L.P.

To develop a correlator you need to:

- Learn to develop correlators through Correlation Composer training and/or documentation.
- Design and debug the correlator on an isolated test machine.
- Deploy the correlator to each NNM management station and collection station, as described in the *Developing_NNM_Event_Reduction.pdf* white paper.

To design an effective correlator, you must:

- Understand typical network event patterns and what they mean.
- Know the architecture of your network management system.
- Identify external data access (ECS Annotation) requirements.
- Define the problem concisely in terms of filtering and time-related if-then statements.
- Determine if all data is available from the event's variable-bindings.

  If not determine if:

  — The external data is subject to change or is relatively stable.

  — The required data is small enough to store in memory.

— The access speed and reliability is sufficient when extracting from your database (particularly if the database is remote).

Then, to develop the correlators, you need to:

- Set up a test platform.
- Develop sample logs of events. See the *Developing_NNM_Event_Reduction.pdf* white paper for information about how to create log files of events, and how to feed them into the NNM event correlation system in your test environment.

Correlator Development Process

Correlator Development Process

- Understand the requirements
- Choose and design the correlator to create
- Configure the correlator
  - Define the incoming events
    - Specify the Alarm Signature
    - Declare variables
    - Define the Message Key
    - Define the Advanced Filter
  - Specify parameters
  - Define new event(s) if necessary
  - Specify callback functions
- Generate a test event stream
- Test the correlator
- Deploy into production

U5089S ©2004 Hewlett-Packard Development Company, L.P. 4

The overall process for designing and developing a correlator begins with understanding the requirements thoroughly. What event reduction is desired for the Operator in the Alarm Browser? Where are those excess events coming from?

Based on the requirements, you can design one or more correlators to solve the problem. This includes selecting the appropriate correlator template, determining which information is relevant, and whether you will need any auxiliary information from outside sources.

You use the Composer developer GUI to define the correlator itself. A correlator uniquely identifies a unit of correlation logic to be applied to an event or a set of events. Every correlator has three main sections:

1. Alarm Definition section

   The first section in the Correlator configuration is the Alarm Definition in the Definition panel in the Correlator window. The Alarm Definition section is divided into five subsections:

   - Alarm Signature

     The Alarm Signature (primary filter) forms the first level of filtering based on event attributes. Further processing takes place when an event matches all attributes set in the Alarm Signature. The Alarm Signature is a set of Attribute Name, Operator and Value groups.

   - Variables

Variables are names assigned to values to be used while defining Correlators or Global Constants. Once assigned, the name can be used in other sections of the Composer.

All attributes in the events can be accessed as variables where the variable name is the attribute name.

- Advanced Filter

  Events that have entered a correlator, can be further filtered based on the Advanced Filter Condition (secondary filter). This condition is typically used to define filters based on external factors like topology. (Contrast this to the Alarm Signature where the filter is defined purely on event attributes.)

- Message Key

  The Message Key is evaluated for each incoming event that passes the Alarm Signature and Advanced Filter. Events with identical MessageKeys are correlated together.

- Parameters

  Parameters are specified to change the default behavior of the basic correlator. Typically the time period for which the correlation is to be monitored is specified here.

2. New Alarm Section

   The user configures the specifications of new events to be created or altered in this section.

3. Callback Section

   A Correlator can result in events getting discarded or new events being created. There are two types of callbacks - discard and create. Every time an event is discarded, the discard callback is invoked if configured. Every time an event is either altered or created by a rule, the create callback is invoked if configured. This can be used to create an audit trail or correlate child events.

Once you have configured the correlator, you can use a test event stream to see how it operates. NNM provides tools to send the test stream through pmd and ECS and you can see the results in the Alarm Browser.

This process may be iterative. Based on the first results you see, you may want to adjust the operation of the correlator and retest it. When it operates to your satisfaction, deploy it in your production environment.

Creating Correlator Requirements

Creating Correlator Requirements

- Which event(s) to look for and how to recognize them
- Which sources or other qualifiers
  - Within the event
  - Defined elsewhere
- What action to take when an event is received
  - Which events to create, discard, correlate
  - Other notifications
  - Pass to other correlators or not

©2004 Hewlett-Packard Development Company, L.P.

For the initial portion of the course, we will assume that your customers provide adequate requirements specifications. Later we'll discuss monitoring the actual event stream to determine your own requirements.

Composer User Interface Modes

The Composer is designed to operate in two modes, namely:

- Operator's Mode

  The Composer Operator maintains correlation in the network. Each operator handles a part of the correlation logic and is responsible for this continuous maintenance. The Operator can refine the correlation logic based on the network requirements. The Operator has limited access to Composer. The access is governed by the permissions set by the Developer in the Security file and the area of operation specified in the NameSpace files.

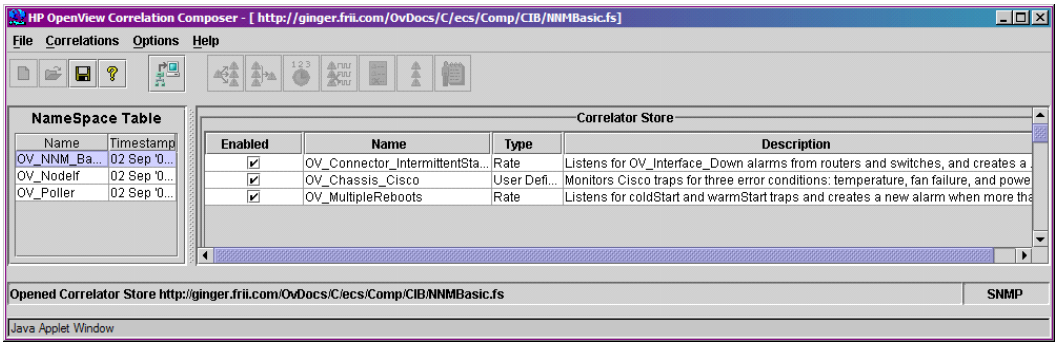
- Developer's Mode

  The Composer Developer can set up, create or modify correlation logic for the network environment. The Developer is responsible for configuring Composer, setting up Operator access rights using Security files and deciding the area of operation for the Operator using NameSpace files.

So, while assigning a user's role as Operator, appropriate conditions and permissions must be set in order to enable the Operator to efficiently manage the network. To simplify this task, Security files and NameSpace files are provided in order to link the Operator to that portion of the network. In this manner, many operators can be given access to different portions of the network by linking them to Security and NameSpace files.

Operator Access to Composer

Operator Access



From the ECS Configuration Interface, select Composer and click [**`Modify`**].
From the command prompt, type **`ovcomposer -m o`**.

 ©2004 Hewlett-Packard Development Company, L.P.

Composer for the Operator

NOTE Throughout NNM documentation and training, these tasks are considered Administrator responsibilities, not tasks that would be done by what the rest of the product calls an Operator.

Correlation logic is developed by the Correlator Store Developer, who is also responsible to provide access rights to the Operator. The Operator has limited access on the Correlator Store files. These rights are governed by the information provided in the Security and NameSpace files. The Operator:

- Cannot create new Correlators and hence no new Correlator Stores either.
- Has access only to those files as specified in the NameSpace file. This file is created and maintained by the Developer. Only those files specified in the NameSpace files will be visible to the Operator in the NameSpace table in Composer.
- Can edit values of only those parameters that are specified in the Security file. This file is created and maintained by the Developer.

Starting Composer Operator Interface from NNM

In the ECS Configuration Management GUI, select the row with Composer and select the `[Modify]` button.

Starting Composer in Operator mode from the Command Line

To start Composer in the Operator's mode, type

```
ovcomposer -m o
```

The command resides in

UNIX: `$OV_BIN`

Windows: `%OV_BIN%`

The Composer in the Operator's mode opens with the list of Correlator Store files and the last modified time of the Correlator Store in the NameSpace table or from the ECS config GUI.

Operator Actions in Composer

`Correlations:Deploy` Deploys the Correlator Store files to the ECS engine.

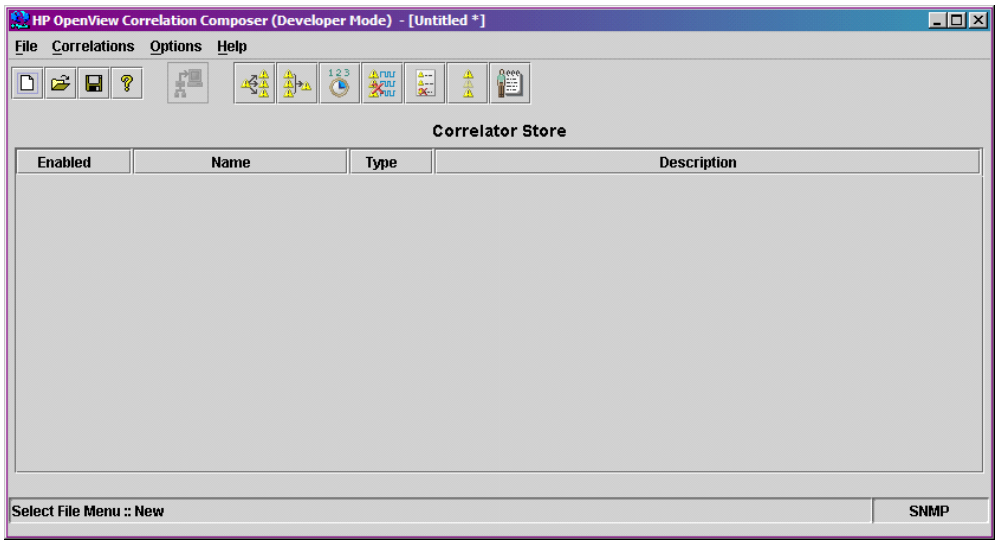
`Options:Forcefully Unlock` Provides mutually exclusive access to the Correlator Store.

`Options:Appearance` Displays a submenu for selecting the kind of Look and Feel of the interface.

`Options:View Backup` Displays a submenu to select the version of backed up file.

Starting the Composer Developer Interface

Starting the Developer Interface



From the command prompt, type **`ovcomposer -m d`**.

 ©2004 Hewlett-Packard Development Company, L.P.

Starting Composer from the Windows Start Menu

1. Click on the `[Start]` button and point to `Programs`.
2. Select *<program_group>*`:Correlation Composer`

Starting Composer from the Command Line

`ovcomposer -m d` to start in Developer mode.

The command resides in

UNIX: `$OV_BIN`

Windows: `%OV_BIN%`

`ovcomposer` recognizes the following options:

`-i` *<Correlator Store name>* Specifies the location and name of the Correlator Store to open.

This option is valid only when Composer operates in the Developer mode. If no filename is specified, Composer opens an untitled Correlator Store with Event Type set to SNMP.

`-N <NameSpace filename>` Specifies the location and the name of the NameSpace configuration file. If no filename is specified, the default NameSpace Configuration file `$OV_CONF/ecs/CIB/NameSpace.conf` is selected.

`-p <deploy configuration filename>` Specifies the location and name of the Deploy Configuration file. If no filename is specified, the default Deploy Configuration file `$OV_CONF/ecs/CIB/deploy.conf` is selected.

`-h` Displays the `ovcomposer` usage message.

NOTE Ensure that the NameSpace file referenced in the Deploy Configuration file is the same file passed with the `-N` option. If the filenames differ while using the two options then it leads to creating one set of Correlator Stores and deploying a completely different set of Correlator Stores.

Refer to the *ovcomposer* manpage for more details.

Example

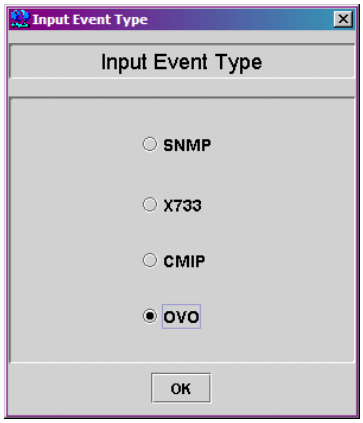
To start Composer in the Operator mode referring to the NameSpace file `c:\composer\names.conf`:

```
ovcomposer -m o -N c:/composer/names.conf
```

Setting the Event Type

Set Event Type

- Can only set when file is created.
- SNMP and OVO are types supported.



U5089S ©2004 Hewlett-Packard Development Company, L.P. 9

The Event Type decides the kind of events that will enter ECS. The event types supported by Composer are listed below.

The Event Type is selected at the time of creating the Correlator Store.

To select the Event Type:

1. Select `File:New`. The Input Event Type window is displayed.
2. Select the Event Type from the list and click on [`OK`].

NOTE The default Event Type is SNMP.

The Correlator Store can be created for one Event Type at a time. If you want to change the Event Type, close the currently opened Correlator Store and repeat Steps 1 and 2.

Table 1 Event Types Supported

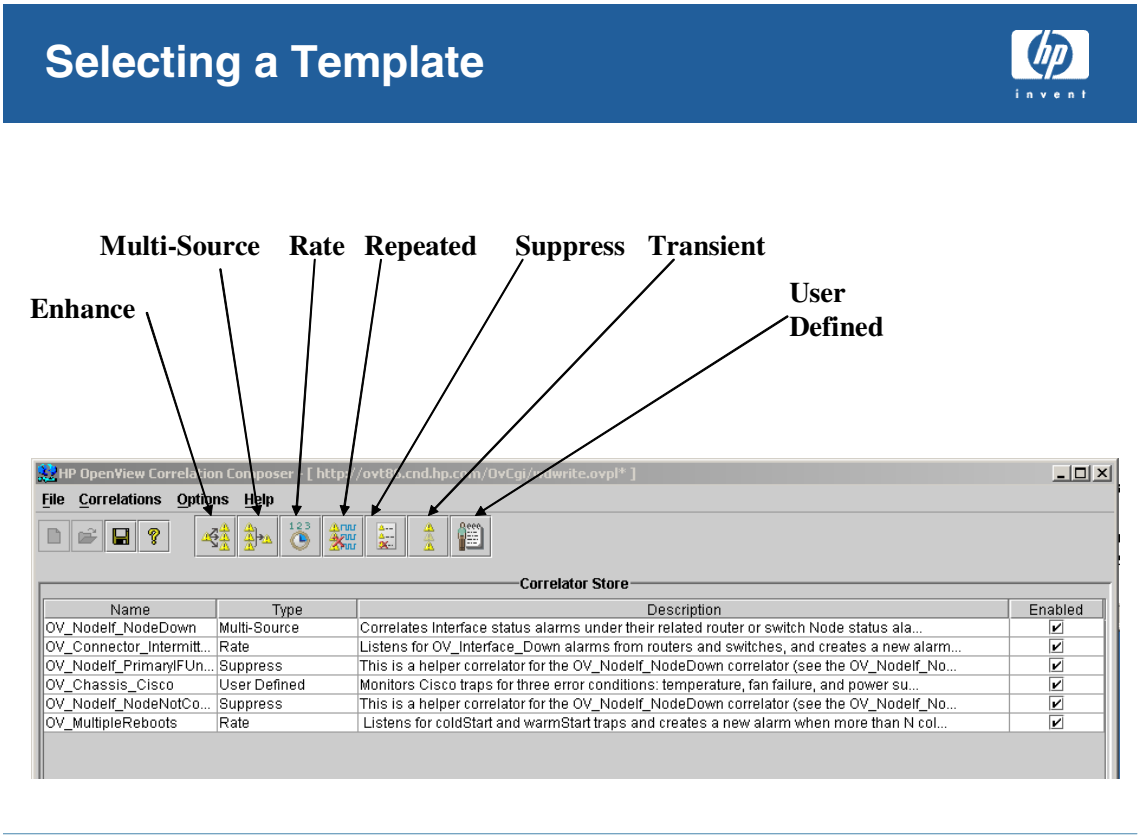
Type	Description
CMIP	CMIP based events
OVO	OVO messages
SNMP	SNMP traps

Table 1 **Event Types Supported**

Type	Description
X733	X.733 based events

NOTE CMIP and X733 are not supported.

Selecting a Template



Correlation Composer provides templates for the most commonly used correlation logic. You can access these from the toolbar icons or from the Correlations menu item. These templates make it easy for you to develop your own correlators:

	Enhance	The Enhance correlator template is used to trigger the creation of new event(s) or to augment the information content of an event.
	Multi-Source	The Multi-Source correlator template is used to define a relationship among different incoming event types.
	Rate	The Rate correlator template is used to count the number of events in a time period. It can issue a threshold event if too many are received.
	Repeated	The Repeated correlator template can be used to discard duplicate events within a time window.
	Suppress	The Suppress correlator template is used to discard unwanted events.
	Transient	The Transient correlator template detects transient failures (down followed by up) and optionally discards them and/or issues a threshold event if too many pairs are received.

In addition to the pre-defined Correlator Templates, the Composer also allows you to define your own correlations using the User-Defined Correlation Template.

NOTE Support for the correlator template named User Defined is not included. If you wish to use the Correlation Composer User Defined template, contact HP to purchase a Partner Care Extended support contract before beginning.

Correlator Store Files

Correlator Store Files

- Logic can be placed in modular files to be merged before deployment
- Place files under **`$OV_CONF/ecs/CIB`**
  - May be in subdirectory
- Name contains letters, digits, or underscores
  - **`.fs`** extension is appended automatically

U5089S ©2004 Hewlett-Packard Development Company, L.P. 11

A Correlator Store contains a set of Correlators which define correlation requirements for the network.

To create a new Correlator Store:

- Select `File:New` from the menu.
- Click on the New icon in the Standard Toolbar.

If a file is already open (and has been modified), Composer prompts you to save the file. Note that the default Event Type is SNMP. To change the Event Type see that topic.

To save the Correlator Store file:

1. Select `File:Save` from the menu to display the file browser.
2. Enter the Correlator Store file name in the `File` panel.
3. Select `[OK]` to save the file.

File Naming Restrictions

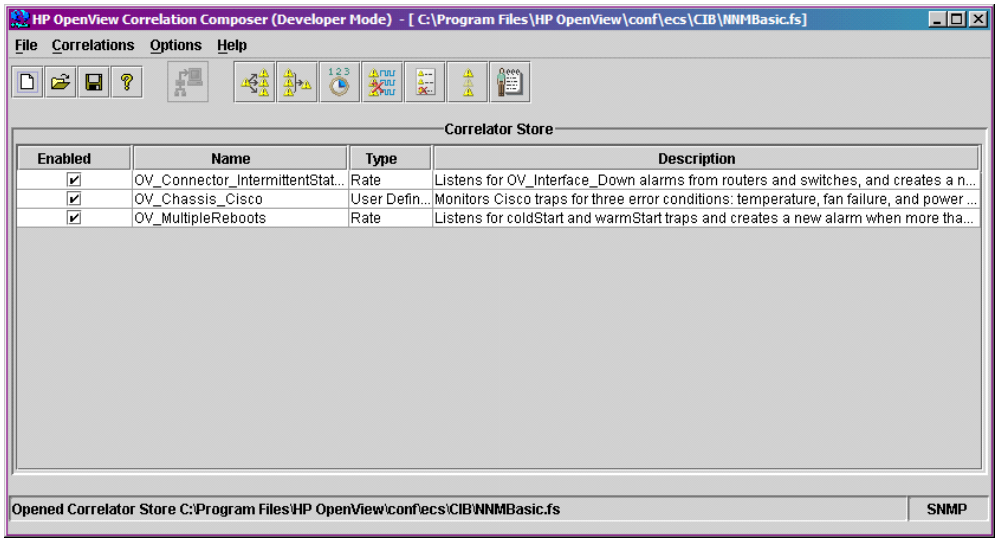
Use file names that start with a letter and contain only letters, digits and underscore(_). For example: `my_configuration` is a valid file name. The extension `.fs` is supplied automatically to the filename entered.

Location of Correlator Store

Correlator Stores should be placed in any directory under `$OV_CONF/ecs/CIB`. Ensure that the correct path is specified before saving the file.

Opening a Correlator Store File

Opening a Correlator Store



Select **`File:Open`**

©2004 Hewlett-Packard Development Company, L.P.

Opening an existing Correlator Store

To view a Correlator Store file:

1. Select the filename that you want to open.
   - Select `File:Open` or Click on the Open icon in the Standard Toolbar.
   - Select the Correlation Store name from the file browser window.
2. The Open file browser window is displayed. Select the name of the file you want to open.
3. Click `[Open]`. The Correlator Store file is displayed.

IMPORTANT Correlator Stores created in a Composer version prior to Version 3.3 must be migrated to the latest version.

Modify an existing Correlator Store

After you have created a Correlator Store, you can modify its properties whenever required. To modify the Correlator Store:

1. Select `File:Open` to open the Correlator Store. This opens the file browser window.
2. Select the filename from the file browser window. The Correlator Store with the Correlators is displayed.
3. To open a Correlator, you can do one of the following
   - Select the Correlator in the table and double click the mouse button. The Correlator window opens.
   - Select the Correlator in the table and right click the mouse button. From the menu displayed select `Modify`. The Correlator window opens.
4. Make the required changes as you did when you created the file.
5. To save the changes, select `File:Save`. The file is saved with the same name.

   To save the changes made into the different file, select `File:Save As`.

Exclusive Access to Correlator Stores

Exclusive Access

•The Correlator Store file locking functions in the following modes:

- Composer's Operator mode
- Composer's Developer mode
- Standalone Deploy script
- Correlator Store Deploy procedure
- Standalone Merge script when invoked with the -namespace option.

©2004 Hewlett-Packard Development Company, L.P.

Mutually Exclusive Access to Correlator Store files

To avoid overwriting of data in Correlator Stores due to concurrent access by multiple users (Developers or Operators), Composer provides the facility to lock a file.

The Correlator Store file locking functions in the following modes:

- Composer's Operator mode
- Composer's Developer mode
- Standalone Deploy script
- Correlator Store Deploy procedure
- Standalone Merge script when invoked with the -namespace option.

A lock file (*<filename>*.lock, where *<filename>* is the name of the Correlator Store) is created when the Correlator Store is opened and the file is not already in use. Acquiring a lock provides total access on the Correlator Store file.

The creation of the lock file fails if the Correlator Store is in use. Each mode operates differently in this situation:

- In the Operator mode, Composer displays an error message and opens the file in read-only mode.
- In the Developer mode, Composer displays an error message and aborts the file open action.
- The Deploy procedure displays an error message and aborts the deploy action.
- The Standalone merge script displays an error message and aborts the merge action.

The lock is removed when the Correlator Store is closed.

If any of the above actions abort abruptly while a Correlator Store is locked, use one of the following mechanisms to recover the Correlator Store:

- In the Operator mode, select `Options:Forcefully Unlock` after highlighting the locked Correlator Store. The same can be done in the case of Deploy operation from Composer.
- In the Developer mode, the Developer can manually remove the lock file, *<Correlator Store filename>*`.lock` file, which resides on the same directory as the correlator store. The same can be done in the case of abort during standalone deploy and merge.

NOTE	Though the user is allowed to make changes to the Correlator Store, the Operator does not have mutual exclusive access to this file.

WARNING	**It is recommended that this option be chosen with caution as there is always a possibility that important data could be lost while multiple operators save the Correlator Store files.**

Configuring Operator Access

Configure Operator Access

- Planning the configuration can be divided into the following steps:

1. Creating Correlator Stores
2. Listing Correlator Stores
3. Creating the NameSpace file
4. Creating the Security file
5. Creating the Deploy Configuration file

©2004 Hewlett-Packard Development Company, L.P.

Step 1: Create Correlator Store

The Developer creates Correlator Store files in such a way that correlators defined for set environments are grouped logically, that is, all Correlators logically bound are put into one Correlator Store.

Step 2: List Correlator Store

Correlator Store files defining the Correlation logic must be made accessible to Operators. A list of Correlator Stores to be displayed to the Operator is created.

Step 3: Create the NameSpace file

A default NameSpace file is available at `$OV_CONF/ecs/CIB`. To override the specifications present in the NameSpace file:

1. Copy the default NameSpace file to a local directory.
2. In the NameSpace file, list the names of Correlator Stores and the path of the Correlation Store (relative to CIB).
3. Save the file with extension .conf.

Step 4: Create the Security file

A default Security file is created when the Correlator Store file is saved the first time. This file is present in the same directory as the Correlator Store. To override the specifications present in the Security file:

1. List the token identifiers and the parameters that can be edited.
2. Save the file as *<Correlator Store filename>*.sec. Ensure that the file is stored in the same directory where the Correlator Store file is stored.

IMPORTANT It is the responsibility of the Developer to ensure that correct permissions are provided to the file, so that this file is not overwritten or edited erroneously.

Step 5: Create the Deploy Configuration file

A default Deploy Configuration file is available at `$OV_CONF/ecs/CIB/deploy.conf`.

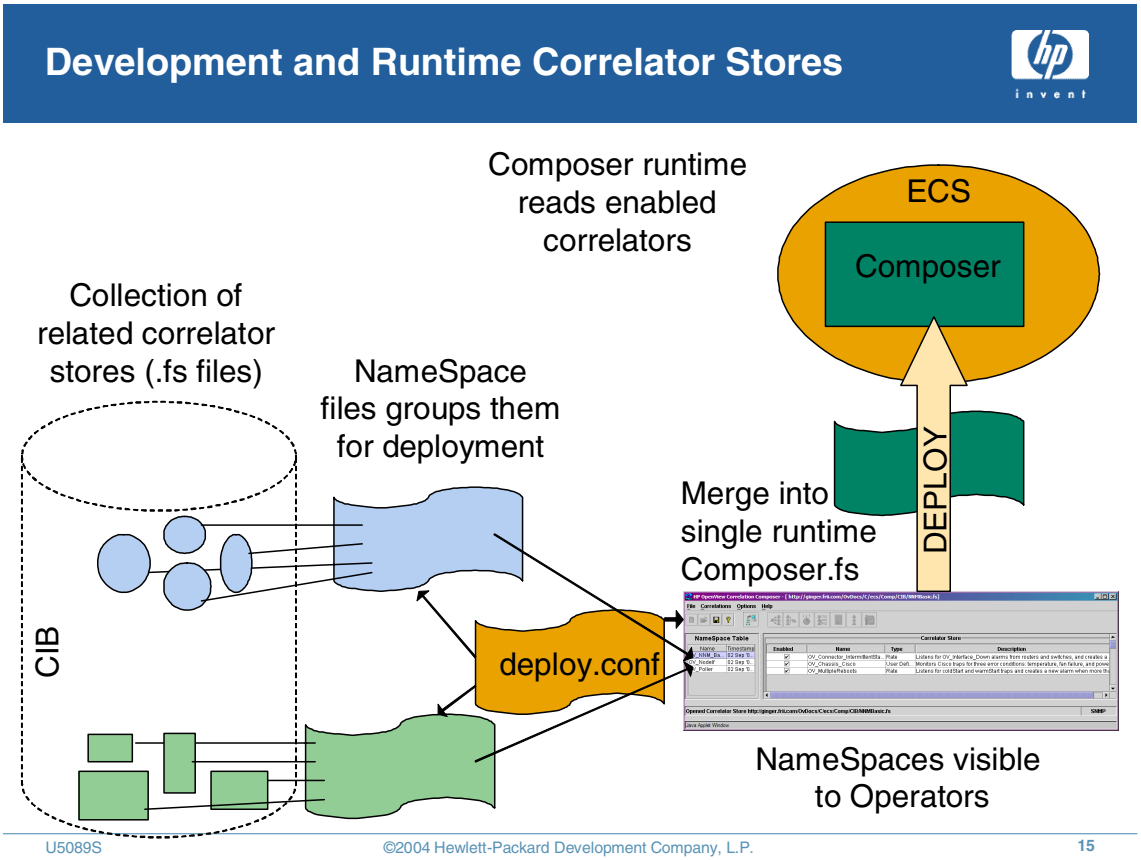
To override the specifications present in this file:

1. Copy the default Deploy Configuration file to any local directory.
2. Edit the file with values and names specific to your environment.
3. Save the file with the extension `.conf`.

The newly created NameSpace and Deploy Configuration files are bound together based on the entry `NAMESPACE_FILE` in the Deploy Configuration file. Hence, ensure that the correct NameSpace filename is provided in the Deploy Configuration file.

IMPORTANT After creating the configuration files for your environment, ensure that the correct filenames with locations are specified at the time of startup of Composer (refer to *ovcomposer* manpage). This is mandatory because if no files are specified, then Composer picks up the default configuration files.

Development and Runtime Correlator Stores



NNM has a default Correlator Store in `$OV_CONF/ecs/circuits/Composer.fs`. To see it, select `File:Open`. This is the only file that the running ECS references. Develop in a separate file, then merge your changes into the `Composer.fs` file.

Configuring a NameSpace

Configuring a NameSpace

- Used to group correlators stores to assign permissions for operators
- No other relevance
  - NOT preventing duplicate correlator or variable names
- Place in or under CIB directory
- Filename must end in .conf
- Each file contains multiple lines, one line for each correlator store to be presented to this operator group.

  logical_CS_name=actual_CS_location
- The actual correlator store location is relative to
  - UNIX: **$OV_CONF/ecs/CIB**
  - Windows: **%OV_CONF%\ecs\CIB**

©2004 Hewlett-Packard Development Company, L.P.

A NameSpace file contains a list of Correlator Store files, grouped logically together to define Operator profiles. This grouping is specifically used to assign access permissions to the Correlator Store files to Operator profiles and has no other relevance. The list of Correlator Stores specified in this file decides the area of operation within which the Operator can work.The NameSpace file is a simple ASCII file that can be edited using any standard text editor. This file is a listing of name-values pairs of the Correlator Store name versus the relative path of the location of this Correlator Store.

NOTE To create/edit the NameSpace file, the user must have root access on the machine where Composer is installed.

The general syntax for a NameSpace file is

```
< Logical Name1>=< Location of Correlator Store file>
< Logical Name2>=< Location of Correlator Store file>
```

where

<Logical Name1>, *<Logical Name2>* are the logical names of Correlator Store files as will be displayed in Composer when started in the Operator mode.

<Location of the Correlator Store file> is the location of the Correlator Store file relative to the directory `$OV_CONF/ecs/CIB`.

Following is a sample of the NameSpace file:

```
#comment line:path relative to the $OV_CONF/ecs/CIB directory

ATM=ATM/atm.fs

OV=OV/ov.fs

CISCO=CISCO/cisco.fs
```

Rules while creating a NameSpace file

The NameSpace file must be edited following the rules provided, to provide access to the Correlator Store files.

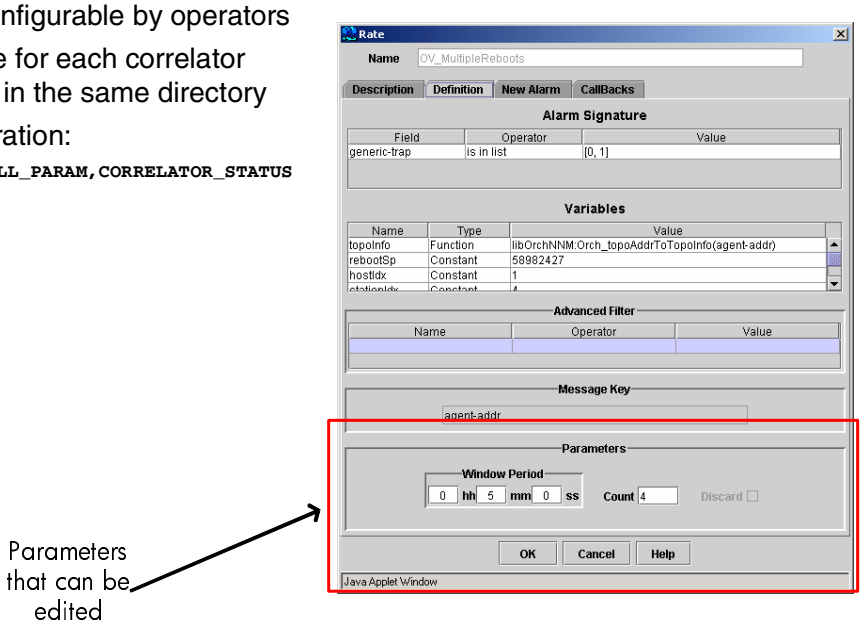
1. The location of the Correlator Store file on the right hand side of "=" is always relative to the directory `$OV_CONF/ecs/CIB`.
2. Correlator Store files present above the directory `$OV_CONF/ecs/CIB` are not accessible. They must be present under a subdirectory (typically, with the same name as that of the Correlator Store) or under the directory `$OV_CONF/ecs/CIB`.
3. No blank space allowed before and after the "=" sign.
4. Every entry for a logical name is made on a separate line.
5. Logical names of Correlator Stores must be unique.
6. All file location paths specified must be on a single line and must not flow over to the next line.
7. Ensure that the file is always saved with the extension `.conf`.
8. All comments are preceded by the hash (`#`) symbol.

IMPORTANT Ensure that the NameSpace file referenced in the Deploy Configuration file is the same file passed with the -N option when Composer is started (refer to the *ovcomposer* manpage). If the filenames differ, then it leads to creating one set of Correlator Stores and deploying a completely different set of Correlator Stores.

Configuring the Security File

Configuring a Security File

- Defines which parameters of a correlator are configurable by operators
- One security file for each correlator store file, stored in the same directory
- Default configuration:
  - **ALL_TEMPLATE=ALL_PARAM,CORRELATOR_STATUS**



 ©2004 Hewlett-Packard Development Company, L.P.

The Security File of Composer contains a list of fields/parameters that can be edited by the Operator. Every Correlator Store file created has a corresponding Security file associated with it and is stored in the same directory as that of the Correlator Store file.

A default Security file is created for every Correlator Store file saved the first time. This file is stored in the same directory as the Correlator Store file. The default Security file(*<Correlator Store filename>*.sec, where <Correlator Store filename> is the name of the Correlator Store for which the security file is created), allows all values of Parameters in the Alarm Definition section of all Correlators to be edited. The Security file can be edited so as to restrict editing of values.

The Security file is a simple ASCII file and can be edited using any standard text editor.

The general syntax of the Security file is as follows:

```
ALL_TEMPLATE=TOK_LIST

GLOBAL_CONSTANT=GC_LIST

CORRELATOR_TEMPLATE=TOK_LIST

CORRELATOR_NAME=TOK_LIST
```

where,

`ALL_TEMPLATE=TOK_LIST` All Correlator Templates have access to edit values of parameters listed. `TOK_LIST` can be any token identifier listed in the table. However, any other condition specified in the Security file will not be overridden by this

statement.

GLOBAL_CONSTANT=GC_LIST List of Global Constants whose values can be edited. GC_LIST is the list of Global Constants whose value can be edited.

CORRELATOR_TEMPLATE=TOK_LIST List of parameters for the specific Correlator Template type for which values can be edited. TOK_LIST can be any token identifier listed in the table and CORRELATOR_TEMPLATE is the Correlator template names as listed.

CORRELATOR_NAME=TOK_LIST List of parameters for the specific Correlator whose values can be edited. CORRELATOR_NAME is the name of the Correlator and TOK_LIST is any token identifier as specified.

Following is a sample of the Security file:

```
OV_Chassis_Cisco=NEW_ALARM

USER_DEFINED=ALARM_SIGNATURE

ALL_TEMPLATE=WINDOW
```

From the above example, the following can be interpreted:

- For the Correlator OV_Chassis_Cisco, parameters defined for New Alarm creation can be edited.
- For all User Defined correlators the values in the Alarm Signature section can be edited.
- For all Correlator Templates other than User Defined and the Correlator OV_Chassis_Cisco, the value for the Window parameter can be edited.

Rules while creating the Security File

The Security file must be created following the rules given below:

1. Every condition to be specified must be made on a separate line.
2. Token parameters are separated by commas.
3. No blank space is allowed before and after the commas used as separators for token identifiers.
4. All comments are preceded by the hash (#) symbol.
5. Save the file as *<Correlator Store filename>*.sec always in the same directory where the Correlator Store is stored.
6. The order of precedence for conditions in the Security file is the Correlator Name, Correlator Template Type and finally the condition for all templates.

   This precedence is arrived at wholly to provide complete security and to have a rigid control on the Correlator Store.
7. To edit the values of Global Constants, use the token identifier GLOBAL_CONSTANT. For example, if you would like to provide permission to the user to edit the values of the Global Constants pi, timeout and createtime, type:

   ```
   GLOBAL_CONSTANT=pi,timeout,createtime
   ```
8. Editing values specific to Correlator Templates Specific changes to values of attributes/variables in Correlators can be made by specifying appropriate token identifiers in the Security file. All identifiers must be specified in upper case. Follow the conventions

provided in the tables below while creating the Security file.

Table 2 **Token Identifiers**

Parameter	Token Identifier
All Correlator Templates	ALL_TEMPLATE
Enhance Correlator Template	ENHANCE
Global Constants	GLOBAL_CONSTANT
Multi Source Correlator Template	MULTI_SOURCE
Rate Correlator Template	RATE
Repeated Correlator Template	REPEATED
Suppress Correlator Template	SUPPRESS
Transient Correlator Template	TRANSIENT
User Defined Correlator Template	USER_DEFINED

Table 3 **Token Identifiers for *TOK_LIST***

Parameter	Token Identifier
Advanced Filter	ADVANCED_FILTER
Alarm Signature	ALARM_SIGNATURE
All parameters	ALL_PARAM
Alter Alarm parameters	ALTER_ALARM
'Clear Alarm' of Transient Correlator Template	CLEAR_ALM
Count of number of alarms of Rate Correlator Template	COUNT
Create Callback function parameters	CRT_CALLBACK
Correlator Description	DESCRIPTION
Discard Callback function section	DIS_CALLBACK
Discard alarm in Rate Correlator Template	DISCARD
Discard Duplicate in Repeated Correlator Template	DISCARD_DUP
Discard Immediately in Repeated Correlator Template	DISCARD_IMD
Discard alarms on set completion in Multisource Correlator Template	DISCARD_ON_SET

Table 3 **Token Identifiers for *TOK_LIST***

Parameter	Token Identifier
Enable Threshold of Transient Correlator Template	ENABLE_THR
Enhance the alarm always of Enhance Correlator Template	ENHANCE_ALWAYS
'Input Function' in User Defined Correlation	INPUT_FUN
Message Key	MESSAGE_KEY
New Alarm parameters	NEW_ALARM
'Output Function' in User Defined Correlation	OUTPUT_FUN
Participate In Other Correlation of Suppress Correlator Template	PAR_OTHCORR
Wait for Set completion in Multi Source Correlator Template	SET
Threshold Count of the Transient Correlator Template	THR_CNT
'Threshold Window' of Transient Correlator Template	THR_WIN
Variables	VARIABLES
Want Original alarm of Enhance Correlator Template	WANT_ORIGINAL
Time period	WINDOW

Configuring Deployment Files

•One Deploy configuration file for every NameSpace file

•The deploy procedure refers to the Deploy Configuration file, which constitutes the following:

- Name of the Correlator Store file after merge
- Path to where the NameSpace file for the associated Correlator Store file(s) is present
- Name of the logfile to which the logs while merging are written into
- ECS Engine Instance to which the merged Correlator Store is loaded
- Logical name of the merged Correlator Store file

 ©2004 Hewlett-Packard Development Company, L.P.

The Operator is finally responsible to ensure that the correlation logic be loaded into the ECS engine. Composer provides this facility by the Deploy feature. However, the Deploy configuration file is maintained by the Developer.

The deploy procedure invokes the `csdeploy` and `csmerge` scripts. These scripts merge the Correlator Store files, remove user Description from the merged Correlator Store and then load the file into the ECS engine. These scripts can also be separately executed from the command prompt.

Creating and Updating Deploy Configuration files

The Deploy Configuration file contains information required by the ECS engine at the time when the Correlator Store files are loaded into the ECS engine. The Deploy Configuration file is an ASCII file to which the above information can be added. A sample of the Deploy Configuration file is shown below:

```
#Following is the default configuration file for the deploy operation from composer GUI in
standalone operator Mode and NNM CMG Mode.

#SUPPORT_DEPLOY_ON_GUI - determines if the deploy should be supported from the GUI.(Not
implemented at the time of this release)
```

```
#FINAL_CS_NAME - path name of the merged Correlator Store to which all the correlator store
files configured in NameSpace.conf file are merged in to.

#NAMESPACE_FILE - path name of the NameSpace.conf configuration file used for deploy operation.

#MERGE_LOG_FILE - path name of the log file where the merge process logs are kept.

#CS_LOGICAL_NAME - logical name of the correlator store loaded in the engine.

#ENGING_INSTANCE - instance number of the ECS Engine to which the correlator store should be
loaded.

SUPPORT_DEPLOY_ON_GUI=yes

FINAL_CS_NAME="$OV_CONF/ecs/circuits/Composer.fs"

NAMESPACE_FILE="$OV_CONF/ecs/CIB/NameSpace.conf"

MERGE_LOG_FILE="$OV_LOG/ecs/csmerge.log"

ENGINE_INSTANCE=1

CS_LOGICAL_NAME=Composer
```

Rules while editing the Deploy Configuration file

- A # sign precedes each comment. All text from the start of the comment to the end of the current line is ignored.
- File locations are always specified with the absolute path. Environment variables can also be used while specifying file locations.
- File locations must be enclosed within quotes.
- No blank space allowed before and after the “=” sign.
- Parameters that must be given values are

  `SUPPORT_DEPLOY_ON_GUI` The user is given the option to choose enabling of deploy of the merged Correlator Store via the GUI. This feature is not supported at the time of this release.

  `FINAL_CS_NAME` Name of the merged Correlator Store file

  `NAMESPACE_FILE` Name of NameSpace file from which the Correlator Stores are picked up from.

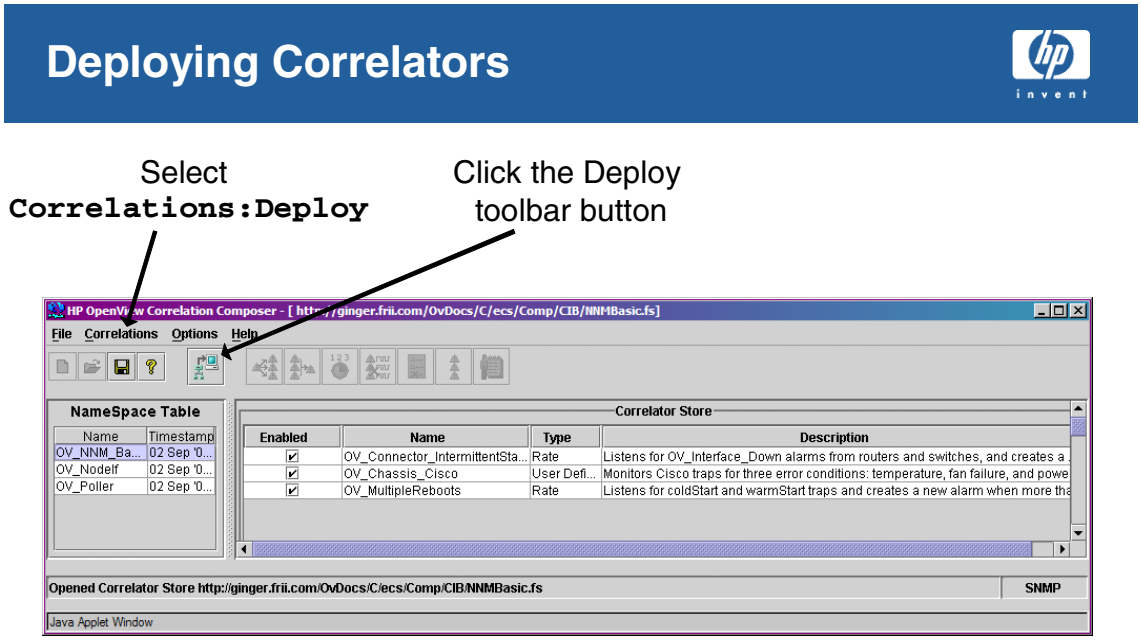
  `MERGE_LOG_FILE` Name of the logfile to which the logs of the Correlator Store merge are written into.

  `ENGINE_INSTANCE` The ECS Engine instance number for which the Correlator Store file will be loaded.

  `CS_LOGICAL_NAME` Logical name of the merged Correlator Store.

IMPORTANT Ensure that the NameSpace file referenced in the Deploy Configuration file is the same file passed with the -N option when Composer is started (see the *ovcomposer* manpage). If the filenames differ, then it leads to creating one set of Correlator Stores and deploying a completely different set of Correlator Stores.

Deploying Correlators



Can also deploy from the command line using a configuration file.

 ©2004 Hewlett-Packard Development Company, L.P.

The Operator loads the correlation logic into the ECS engine. Composer provides this facility by the Deploy feature. However, the Deploy configuration file is maintained by the Developer.

Deploy the Correlator Store

The Operator deploys the Correlator Store into the ECS engine. To load the Correlator Store file into the ECS engine:

1. Ensure that all Correlator Stores have been saved and closed.
2. Do one of the following:
   - Select `Correlations:Deploy`.
   - Click on the `Deploy` icon.

If the deploy is successful, the `Deploy Status` window indicates success.

If an error occurred, the `Deploy Status` window indicates the failure. To view the details of the error, select `[Details]` in the `Deploy Status` window. To close the window, select `[OK]`.

The `Deploy` button in the Operator mode deploys the namespace file pointed to by `deploy.conf`. The `Deploy` button in Developer mode deploys the namespace file pointed to by `Devdeploy.conf`.

Deploy from command prompt

The Correlator Stores can be deployed from the command prompt using the `csdeploy.ovpl` script (in `$OV_BIN` for UNIX, `%OV_BIN%` for Windows). The `csdeploy.ovpl` script refers to the Deploy configuration file required by Composer. To deploy the Correlator Store, type

`csdeploy.ovpl -p` *<Deploy Configuration filename>*

where,

<Deploy Configuration filename> is the name of the Deploy Configuration file. If no filename is specified, the default Deploy Configuration file `$OV_CONF/ecs/CIB/deploy.conf` is selected.

The `csdeploy.ovpl -h` command summarizes the usage of `csdeploy`.

Performance Implications

- Correlations (from ECS Designer) can incur a lot of **pmd** overhead.
- All Correlators (inside the Composer correlation) affect **pmd** less than adding a new correlation.
- De-duplication adds overhead to **ovalarmsrv** rather than **pmd**.

- Test alone and with an event storm

©2004 Hewlett-Packard Development Company, L.P.

Introducing a new ECS or Composer correlator obviously adds more overhead to the `pmd` process and de-duplication adds more overhead to `ovalarmsrv`. This may factor into your decision as to how to implement the reduction.

The performance implications of a new correlator may not be evident with just simple testing. Any new correlation mechanism should be tested under an event storm condition and with all other correlations enabled before determining if performance is acceptable.

Correlator Development Cautions

- Ability to crash pmd
- Holding events too long
- Event storm performance
- Conflict with other correlators

 ©2004 Hewlett-Packard Development Company, L.P.

Developing your own event correlators is a powerful option, but a non-trivial task. You should consider carefully whether this is something you want to do or whether it might be quicker, simpler, and cheaper to contract with a specialist to supply the correlator or correlators you need.

You may contact HP Consulting Service or other system integrators for assistance.

What can go wrong in a Composer correlator?

Although the Composer development interface makes it easy to configure a template to create a new correlator, practical experience has shown that it may take a significant amount of time and expertise to debug and troubleshoot a Composer correlator.

The following is a brief description of things that can go wrong:

- Crashing the `pmd` process with C or Perl callouts that fail

  Synchronous functions and perl scripts (this includes all Composer callbacks) are executed from within the `pmd` process. If the function or script aborts, it will abort the `pmd` process. When this happens it usually requires a restart of NNM (`ovstop`/`ovstart`).

- Retaining events for too long

Released new events for a particular correlator that are marked to be fed back into Composer may be retained by other correlators. As a result they will not be released back to OVEvent and appear in the browsers when expected.

- Performance problems in handling event storms in `pmd`

  New correlators that perform external synchronous functions or scripts can slow down (block) the `pmd` significantly and seriously impact the ability to handle an event storm.

- Breaking other correlators because of interaction

  The input events or released events may overlap with those of existing NNM product correlators. This may break or impair existing correlators.

Composer References

Print out and read the following reference materials:

- Access the following pdf format manuals from the NNM main window, select `Help:Online Manuals`:
  - — *HP OpenView Correlation Composer's Guide*
  - — *Managing Your Network*, Event Reduction Capabilities chapter
- Find the *csmerge* reference page in NNM's online help (or the UNIX manpage) for information about Correlation Composer's tool for merging new correlators (that are developed and tested in an isolated environment) back into the NNM correlator fact store file on your production NNM management station.
- Access the following pdf-format white paper file: *Developing_NNM_Event_Reduction.pdf*

  Windows: *install_dir*`\Doc\WhitePapers\`

  UNIX: `$OV_DOC/WhitePapers/`

  This information explains how to ensure that you do not break existing event reduction implementations. It also explains the process required for carefully merging your new correlators into the NNM correlator fact store file.
- The *TroubleshootingEventReduction.txt* file explains how to create event logs and play them back into the NNM event correlation system for testing purposes:

  Windows: *install_dir*`\contrib\ecs\`

  UNIX: `$OV_CONTRIB/ecs/`

Lab Exercises

Lab Exercises

- Start the Composer user interface
- Review existing correlators

©2004 Hewlett-Packard Development Company, L.P.

1. Start the Composer developer interface.

2. Open the NNM Composer configuration file in the Composer Information Base for the basic correlators.

3. Briefly examine each template definition area. Cancel out of each screen without saving any changes.
4. Examine the shipped Suppress correlator OV_Connector_IntermittentStatus. Do not make

any changes.

a. Double-click on the correlator.

b. Review the Description tab. This is the help for the correlator. Which parameter is modifiable by customers?

c. Select the Definition tab. Which specific trap numbers are processed by this correlator?

d. What other data is used to further qualify which events are processed?

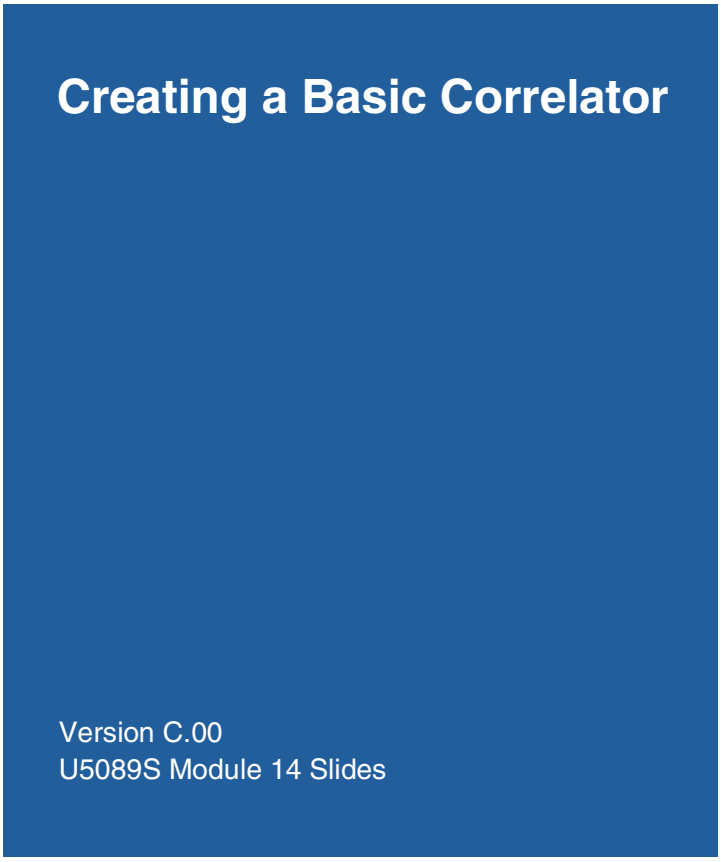
e. What is the content of that varbind?

f. Cancel out of the Composer screen.

5. Exit the Composer interface.

14 Creating a Basic Correlator

Module Objectives



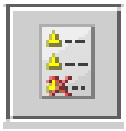
At the completion of this module, you will be able to:

- Define a Suppress correlator.
- Specify an Alarm Signature.

Suppress Correlator Template

Suppress Correlator Template

- The Suppress Correlator Template can be used to discard a specific category of events.
- Events that match the conditions in the Alarm Signature (and Advanced Filter, if specified) are discarded.



©2004 Hewlett-Packard Development Company, L.P.

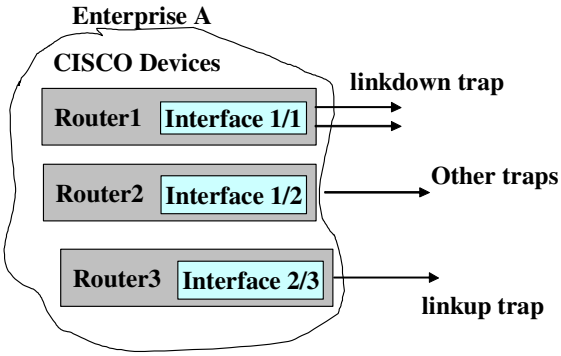
Suppress correlator is used when a specific category of events needs to be discarded, so that they never appear in the NNM Alarm Browser. It is your best option when you want to base the decision to discard on more than the packet header. Otherwise, you can use LogOnly or de-duplication.

Events that match all the conditions in both the Alarm Signatures and Advanced Filter are discarded.

Suppress Example - Problem Statement

Suppress Example - Problem Statement

- In Enterprise A, some interfaces are used for testing.
- You want to block all linkup and linkdown traps.



©2004 Hewlett-Packard Development Company, L.P.

In an Enterprise A, there are multiple interfaces, Each of these interfaces emit link_up and link_down traps. The requirement is to suppress all of these traps emitted.

Suppress Example - Sample PDU

Suppress Example - Sample PDU

•The linkup/linkdown Trap-PDU looks like:

```
Trap-PDU{
  enterprise {1 2 3 4 5 6},
  agent-addr internet : "\x0A\x00\x01\x7F",
  generic-trap 6,
  specific-trap 2,
  time-stamp 414746291,
  variable-bindings{
       {
       name {1 2 3 4 1},
       value simple : number : 3
       },
       name {1 2 3 4 1}
       value simple : string : "1/1"
       },
       {
       name {1 2 3 4 1},
       value simple : number : 1
       }
  }
}
```

U5089S ©2004 Hewlett-Packard Development Company, L.P. 4

Before you proceed to configure this correlator, what do you need to know?

- What type of template should be used?

  To block or discard traps, use the Suppress correlator template.
- How do you identify which events are to be suppressed?

  All link_up/link_down traps have the following attributes which identify them

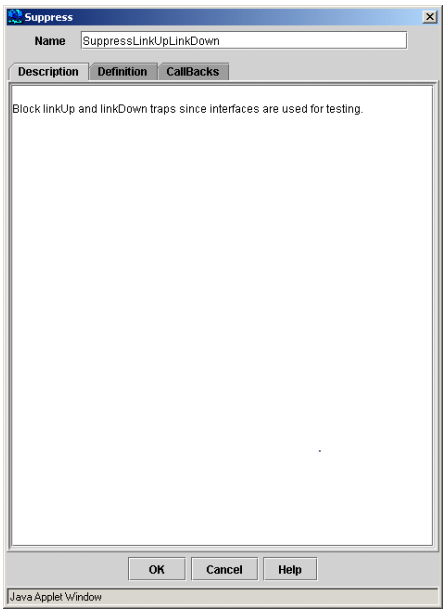
  — Enterprise-id begins with 1.2.3.4.5.6

  — generic-trap is set to 6

  — specific-trap is set to 2

Creating a Correlator

Creating a Correlator



- Select Description tab
- Name the correlator
  - Letters, digits, underscore
  - Must be unique
- Type comments
  - Purpose of correlator
  - Input events
  - Actions taken
  - Configurable parameters

©2004 Hewlett-Packard Development Company, L.P.

To create a new Correlator

1. Click a toolbar icon or select `Correlation:Correlation Templates`. From the menu displayed select the Correlator Template you would like to create.

   The Correlator configuration is divided into three or four tabs:

   - Description
   - Alarm Definition
   - New Alarm Creation
   - Callback Functions

2. In the Correlator window on the Description tab, enter the Correlator Name in the Name text box. Note that the correlator will be referred by this name throughout the Correlator Store.

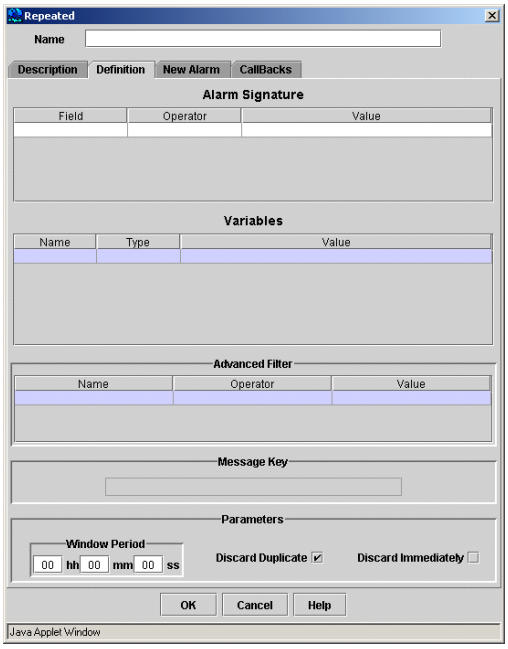
   Naming Restrictions

   Use names that start with a letter and contain only letters, digits and underscore(_). Usage of special characters like!, @, #, $, !, ^, & and * is not allowed. For easier reference let the name indicate the problem type. For example “Generator_OFF” is a valid name.

3. Enter the Description of the correlator. The description can briefly state what the cause of the event can be and what the Correlator is expected to do.

Select Incoming Events

Select Incoming Events



U5089S ©2004 Hewlett-Packard Development Company, L.P. 6

The first section in the Correlator configuration is the Alarm Definition. The Alarm Definition consists of the subsections described below.

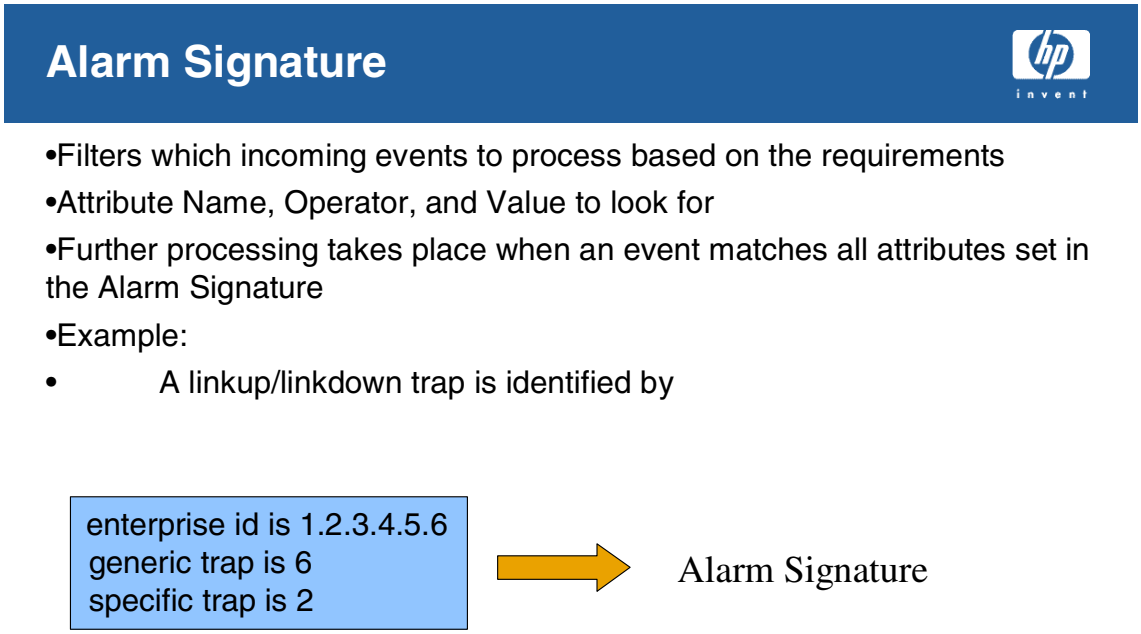
1. Define the Alarm Signature. All correlators contain an Alarm Signature.
   a. Alarm Signature is a set of values that specifies a filter. Select the Field name from the drop-down list.
   b. Select the Operator value from the drop-down list.
   c. Enter/select the value of the field to be compared against. To enter a value, click in the value cell and type in the value.

      To select a Global Constant, double click in the Value cell. A pop up menu displaying the previously defined constants is displayed. Choose the constant from the menu displayed.
2. Declare the Variables. Variables are optional in correlators, but most of the user interface requires named variables; you can not type in values in other areas of the interface.
   a. Variables are names with associated values that can be used inside the Correlator definition. Enter a name for the variable.
   b. Select the variable type from the drop down menu. For a list of variable types supported refer to the *HP OpenView Correlation Composer's Guide*.

c. Depending on the variable type chosen, the value has to be entered. Click inside the Value cell to enter the value. For a detailed procedure on how to enter values for the different variable types refer to the *HP OpenView Correlation Composer's Guide*

3. Define the Message Key. The Message Key is required for Rate, Repeated, Transient, and MultiSource correlators. It does not appear on Suppress or Enhance correlator templates. To enter the Message Key, click in the Message Key box. A pop up menu displays all possible values. Select the variable/attribute that you would like to declare as Message Key.
4. Define the Advanced Filter (if any). The Advanced Filter is an optional additional specification of which events are handled by this correlator. It allows you to access information beyond what is directly contained in the event attributes. Select the Attributes, Operator and the Value from the pop up menu displayed in the respective columns. This is a non-editable field. All values to be displayed must have been previously defined.
5. Enter the Parameters.
6. Click [OK] to complete the creation of the Correlator.

Alarm Signature



The Alarm Signature (primary filter) is the first level of filtering performed by the Composer. Alarms whose attributes match the attribute specification specified in the Alarm Signature are processed further. The Alarm Signature is a set of tuples constituting Attribute name, Operator and Value.

The standard Event Attributes are listed in Appendix C, “Event Attributes,” of the *HP OpenView Correlation Composer’s Guide*. Event Attributes vary depending on the Event Type selected. To customize the list of attributes, refer to the *HP OpenView Correlation Composer’s Guide*.

Table 4 Event Attributes Available in NNM

Message Attribute	Type	Description
enterprise	Object ID	Identifies the network management subsystem that generated the trap.
generic-trap	integer	One of the predefined values in the definition. Values must be between 1 and 6.

Table 4 Event Attributes Available in NNM

Message Attribute	Type	Description
specific-trap	integer	A code that indicates the nature of the trap more specifically than the generic-trap number. Specific trap numbers are defined by the owning enterprise and are meaningful only in conjunction with the enterprise attribute.
time-stamp	integer	The number of time ticks, in hundredths of a second, between the last intialization of the device and the generation of the trap. Generally meaningless for correlation purposes.
variable bindings	string, integer	Additional information about the trap. The contents of this field is dependent on the enterprise ID and specific-trap values.
agent-addr	string in dot notation	The network address of the object that generated the trap in the form of an ECDL tuple.

Operators

Operators

	Operator	Description	Value Must Be One Of
*	=	Equals	Integer, Float, String or Object ID
	!=	Not Equal	Integer, Float, String or Object ID
	<, >	Less Than, Greater Than	Integer, Float
	<=, >=	Less than or equal, Greater than or equal	Integer, Float
*	matches	The pattern specified by value is matched against the value contained in the attribute	Integer, Float, String or Object ID
	does NOT match	The pattern specified by the value is NOT present in the attribute	Integer, Float, String or Object ID
*	is in list	Equals the list of values returned by the attribute	List. For example, a is in the list [a, b, c, d]
	is NOT in list	Is not equal to any of the values specified in the list	List

* Most relevant to Alarm Signature

U5089S ©2004 Hewlett-Packard Development Company, L.P. 8

In addition to the customary operators, you may use:

- `matches` - If the pattern represented by Value is present in the attribute. For regular pattern matching expressions refer to Appendix D:Pattern Matching in the *HP OpenView Correlation Composer's Guide*.

  Examples

  1."1234510" (which is the value contained in the specified attribute) matches "10". Here the string pattern "10" is searched for in the string "1234510".

  2."1234510" matches "^10" returns False. If the value extracted from the attribute is not a string, then the value is converted to a string and the pattern is matched.

  3.If the attribute enterprise contains an Object ID, 1.2.3.4.5.6 and the requirement is to discard traps with an enterprise Object ID of 1.2.3.4, then the following expression would meet the above requirement enterprise matches 1.2.3.4. Internally 1.2.3.4 is converted to a string "1.2.3.4" and this pattern is looked for in the string version of the enterprise. This however would also match an enterprise id of 5.6.1.2.3.4 which is not the requirement. The following ensures that the correct pattern is matched:

  enterprise matches "^1.2.3.4"

  Note that the pattern is given as a string.

- `does NOT match` - The string pattern represented by Value is not present in the attribute. The

does NOT match operator is the opposite of the match operator.

Example - The Value “1020” must not be present in the attribute.

- `is in list` - The value entered must be present in the list of values.

  Example - a is in list [a, b, c, d]

- `is NOT in list` - The value entered must not be present in the list of values entered.

NOTE In the NNM environment:

The attribute `agent-addr` in the SNMP trap is represented as a string in the ‘.’ notation. For example, if the `agent-addr` is passed to a function, it will be passed as “a.b.c.d” and NOT a.b.c.d. If the `agent-addr` needs to be set while creating/altering an event, the variable carrying the `agent-addr` should also be in the same format. For example, to set the `agent-addr` to an IP address of 15.10.76.143, define a variable whose value is a string like “15.10.76.143”.

Suppress Example - Solution

Suppress Example - Solution

•Map the trap to an Alarm Signature

Linkup/Down Trap

```
Trap-PDU{
  enterprise{1 2 3 4 5 6},
  agent-addr internet :
        "\x0A\x00\x01\x7F",
  generic-trap 6,
  specific-trap 2,
  time-stamp 414746291,
  variable-bindings{
  ...}
```

Suppress

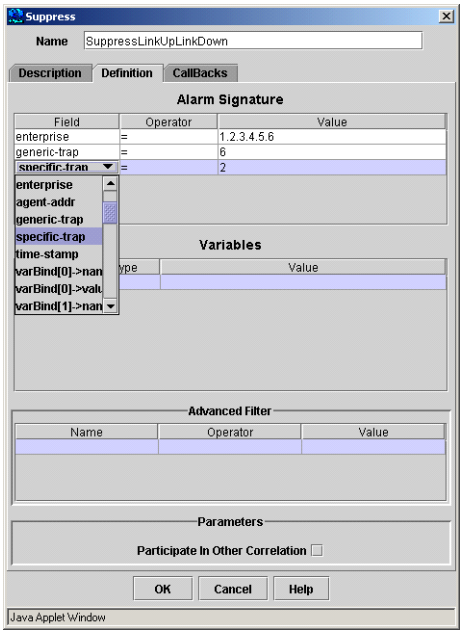
Alarm Signature

Field	Operator	Value
enterprise	=	1.2.3.4.5.6
generic-trap	=	6
specific-trap	=	2

©2004 Hewlett-Packard Development Company, L.P.

Suppress Example - Definition Window

Definition Window



- Select the Definition tab.
- Right-click and select Add to add a row.
- Select field name from the list.
- Select operator from the list.
- Type a value.

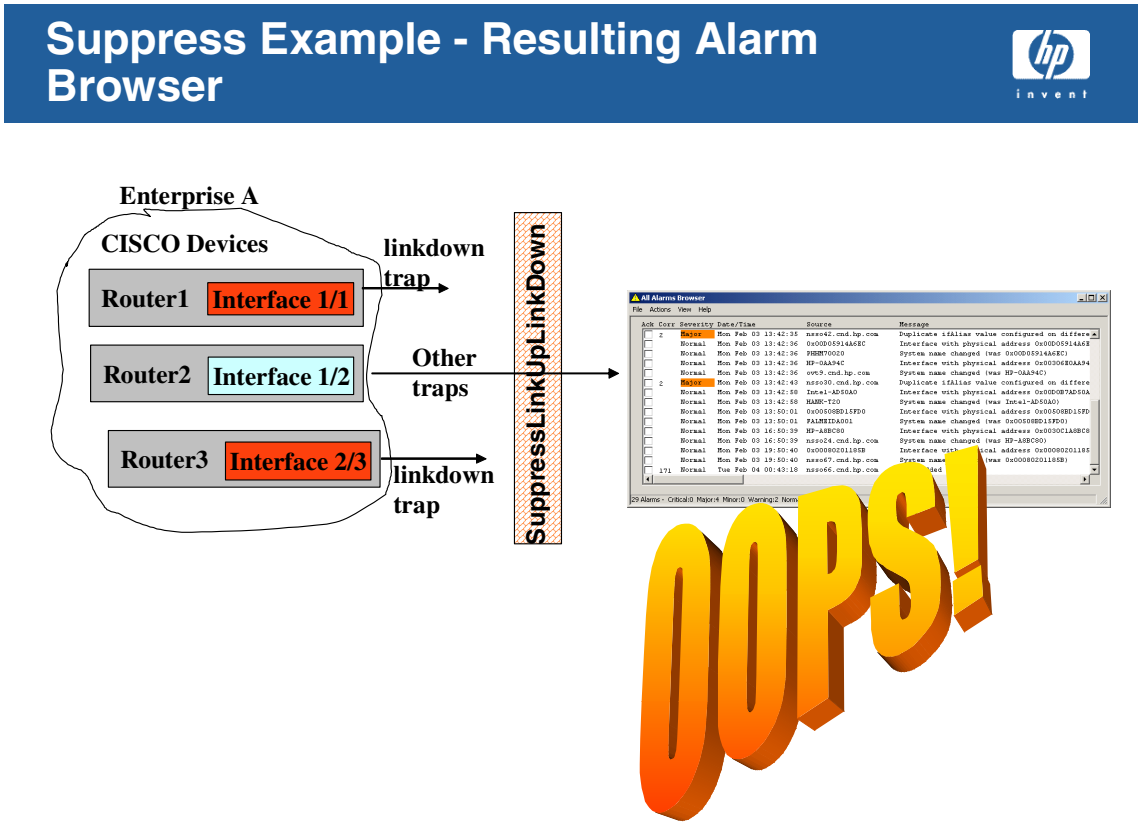
U5089S ©2004 Hewlett-Packard Development Company, L.P. 10

The Suppress Correlator Template has been identified to perform this kind of correlation. Follow the procedure given below to configure the Correlator.

1. Select `Correlations:Correlator Templates->Suppress` from the Correlator Store window. You can also click the Suppress icon in the Correlator Templates toolbar.
2. Enter the name of the Correlator in the Name text field.
3. Enter the description of the Correlator. The Description can briefly state what the Correlator is expected to do.
4. Click the Definition tab to display the Alarm Definition panel. Set the following values to define the Alarm Signature
   - `enterprise = 1.2.3.4.5.6`
   - `generic-trap = 6`
   - `specific-trap = 2`

   While defining variables in NNM environment, specification of Object IDs do not have the leading dot. For example 1.2.3.4 is valid while .1.2.3.4 is not.
5. Click [`OK`] to complete the Correlator. Notice that the correlation you have just defined is displayed in the Correlator Store table.

Suppress Example - Resulting Alarm Browser

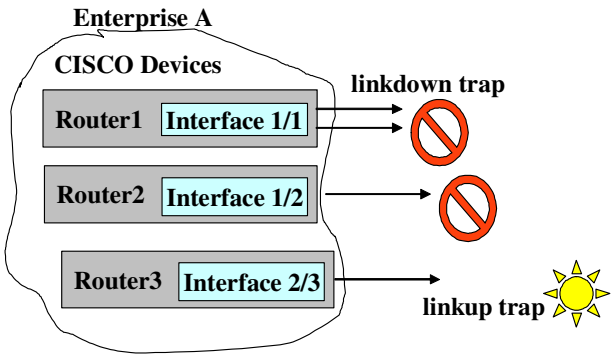


All linkdown traps from all interfaces, whether used for testing or not, are being suppressed. That was not the original intention, so we need to iterate on the requirements.

Suppress Example Update - Problem Restatement

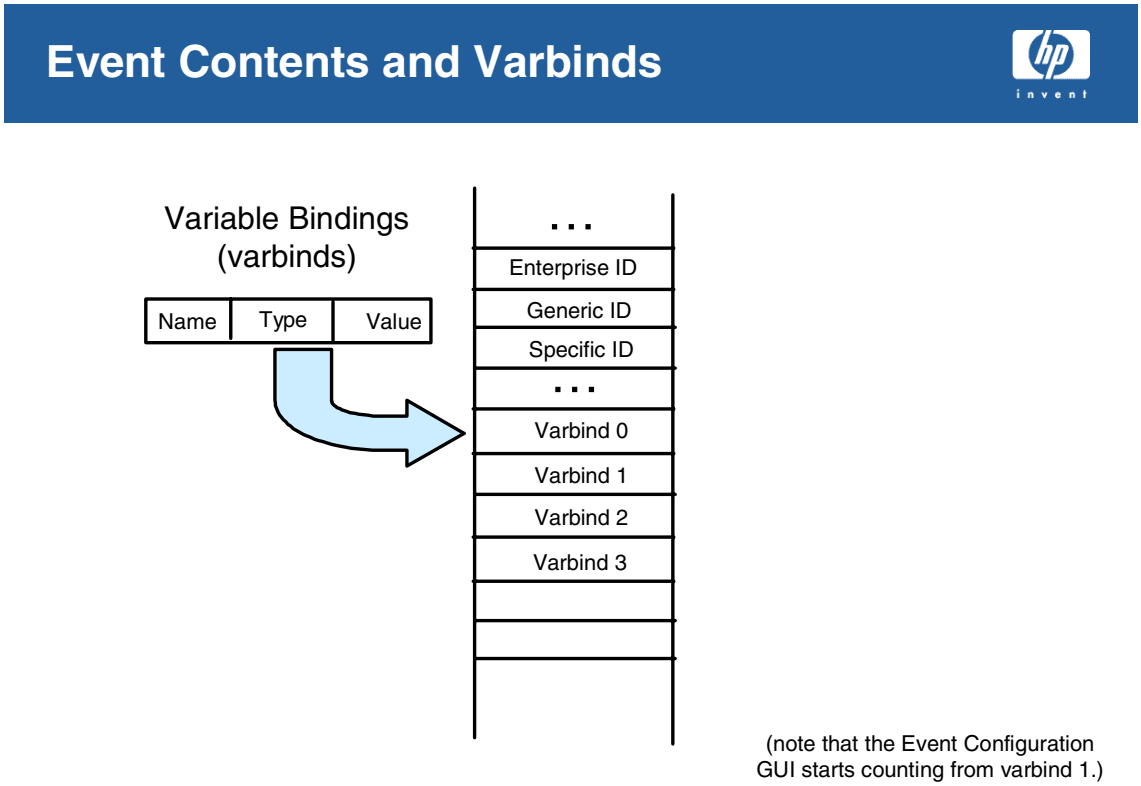
Suppress Example Update - Problem Restatement

- Customer is missing too many alarms!
- Actually, only interfaces 1/1 and 1/2 are used for testing.
- (Original specification was incomplete.)
- Want to block all linkup and linkdown traps from these interfaces.



©2004 Hewlett-Packard Development Company, L.P.

Event Contents and Varbinds



In addition to the information in the header of the packet, you can access information contained in the packet body.

Each item of data in the packet has a variable name (in dot notation), a data type, and a value. The combination of the name-type-value is called a variable binding, or varbind.

Varbinds are referred to in numerical order. Composer refers to the first varbind as varbind[0]. In the NNM Event Configuration GUI, where you configure messages for the Alarm Browser, varbinds are referred to starting with number 1.

Suppress Example Update - Sample PDU

Suppress Example Update - Sample PDU

•The linkup/linkdown Trap-PDU looks like:

```
Trap-PDU{
  enterprise {1 2 3 4 5 6},
  agent-addr internet : "\x0A\x00\x01\x7F",
  generic-trap 6,
  specific-trap 2,
  time-stamp 414746291,
  variable-bindings{
       {
       name {1 2 3 4 1},
       value simple : number : 3          → varbind0 value
       },
       name {1 2 3 4 1}
       value simple : string : "1/1"      → varbind1-Interface number
       },
       {
       name {1 2 3 4 1},
       value simple : number : 1          → varbind2 value
       }
  }
}
```

U5089S ©2004 Hewlett-Packard Development Company, L.P. 14

Looking at this sample PDU, you can see sample values for the variable bindings. The first varbind is numbered varbind.0. The one referred to as varbind.1 contains the interface number as a string.

- Which correlation template should be used?

  It is still appropriate to use a Suppress correlator.

- How do you identify which link_up and link_down traps come from testing interfaces?

  The variable-bindings[1].value has the string 1/1 or 1/2. All other interfaces are for production in this customer's environment.

Suppress Example Update - Solution

Suppress Example Update - Solution

- Step 3. Map the trap to Alarm Signature
  - Include the varbind requirement

Linkup/Down Trap

```
Trap-PDU{
  enterprise{1 2 3 4 5 6},
  agent-addr internet :
        "\x0A\x00\x01\x7F",
  generic-trap 6,
  specific-trap 2,
  time-stamp 414746291,
  variable-bindings{
  ...}
```

Suppress

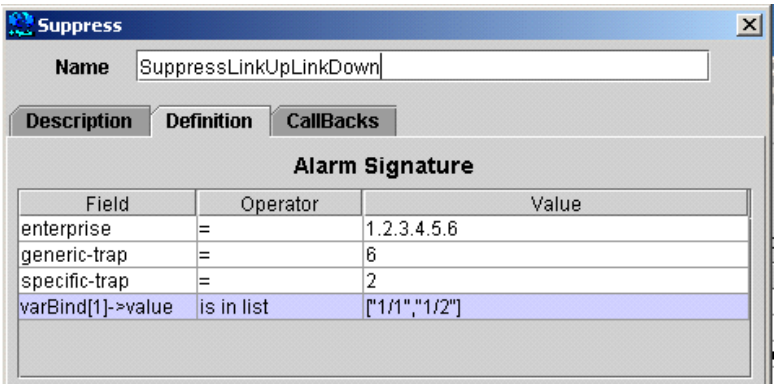
Alarm Signature

Field	Operator	Value
enterprise	=	1.2.3.4.5.6
generic-trap	=	6
specific-trap	=	2
varbind[1]->value	Is in list	[“1/1”,”1/2”]

©2004 Hewlett-Packard Development Company, L.P.

Suppress Example Update - Definition Window

Definition Window



©2004 Hewlett-Packard Development Company, L.P.

1. Double-click the correlator created in Suppress Example.
2. Click the Definition tab to display the Alarm Definition panel. Add the following value to define the Alarm Signature
   - `variable-Binding[1].value is in list ["1/1","1/2"]`

     Ensure all values are enclosed in square brackets, individual values enclosed in double quotes and separated by commas.
3. Click [OK] to complete the Correlator.

Lab Exercises

- Define and create a Suppress correlator.
- Test its operation.

©2004 Hewlett-Packard Development Company, L.P.

Lab Case: Suppress Correlation

Movement traps in general need investigation. However if the movement event are from exchanges emitted from the City offices, they can be discarded as there is always movement and the Alarm Browser is filled with these events. The requirement is to discard all movement events emitted from the City offices.

A sample SNMP trap PDU appears in a log as:

```
Trap PDU {
enterprise{1 2 3 4 999},
agent-addr internet : "\x0A\x00\x01\x7F",
generic-trap 6,
specific-trap 1,
time-stamp 414746291,
```

```
variable-bindings{
{
name {1 3 6 1 4 1 11 2 17 2 1 0},
value simple : number 2
},
{
name {13 6 1 4 1 11 2 17 2 2 0},
value simple : "City-Bangalore"
}
{
name {1 3 6 1 4 11 2 17 2 17 0},
value simple : string : "There is movement"
}
}
}
```

What do you need to know from the requirements?

1. How do you identify which events are to be Suppressed?

View Pre-Correlator Results

1. Event configurations for events used in the labs for this course are provided in `$OV_CONTRIB/OVTraining/NNM3/composerLab_trapd.conf`. To add these to your system configuration, type the commands:

   ```
   cd $OV_CONTRIB/OVTraining/NNM3
   xnmevents -merge composerLab_trapd.conf
   xnmevents &
   ```

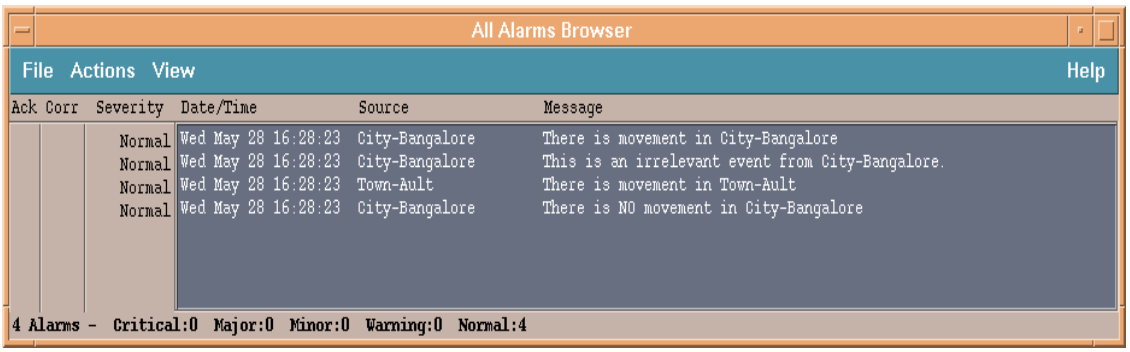
2. On Windows, correct the path for your system to include `%OV_BIN%\Perl\bin` if necessary.
   a. Right click My Computer on the desktop.
   b. Select Properties.
   c. Select the Advanced Tab.
   d. Click [`Environment Variables`].
   e. In the System variables window, select Path.
   f. Click [`Edit`].
   g. At the end of the Variable Value, append `;C:\Program Files\HP OpenView\bin\Perl\bin`
   h. Click [`OK`].
   i. Click [`OK`].
   j. Click [`OK`].
   k. Log out and log back in.
3. A sample event stream from this environment is available for testing. Copy the following file from `$OV_CONTRIB/OVTraining/NNM3` to `$OV_CONTRIB/ecs`:

- `movement.evt`: sends the following events:
  - — Movement from a city.
  - — Wrong specific ID.
  - — “City” replaced by “Town”.
  - — String contains “There is NO movement”.

4. Open the Composer Labs Alarms Browser.
5. Empty all old alarms by selecting `Actions:Delete->All Alarms in Category`.
6. Type the commands:

```
cd $OV_CONTRIB/ecs
ecsevgen -n movement.evt
```

7. Review the alarms in the Alarm Browser.



8. To mark which alarms came from this pre-test run, select `Actions:Acknowledge->All Alarms in Category`.

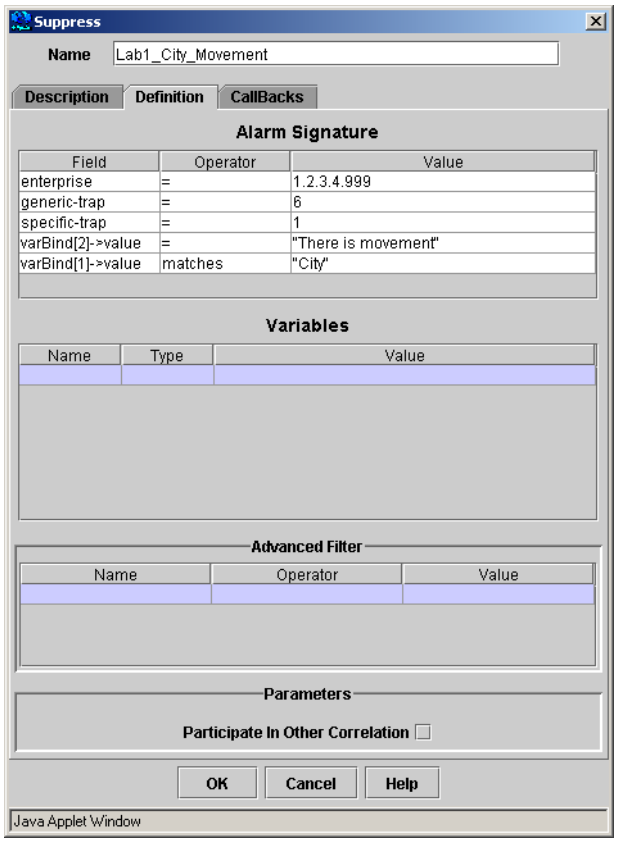
Configure the Correlation

Follow the procedure given below to define the Suppress Correlator Template:

9. Start the Composer development interface.
10. Create a new correlator store with SNMP events.
11. Select `Correlations:Correlator Templates->Suppress` from the Correlator Store window. The Suppress Correlator Template window opens.

    You can also click on the Suppress icon in the Correlator Templates toolbar.
12. Enter the name of the Correlator in the Name text field.
13. Enter the description of the Correlator. The Description can briefly state what the Correlator is expected to do.
14. Set the following values to define the Alarm Signature
    - enterprise = 1.2.3.4.999
    - generic-trap = 6
    - specific-trap = 1
    - variable-binding [2].value = “There is movement”
    - variable-binding [1].value matches “City”

15. Click [OK] to complete the Correlator. Notice that the correlation you have just defined is displayed in the Correlator Store table.



16. Save your Correlator Store in the `$OV_CONF/ecs/CIB` directory.
17. Exit the Composer developer interface.
18. Review the files created by saving the Correlator Store.
19. Provide Operator access to your correlator using the existing files.
20. As an Operator, enable and deploy your correlation.

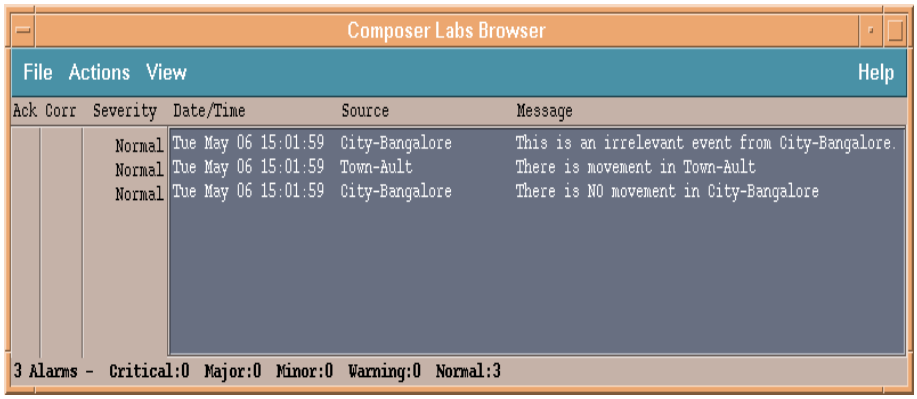
Test Your Correlator

21. Watch your Composer Labs Alarms Browser to watch the operation of your correlation.
22. Type the commands:

```
cd $OV_CONTRIB/ecs

ecsevgen -n movement.evt
```

23. Examine the Alarms Browser. You should see no indication of the first event, which was suppressed. The rest of the events fail to suppress and show in the Alarm Browser. Look at

the information there about the actual contents of the various varbinds.

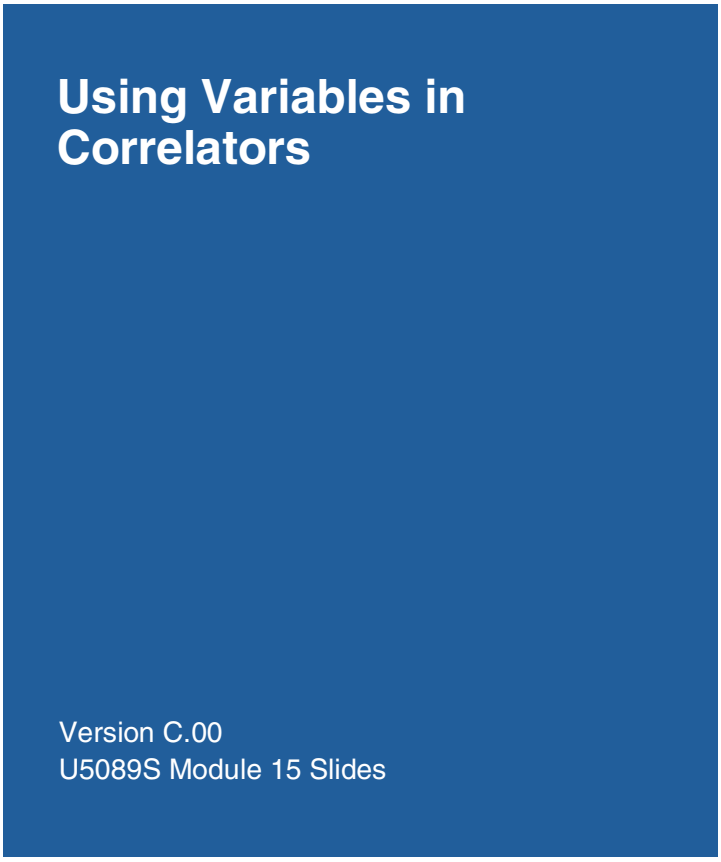


a. update the line for the suppress lab to include the subdirectory.

24.

15 Using Variables in Correlators

Module Objectives



At the completion of this module, you will be able to:

- Define a constant variable.
- Specify an Advanced Filter.
- Look up variable data in a data store.
- Define an Enhance correlator.
- Access event varbinds from within a correlator.
- Combine strings to create a variable.
- Define a new event to be generated.
- Create an Extract variable to get data contained in a varbind.

Variable Types

- Variables are names assigned to values.

- Constant - Value entered is bound to the variable name
- Lookup - Represents data returned from a datastore lookup
- Combine - A string concatenation of two or more variables
- Extract - A matched string is extracted from event and assigned to a variable
- Function - Represents data returned from a function

- Evaluated only when used

U5089S ©2004 Hewlett-Packard Development Company, L.P. 2

Variables are names given to values to be used while defining Correlators or Global Constants. All attributes (header information and varbinds) in an event can be accessed as variables where the variable name is the attribute name, such as agent-addr or varbind[1].value.

- Constant - Constant values are used for reference while defining a Correlator. The variable name is bound to the value specified in the value field.

  Example: A variable `ErrStr` is bound to the value “Temperature High. Check for A/C Failure”. The variable `ErrStr` can be used locally within the Correlator under which the variable is declared.

- Lookup - Values from a datastore can be queried and bound to variables using the Lookup operator. Parameters for the Lookup are one or more variables. The values referred to by these variables are concatenated and the resulting value is used as the key in the Lookup to the datastore.

- Combine - Variables can be combined to form a new variable by concatenating variables to a single string.

  Example:

  Assume the following variables defined with these values:

  – `a constant ‘Hello’`

  – `b constant ‘World’`

– c constant 'Rate is'

– d constant 10

– e constant 20

Combine [a b] results in "Hello World".

Combine [c <AlarmCnt>] results in "Rate is <AlarmCnt>".

Combine[d e] results in results in 1020.

- Extract - Sub-strings within an event attribute can be extracted.

  Example: <*.err_text> - <#.link_num> on Signalling Set <#.set_num> matches a message such as "Link Failure - 10 on Signalling Set 2" and assigns Link Failure to err_text, 10 to link_num, 2 to set_num.

- Function - The return value of a function can be bound to a name. Functions can be called synchronously or asynchronously.

  Example - Assume a variable X which is bound to the return value of a function, say xyz. Also assume that xyz returns an integer. In this case the variable X would be bound to the integer value returned by the function xyz.

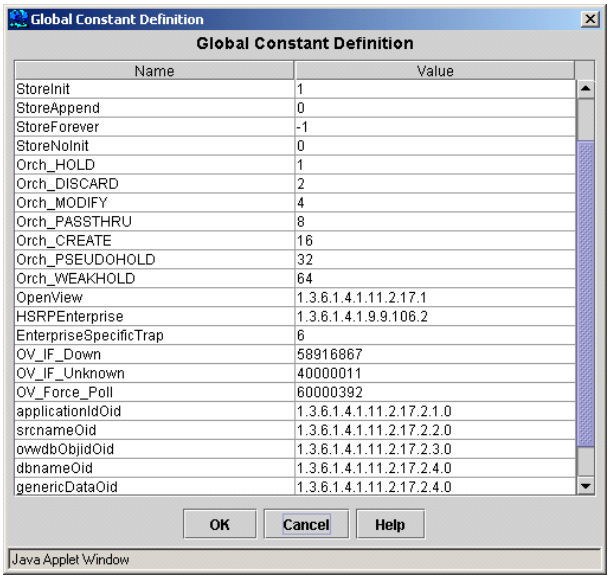
  In cases where functions return more than a single value, individual elements can be accessed via the Built-In function getByIndex. For example if a function, myFunction, returns 10, 20, 30, this is bound to a variable, myVariable. To access individual elements use the Built-in function getByIndex.

There are no explicit type conversion mechanisms available. For example, if you try to combine an integer and string, the resultant will take string type.

Scope of Variables

Scope of Variables

- The scope of a variable is limited to the correlator in which the variable is defined.
- Global Constants can be defined and used anywhere inside the Correlator Store



 ©2004 Hewlett-Packard Development Company, L.P.

The parameters that you enter to define the named value pairs can refer to the global constants. For example, a named value called DEVICE that you have defined can have the event attribute “eventInfo.notificationIdentifier”.

To define a Global Constant:

1. Select `Correlations:Global Constants` from the Main Menu.
2. Enter the Name and Value.

   These values can be referenced anywhere inside the Correlator Store.

   The value types are integer, float, string, and OID.
3. To add more Global Constants, right click the mouse button and select [`Add`]. A new row is added to the Global Constants table.
4. When you have finished defining Global Constants, click [`OK`] to close the window.

To delete any Global Constant, you can do one of the following:

- select the Global Constant, right click the mouse button and select [`Delete`].
- select the Global Constant, and press the [`Delete`] key.

Advanced Filter

- Further filter incoming events
- Name, Operator and Value
- Typically used to define filters based on external factors like topology
- Not a mandatory section
- Example
  - Generate a new event when a Router_Failure trap is received from a Core router.
  - The failure trap doesn't contain sufficient information to determine if the source is a Core router.

 ©2004 Hewlett-Packard Development Company, L.P.

Events coming into a Correlator can be further filtered based on the Advanced Filter condition. The Advanced Filter contains Name, Operator, Value structures. The Advance Filter is evaluated after events have passed the primary filter (that is, the Alarm Signature).

Example of Advanced Filter

The requirement is to generate a new event if a Router_Failure trap is received from a core router.

Just by examining the trap attributes, Composer cannot deduce whether the trap is from a core router. To solve this problem, define a variable, `isCoreRouter`. Let this variable be bound to the return value of a function, `GetIsRouter`. This function takes the `agent-address` as its parameter and returns 1 if the router is a core router, and returns 0 otherwise. In the Advanced Filter, define a Correlator that checks if the variable, `isCoreRouter` is set to 1 which ensures that the Correlator is applied only to Core routers' traps.

Variable Example

•Problem Statement

- Suppress all Interface_Down events from a list of equipment undergoing scheduled maintenance.

•Solution Design

- Filter Interface_Down events

- Recognize events from equipment undergoing scheduled maintenance
  - Declare a variable that holds the IP address of all equipment under scheduled maintenance
  - Compare the agent-address in the event with the variable

 ©2004 Hewlett-Packard Development Company, L.P.

Following from the previous examples, events can be suppressed based on the primary filter – Alarm Signature. This eliminates events from all devices equally.

However the requirement in this case is to discard or suppress events from specific equipment undergoing scheduled maintenance. Hence a mechanism is required to identify events from these devices and filter based on this parameter.

You could add the `agent-addr` of a device to the Alarm Signature, but then the correlator would only work for that one device, whereas many devices may be scheduled for maintenance.

Before you begin to configure this correlator, what do you need to know?

1. How do you recognize the Interface_Down events?

   All Interface_Down events have the following attributes which identify them

   - enterprise is set to 1.2.3.4
   - generic-trap is set to 6
   - specific-trap is 2

   Note that providing this information in the Alarm Signature selects all Interface_Down events.

2. How do you recognize events from equipment undergoing scheduled maintenance?

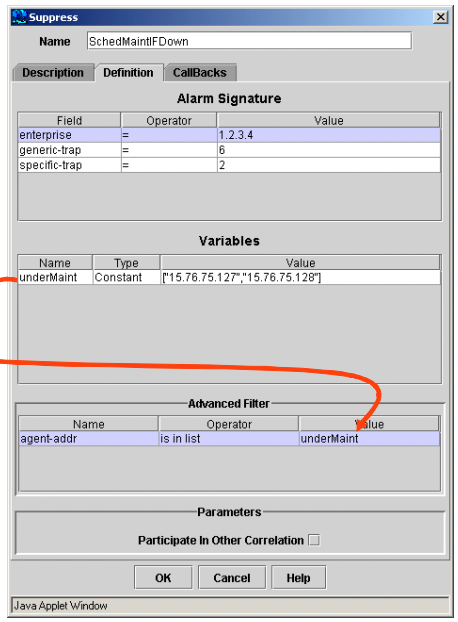
The agent-address identifies the source of the event. If the agent-address in the event is equal to the IP address of the equipment, then it is from the same equipment.

3. How do you select only those events?

   Declare a variable, `underMaint`, to hold the IP address list. The Advanced Filter can compare the incoming event to the values in the named list.

Variable Example: Definition Window

Definition Window



- Right-click in Variables area and select Add.
- Type the variable name.
- Select the type from the list.
- Type desired the value.

- Click on the Advanced Filter name cell and select from the list.
- Select operator from the list.
- Select value from the list.

In this example, the agent-address specified in the incoming event is compared with the variable `underMaint`. If the values are equal, then the equipment from which the event has been emitted is under maintenance.

Follow the procedure given below to configure this correlator

1. Select `Correlations:Correlator Templates->Suppress` from the Correlator Store window. You can also click the Suppress icon in the Correlator Templates toolbar.
2. Enter the name of the Correlator in the Name text field.
3. Enter the description of the Correlator. The Description can briefly state what the Correlator is expected to do.
4. Click the Definition tab to display the Alarm Definition panel.

   Set the following values to define the Alarm Signature:

   - `enterprise = 1.2.3.4`
   - `generic-trap = 6`
   - `specific-trap = 2`
5. Create a variable of type Constant `underMaint`.

a. In the Name cell, enter the name of the variable `underMaint`.

b. Click in the Value cell and enter the values, in this case the IP addresses of all equipment under maintenance.

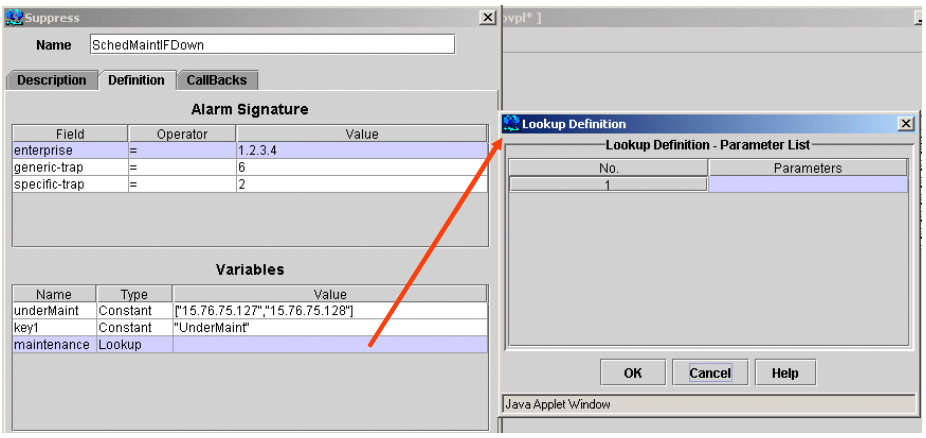
6. In the Advanced Filter section, select the following from the pop-up lists:

- `Name = agent-addr`
- `Operator = =`
- `Value = underMaint`

7. Click [`OK`] to complete the Correlator. Notice that the correlation you have just defined is displayed in the Correlator Store table.

Configuring a Lookup Variable

Configuring a Lookup Variable



- Add variable row, give it a name.
- Select Lookup from Type list.
- Click in value cell to open Lookup Definition dialog.
- Select parameter from list.

 ©2004 Hewlett-Packard Development Company, L.P.

1. Create a variable to hold the key value.
2. Create another variable to do a Lookup on the key variable.
3. Click in the value cell. The Lookup Definition window is displayed.
4. The Parameter list has to be filled with the keys to concatenate for the Lookup to look up the variables. Click the Parameters cell. A pop up menu displaying all attributes, pre-defined variables and Global Constants is displayed.
5. Select the key from the pop up menu.
6. Click [OK].

Editing the Datastore

- Manually place lines in **$OV_CONF/ecs/circuits/Composer.ds**
- Syntax is

```
#path#date#version#0
ADD DATA (keyValue, returnValue)
ADD DATA (keyValue, returnValue)
```

- Activate the changes by disabling and re-enabling Composer in the ECS GUI.

 ©2004 Hewlett-Packard Development Company, L.P.

All Composer correlators share the datastore located in `$OV_CONF/ecs/circuits/Composer.ds`. A dummy file is shipped with the product, to which you may add lines.

The format of the datastore is `ADD DATA(keyValue, ReturnValue)` where keyValue must be an integer or string and the ReturnValue can be any. The datastore file can have multiple such lines. A comment is started by two hyphens(--). The first line in the file must be the header whose format is `#path#date#version#0`.

For example, assume the datastore loaded has one entry ADD DATA(“Overheated”, 80). Also assume a variable X exists whose value is “Overheated”. If X is used as a parameter to Lookup, the value returned will be 80.

Using two variables Y and Z whose values are “Over”, “heated” would result in a key value of “Overheated” and the same value 80 would be returned.

To have changes in the `Composer.ds` file take effect, use the ECS configuration GUI to disable and re-enable the Composer correlation.

TIP	Typically the datastore is used to hold static topological information. (One could think of using scripts to run once a day, say midnight, to create the datastore file and update the ECS engine with the new file.)

Variable Example Update

Variable Example Update

- Problem Restatement
  - Maintaining the IP address list in Composer is confusing for Operators.
  - Move the IP address list to an external data store (file).
  - Composer needs to retrieve the list of equipment from the data store.

- Solution Design
  - Use Lookup variable type to retrieve the equipment list from the datastore

Variables

Name		
key1	Constant	"UnderMaint"
maintenance	Lookup	key1

Data store lookup key

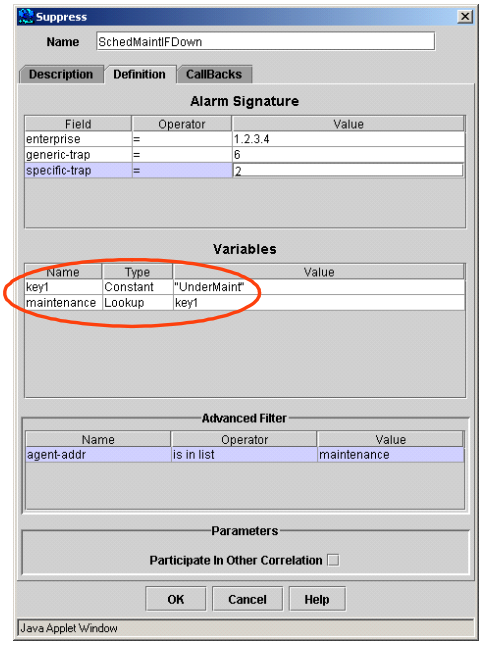
©2004 Hewlett-Packard Development Company, L.P.

Having seen this solution in operation, the customer has decided to adjust the requirements. Rather than maintaining the list of equipment within the Composer GUI, they would like to have it kept in a separate file which is easily maintainable by an Operator. Composer will have to look up values in this file to see if equipment emitting a trap is in the list.

Lookup `key1` and put the resulting list of strings in `maintenance`.

Variable Example Update - Definition Window

Definition Window



- Change Advanced Filter to match new variable name (select from list).

- Composer.ds:

ADD DATA (UnderMaint,[“15.75.76.127”,”15.75.76.128”])

©2004 Hewlett-Packard Development Company, L.P.

Follow the procedure given below to configure this correlator

1. Double-click the correlator from Variable Example to modify it.
2. Click the Definition tab to display the Alarm Definition panel.
3. Delete the row from the previous Advanced Filter definition.
4. Delete the variable `underMaint`.
   a. Right-click the variable.
   b. Select Delete.
5. Create a Constant variable `key1`.
   a. In the Name cell, enter the name of the variable `key1`.
   b. Click in the Value cell and enter the value to be used to index into the datastore.
6. Create the Lookup variable `maintenance` to hold the list of devices.
   a. In the Name cell, enter the name of the variable `maintenance`.
   b. Click in the Value cell to open the Lookup Definition dialog.
   c. Select `key1` from the parameter list.

7. In the Advanced Filter section, add a new line with the following value:
   - `Value = maintenance`
8. Click [OK] to complete the Correlator.

Extract Patterns

- Used in NNM and OVO
- Parse an input string to extract tagged variables
- Similar to regular expressions
- These tags appear as sub-variables to the assigned variable, and can be used like any other variables.

©2004 Hewlett-Packard Development Company, L.P.

Event attribute values can be extracted and used inside a Composer correlator or syslog trap definition.

Pattern Matching

Pattern Matching

- Special characters
  - ^ anchors to beginning of line
  - $ anchors to end of line
  - | Or operator allows a string to match one of two possibilities
  - \ mask or disable special meaning of special character and treat as itself
  - < > enclose a pattern to match
- Special sequences are
  - <#> for a number
  - <@> for a word
  - <S> or <_> for whitespace (note: Composer only uses <S>)
  - <*> to match anything

©2004 Hewlett-Packard Development Company, L.P.

Pattern-Matching

ECS provides a powerful text pattern-matching language that allows logical testing for the existence of substrings and patterns. Parts of a text string can be extracted and assigned to tags, which may be reused within the same scope. This section describes the operators and syntax of the pattern-matching language.

The pattern-matching language used in the match functions is the same as that used in HP OpenView Operations.

Frequently, pattern-matching means simply scanning for a specific substring in the target string. For example, to search for the substring ERROR anywhere in the target string you search for the pattern:

```
"ERROR"
```

Similarly, should you wish to match text not containing a specific substring (for example, WARNING), you type:

```
"<![WARNING]>"
```

This uses the not operator “!”, together with the chevrons “< >” that must enclose all operators,

and the square brackets “[]” that isolate sub-patterns.

You control case-sensitivity with a separate argument to the Match.make function.

Defining Match Expressions

- Ordinary Characters

  Ordinary characters generally represent themselves. However, if any of the following special characters are used they must be prefaced with a backslash escape character (\ ) to mask their usual function.

  [ ] < > | ^ $

- Expression Anchoring Characters (^ and $)

  If the caret (^) is used as the first character of the pattern, only expressions discovered at the beginning of lines are matched. For example, “^ab” matches the string “ab” in the line “abcde”, but not in the line “xabcde”.

  If the dollar sign is used as the last character of a pattern, only expressions at the end of lines are matched. For example, “de$” matches “de” in the line “abcde”, but not in the string “abcdex”.

  If ^ and $ are not used as anchoring characters, that is, not as first or last characters, they are considered as ordinary characters without masking.

- Expressions Matching Multiple Characters

  Patterns used to match strings consisting of an arbitrary number of characters require one or more of the following expressions:

  • <*> matches any string of zero or more characters (including separators)

  — <n*> matches a string of n arbitrary characters (including separators)

  • <#> matches a sequence of one or more digits

  • <n#> matches a number composed of n digits

  • <S> or <_> matches a sequence of one or more separator or whitespace characters. <_> is the preferred syntax.

  • <nS> matches a string of n separators

  • <@> matches any string that contains no separator characters, in other words, a sequence of one or more non-separators; this can be used for matching words.

  Separator characters are configurable for each pattern. By default, separators are the space and the tab characters. The separator string is specified as the second element in the 3-tuple passed to the Match.make function.

- Bracket ([and]) Expressions

  The brackets ([and]) are used as delimiters to group expressions. To increase performance, brackets should be avoided wherever they are superfluous. In the pattern:

  “ab[cd[ef]gh]”

  all brackets are unnecessary—"abcdefgh" is equivalent.

  Bracketed expressions are used frequently with the OR operator “|”, the NOT operator “!” and when using sub-patterns to assign strings to tags.

- The OR (|) Operator

  Two expressions separated by the vertical bar character "|" matches a string that is matched by either expression. For example, the pattern:

  "[ab|c]d"

  matches the string "abc" and the string "cd".

- The NOT (!) Operator

  The not operator "!" must be used with delimiting square brackets, for example:

  "<![WARNING]>"

  The pattern above matches all text which does not contain the string "WARNING".

  The not operator may also be used with complex sub-patterns:

  "LN<*>: R< ![490|[501[a|b]]] >-<*>"

  The above pattern makes it possible to generate a message for any line connection other than from repeaters 490, 501a or 501b.

  Therefore, the following would be matched:

  "LN270: R300-427"

  However, this string is not matched, because it refers to repeater 501a:

  "LN270: R501a-800"

  If the sub-pattern including the not operator does not find a match, the not operator behaves like a <*>: it matches zero or more arbitrary characters. For this reason, there is a difference between the UNIX expression "[!123]", and the corresponding ECS pattern matching expression: "<![1|2|3]>". The ECS expression matches any character or any number of characters, except 1, 2, or 3; the UNIX expression matches any one character, except 1, 2, or 3.

- The Mask (\ ) Operator

  The backslash "\" is used to mask the special meaning of the characters:

  [ ] < > | ^ $

  A special character preceded by \ results in an expression that matches the special character itself.

  Because ^ and $ only have special meaning when placed at the beginning and end of a pattern respectively, you need not mask them when they are used within the pattern (in other words, not at beginning or end). The only exception to this rule is the tab character, which is specified by entering "\t" into the pattern string.

Extract Variable Assignment

Extract Variable Assignment

- Use tags to define sub-variables.
  - Match attempts to move from left to right to achieve success if possible.
  - <@>, <#>, <S> match as many characters as possible.
  - <*> matches as few characters as possible.
  - <*> at the start/end of the pattern takes the start/end of the line.

- Example: To extract the card and port numbers from a message such as
  Card = 10 : Port = 1
  Card<S>=<S><*.Card><S>:<S>Port<S>=<S><*.Port>

- Assigns 10 to Card
  1 to Port

U5089S ©2004 Hewlett-Packard Development Company, L.P. 13

Tags

Search patterns may use tags to identify part(s) of the target string to, for example, compose a new string from selected parts of the target string. To define a tag, add ".tagname" before the closing chevron. The pattern:

```
^errno: <#.number> - <*.error_text>
```

matches a string such as:

```
errno: 125 - device not in service
```

and assigns "125" to the tag number and "device not in service" to the tag `error_text`. The tags may be accessed as members of a dictionary. See the *HP OpenView Correlation Composer's Guide*.

Assignment Rules

In matching the pattern "<*.tag1><*.tag2>" against the string "abcdef", it is not immediately clear which substring of the input string is assigned to each tag. For example, it is possible to

assign an empty string to tag1 and the whole input string to tag2, as well as assigning “a” to tag1 and “bcdef” to tag2, and so forth.

The pattern-matching algorithm always scans both the input line and the pattern definition (including alternative expressions) from left to right. <*> expressions are assigned as few characters as possible. <#>, <@>, <S> expressions are assigned as many characters as possible.

Therefore, tag1 will be assigned an empty string in the above example. To match an input string such as:

```
"this is error 100: big problem"
```

use a pattern such as:

```
error <#.errnumber>:<*.errtext>
```

In which:

- “100” is assigned to the tag `errnumber`.
- “big problem” is assigned to the tag `errtext`.

For performance and pattern readability purposes, you can specify a delimiting substring between two expressions. In the above example, “:” is used to delimit <#> and <*>.

Matching <@.word><#.num> against “abc123” assigns “abc12” to `word` and “3” to `num`, as digits are permitted for both <#> and <@>, and the left expression takes as many characters as possible.

Patterns without expression anchoring can match any substring within the input line. Therefore, patterns such as:

```
"this is number<#.num>"
```

are treated in the same way as:

```
"<*>this is number<#.num><*>"
```

Sub-Patterns Assignment

In addition to being able to use a single operator, such as * or #, to assign a string to a tag, you can also build up a complex sub-pattern composed of a number of operators, according to the following pattern:

```
<[sub-pattern].tag>
```

For instance: <[rack<#>.brd<#>].hware>

In the example above, the period (.) between rack<#> and brd<#> matches a similar dot character, while the dot between] and hware is necessary syntax. This pattern would match a string such as “rack123.brd47” and assigns the complete string to hware.

Other examples of sub-patterns are:

```
<[Error|Warning].sev>
```

and

```
<[Error[<#.n><*.msg>]].complete>
```

In the first example above, any line with either the word “Error” or the word “Warning” is assigned to the tag, `sev`. In the second example, any line containing the word “Error” has the error number assigned to the tag, n, and any further text assigned to msg. Finally, both number and text are assigned to complete.

Extract Example

Extract Example

- A print spooler sends a message containing the following string:

  JobID=345;Target=ljet1;Prio=7;Model=Laserjet 5 MX; Status=TonerLow;Error=37
- Extract the status of the printer to see whether it is “Normal.”

- Simplest:

  <*>Status=<*.status>;<*>

- Most Versatile:

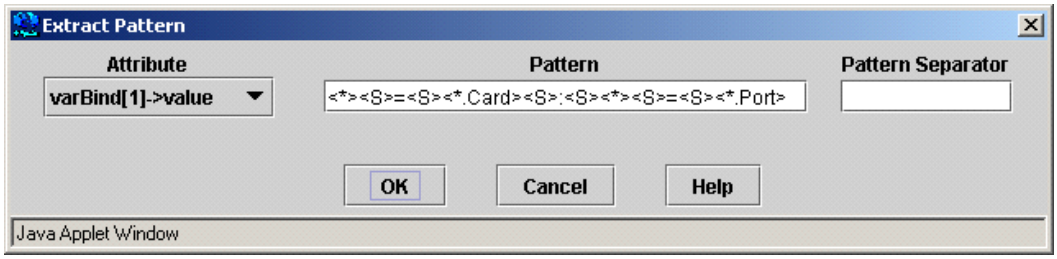
  JobID=<*.jobid>;Target=<*.target>;Prio=<*.priority>;<*>; Status=<*.status>;Error=<*.errornum>

©2004 Hewlett-Packard Development Company, L.P.

Extract Variable Definition

Extract Variable Definition

1. Select Extract from the drop down menu. Click in the value cell. The Extract Pattern dialog is displayed.
2. Select the attribute from which you want to extract a substring.
3. Enter the pattern in the pattern text box.
4. Click [OK] to close the Extract Pattern window.



U5089S ©2004 Hewlett-Packard Development Company, L.P. 15

Event attribute values can be extracted and used inside a correlation definition. Follow the procedure described below to define an extract pattern.

1. Select Extract from the drop down menu. Click in the value cell. The Extract Pattern dialog is displayed.
2. Select the attribute from which you want to extract a substring.
3. Enter the pattern in the pattern text box. Refer to Appendix D, “Pattern Matching,” in the *HP OpenView Correlation Composer’s Guide* for extract pattern examples.
4. Enter the pattern separator in the Pattern separator text field. The Pattern Separator is by default an empty space.
5. Click [OK] to close the Extract Pattern window.

Extract Example

Extract Example

- Problem Statement
  - A network spooler emits an event whenever a new job is submitted.
  - Suppress the event if the size of the current job can be handled adequately by the amount of free space.
- Solution Design
  - Extract the job size and the spooler free space from the varbinds.
  - Compare them in the Advanced Filter to decide which to suppress.

©2004 Hewlett-Packard Development Company, L.P.

What do you need to know?

```
Trap-PDU{
enterprise {1 2 3 4},
agent-addr internet : "\x7f\x00\x00\x02",
generic-trap 6,
specific-trap 10,
variable-bindings{
{
name {1 2 3 4 1},
value simple : string : Location=Building 41, Roswell; Contact=John Bigboote, 729-315-4545
},
{
name {1 3 6 1 4 11 2 17 2 1 0},
value simple : string : Model=5317;MaxSpace=6420000K;MaxQueues=12;MaxJobs=512
},
{
name {1 3 6 1 4 1 11 2 17 2 1 0}
value simple : string : CurJobs=5;CurFreeSpace=23980;NonEmptyQueues=2;CurMaxDepth=3
},
{
name {1 3 6 1 4 1 11 2 17 2 1 0}
value simple : string :
JobID=345;Target=ljet1;Size=45135;Submit=20030519.114525;Prio=7;Type=Binary
},
{
```

```
name {1 3 6 1 4 1 11 2 17 2 1 0}
value simple : string : Type=printer;Model=Laserjet 5 MX;Avail=N;Status=TonerLow;Error=37
}
}
}
```

1. How do you identify the spooler trap?

   All traps with the following attributes are identified as spooler traps.

   - enterprise is set to 1.2.3.4
   - generic-trap is set to 6
   - specific-trap is set to 10

2. What information is included with the trap?

   var-bind[0] of the event provides the location and contact of the spooler.

   var-bind[1] provides information about the spooler.

   var-bind[2] provides information about the spooler status.

   var-bind[3] provides information about the job.

   var-bind[4] provides information about the status of the output device.

3. How do you know the size of the current job?

   Extract it from varbind[3].value into a variable named `JobInfo`. You can reference the extracted variable by `JobInfo.Size`.

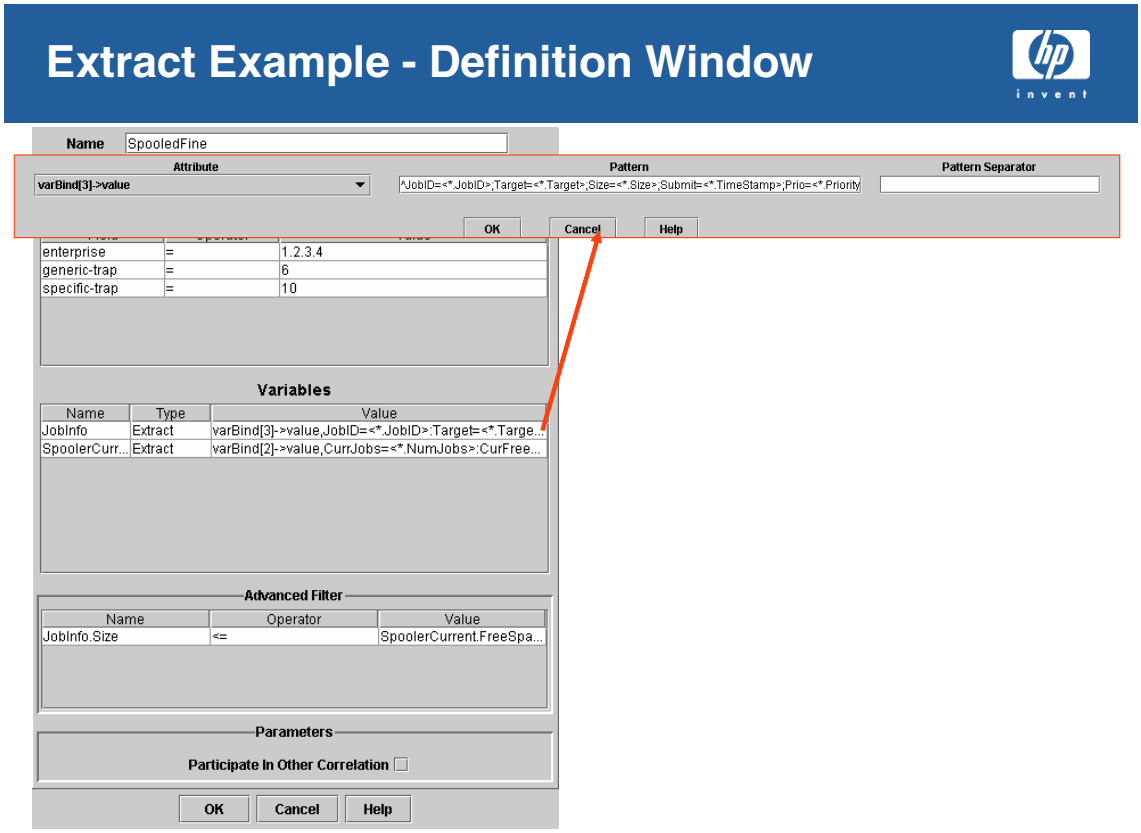
4. How do you know the amount of space free on the spooler?

   Extract it from varbind[2].value into a variable named `SpoolerCurrent`. You can reference the extracted variable by `SpoolerCurrent.FreeSpace`.

5. Which traps should be suppressed?

   Traps should be suppressed if `JobInfo.Size <= SpoolerCurrent.FreeSpace`. Place that qualification into the Advanced Filter.

Extract Example - Definition Window



Follow the procedure given below to configure this correlator:

1. Select `Correlations:Correlator Templates->Suppress` from the Correlator Store window. You can also click the Suppress icon in the Correlator Templates toolbar.
2. Enter the name of the Correlator in the Name text field.
3. Enter the description of the Correlator. The Description can briefly state what the Correlator is expected to do.
4. Click the Definition tab to display the Alarm Definition panel.

   Set the following values to define the Alarm Signature:

   - `enterprise = 1.2.3.4.`
   - `generic-trap = 6`
   - `specific-trap = 10`
5. Create an Extract variable `JobInfo`.

   a. In the Name cell, enter the name of the variable `JobInfo`.

   b. Click in the Value cell to open the Extract Pattern dialog.

   c. Select `varbind[3]->value` as the Attribute.

d. Type the Pattern (carefully!):
`JobID=<*.JobID>;Target=<*.Target>;Size=<*.Size>;Submit=<*.TimeStamp>;Prio=<*.Priority><*>`

e. Leave the Pattern Separator blank.

f. Click [OK].

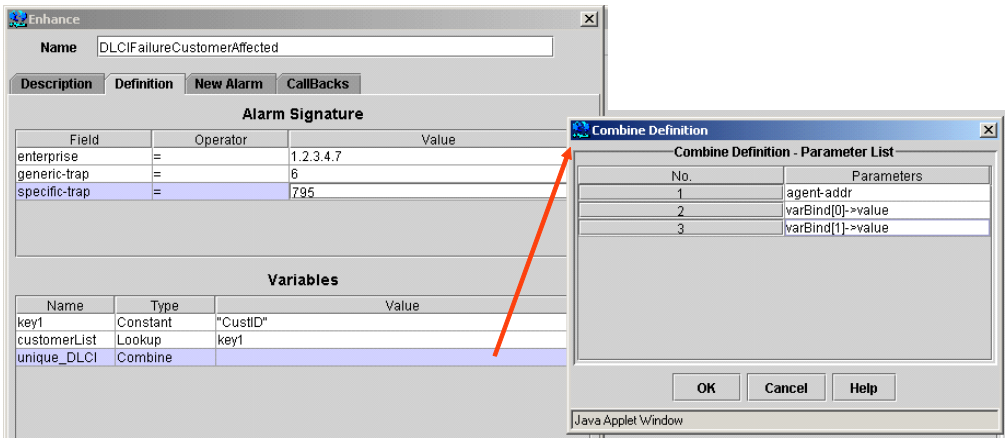
6. Create an Extract variable `SpoolerCurrent`. Extract the free space into a variable using the pattern:
`CurJobs=<*.CurJobs>;CurFreeSpace=<*.FreeSpace>;NonEmptyQueues=<*NonEmptyQueues><*>`

7. In the Alarm Filter, compare `JobInfo.Size <= SpoolerCurrent.FreeSpace`.

8. Click [OK].

Creating a Combine Variable

Creating a Combine Variable



- Add the variable row, give it a name.
- Select Combine from the list.
- Click in value cell to open Combine Definition dialog.
- Right-click to add a row.
- Select parameter from list.

U5089S ©2004 Hewlett-Packard Development Company, L.P. 18

A new variable can be defined, when you combine two or more previously defined variables, attributes or Global Constants. Follow the procedure given below to combine values of two or more variables.

1. Click in the value cell. The Combine Definition window is displayed.
2. The Parameter list has to be filled to combine the variables. Click the Parameters cell. A pop up menu displaying all attributes, pre-defined variables and Global Constants is displayed.
3. Select the attribute or variables from the pop up menu. The selected item is displayed in the parameters cell.
4. To add a variable or attribute to the list, right click the mouse button and select [Add]. A new row is added.
5. Repeat to combine more variables/attributes.

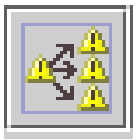
Enhance Correlator Template

Enhance Correlator Template

•The Enhance Correlator Template is used to:

- Trigger the creation of one or more new events
- Augment the information content of an event by modifying it's event attributes

•Example - Modify the severity of an event depending on the customer who is affected



©2004 Hewlett-Packard Development Company, L.P.

The Enhance Correlation template is used to trigger the creation of one or more new events or to augment the information content of an event.

Enhance Correlation can:

- trigger the creation of one or more new events. For example, you can create a new event that enumerates the set of customers affected by a failed entity
- augment the information content of an event by modifying event attributes of an event. For example, modify the severity of an event depending on the customer who is affected.

By default, the event is enhanced only if no other rule has chosen to discard the event. However, this default behavior may be overridden. Select the [`Enhance Always`] button to enhance the event at all times. The event is then enhanced even if any other correlator discards the event.

NOTE When an event is altered, a copy is made of the original event and then the copy is modified. In NNM, the UUID of the event changes when altered.

Enhance Example

Enhance Example

•Problem Statement

- In a FrameRelay (FR) environment when a FR circuit fails (DLCI fails), a trap is emitted by the equipment.
- The mapping of DLCI to customer ID is available in a DataStore.
- Create a new event where
  - specific-trap is set to 5 (the other attributes are the same as the original trap)
  - a new varbind contains the name of the customer affected by the DLCI failure

•Solution Design

- Identify the customer affected by the DLCI failure
  - A DLCI is uniquely identified by a combination of the agent-address, interface-number (varbind[0].value), DLCI ID (varbind[1].value)
  - For each unique DLCI, Lookup a customer ID

 ©2004 Hewlett-Packard Development Company, L.P.

Before you begin to configure this correlator, what do you need to know?

1. How do you recognize a DLCI failure trap?

   By the specific trap ID (this example uses 795). All DLCI_failure traps will enter the correlator.

2. How do you retrieve the name of the customer who has been affected?

   The datastore file contains the mapping of the Customer ID with a unique DLCI. Given a DLCI, a Lookup variable can retrieve the Customer ID.

3. How do you get the unique DLCI from the trap that has just arrived?

   - Combine the agent-address, interface-number (variable-binding[0].value) and DLCI ID (variable-binding[1].value) into a variable, `unique_dlci`.

4. What happens once the trap has been identified?

   Create a new event.

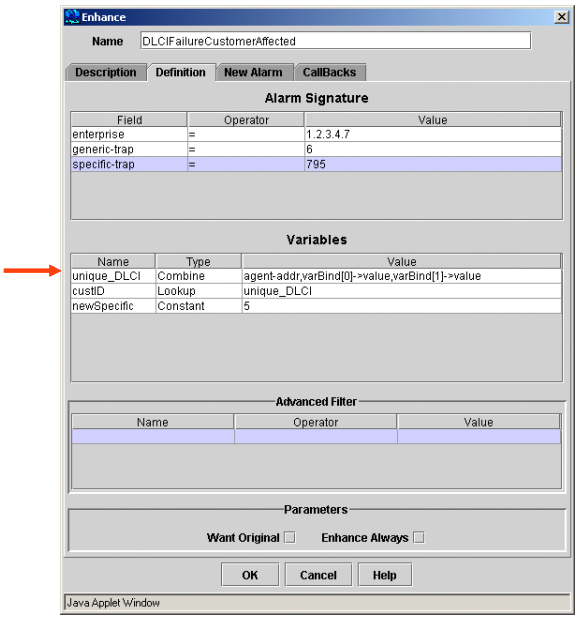
5. What are the specifications of the new event?

   - `specific-trap= newSpecific`. Create a variable `newSpecific=5` in the Variable section.
   - `variable-bindings[2].name = variable-bindings[1].name`.

- `variable-bindings[2].value= custid`.
- map all other attributes to the original trap's attributes.

Enhance Example - Definition Window

Enhance Example - Definition Window



- Create the unique_DLCI string by combining the agent-addr, varbind[0], and varbind[1].
- Lookup the custID using unique_DLCI as the index to the datastore.

©2004 Hewlett-Packard Development Company, L.P.

Follow the procedure given below to configure this correlator

1. Select `Correlations:Correlator Templates->Enhance` from the Correlator Store window or click the Enhance toolbar icon.
2. Enter the name of the Correlator in the Name text field.
3. Enter the description of the Correlator. The Description can briefly state what the Correlator is expected to do.
4. Click the Definition tab to display the Alarm Definition panel.

   Set the following values to define the Alarm Signature:

   - `enterprise = 1.2.3.4.7`
   - `generic-trap = 6`
   - `specific-trap = 795`
5. Create a Combine variable `unique_DLCI`.

   a. In the Name cell, enter the name of the variable `unique_DLCI`.

   b. Click in the Value cell to open the Combine Definition dialog.

   c. Select `agent-addr` as the first parameter.

d. Right-click and Add an additional row.

e. Select the variable `varbind[0]->value`.

f. In another row, select the variable `varbind[1]->value`.

g. Click [`OK`].

6. Create the Lookup variable `CustID` to hold the customer identifier.

   a. In the Name cell, enter the name of the variable `CustID`.

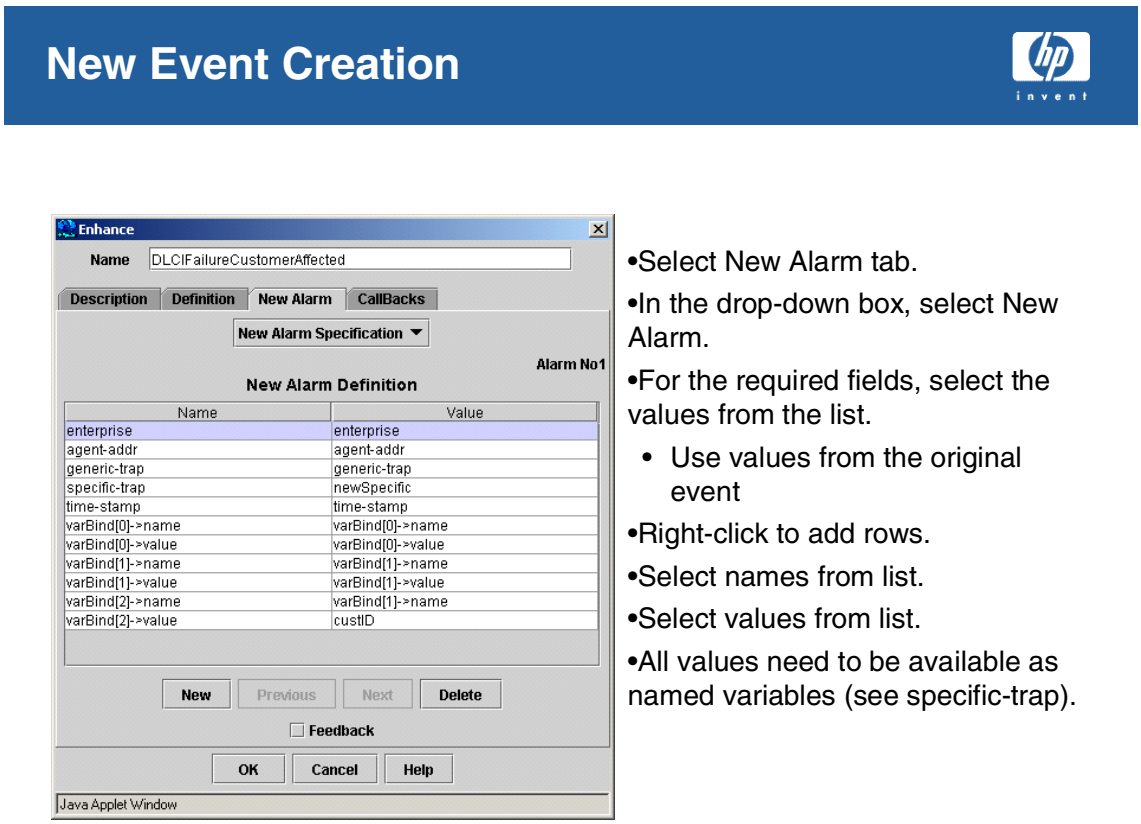
   b. Click in the Value cell to open the Lookup Definition dialog.

   c. Select `unique_DLCI` from the parameter list as the key.

7. Create a Constant variable `newSpecific` to hold the event ID to be created.

   a. Give the variable the value of `5`.

New Event Creation



The Correlator configuration contains the New Alarm Creation tab. Correlators can include definitions to trigger new events with information content that is useful for the operator. For example, in the case of Repeated Correlation, a new event can be output at the end of the window period, reporting the number of events that arrived in this time period.

New events can be created in two ways:

Alter Specification

New events can be created by altering some of the attributes of the existing event. The Alarm Specification contains Field, Mode and Value.

- Field - Name of the field to be altered. The Field is a drop down menu and displays all attributes for the selected Event type.
- Mode - There are two modes that can be used to alter the event's attributes.
  - — Replace: The new value replaces the event attribute of the original event.
  - — Append: The new value specified is appended to the existing event attribute.
- Value - The value of event's attributes is appended or replaced with new values.

NOTE	The attributes displayed to alter an event are always the attributes of the last event that arrived. For example, while using Transient correlation, the attributes displayed are always those of the last Clear Alarm.

Follow the steps given below to alter the attributes of any event.

1. Click the New Alarms tab in the Correlator window. The New Alarms panel opens. Note that the None option is chosen by default.
2. Select Alter Specification from the drop down menu. The Alter Alarm Definition table is displayed.
3. Select the attribute that you would like to alter from the drop down menu in the Field column.
4. Select the mode of alteration from the drop down menu in the Mode Column. The two options are
5. Select the variable that is to replace or added to the attribute contents from the pop up menu.

Note that any new values to be appended must have already been defined in the Variables section of the Correlator.

New Alarm Specification

New events can be created with all new attributes. The New Alarm Specification includes Name and Value. The New Alarm Specification displays all the mandatory attributes that have to be entered to create a new event.

Follow the procedure given below to create a new event.

1. Select New Alarm Specification from the drop down menu in the New Alarms panel. The New Alarm Definition table is displayed.
2. The New Alarm Definition table displays all the mandatory attributes that must be filled in. Select values for the fields from the pop-up list displayed.

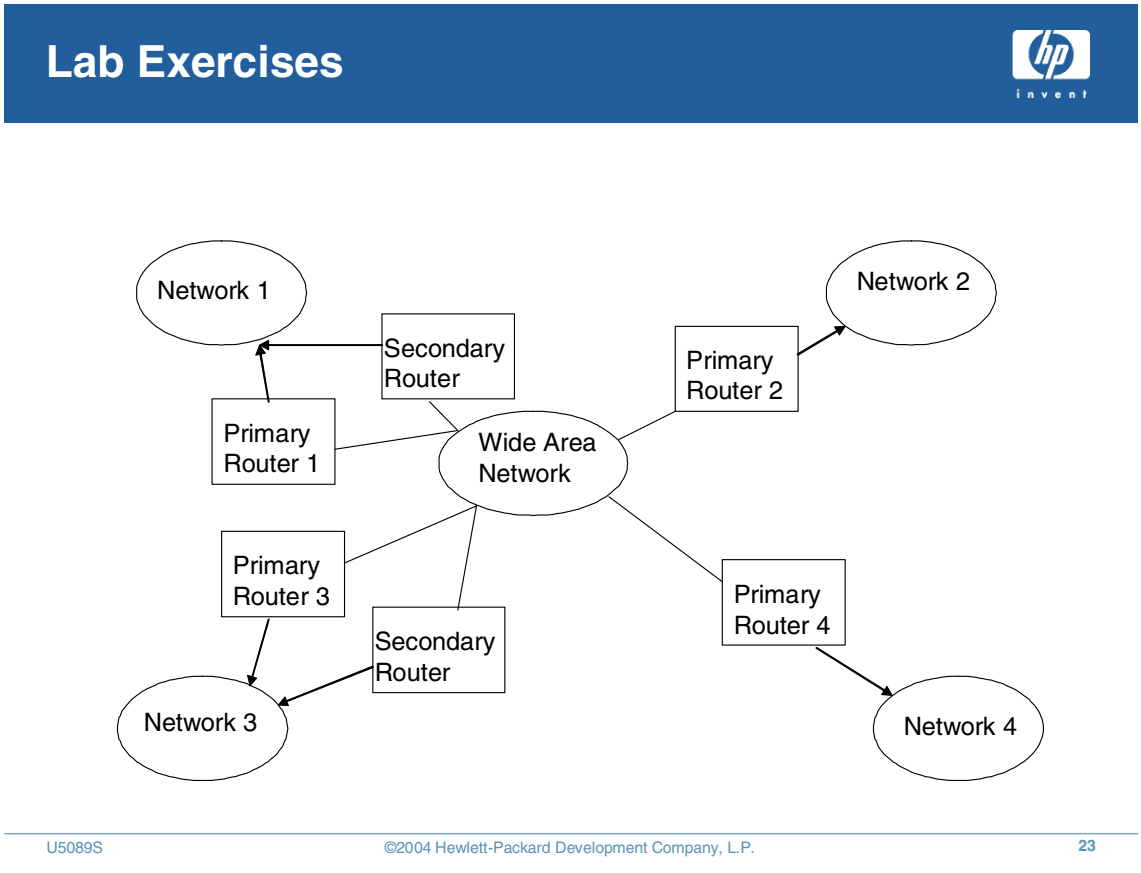
   The list of attributes are picked up from a configuration file. Refer to the *HP OpenView Correlation Composer's Guide* for details on how to edit the configuration file.
3. To add new attributes to the new event created, right click the mouse button and select `Add`. A new row is added to the New Alarm Definition table.
4. Notice that the text "Alarm Number:1" is displayed on the right hand top corner of the table. The Alarm number indicates the number of the new event getting created. You can navigate through the list of events defined.

Additionally you can feedback the event into the Composer correlation. Select the Feedback button if the new event has to be fedback to participate in other Correlators.

NOTE	The New Alarm tab is not available for Suppress Correlation as no new events can be output.

Variable bindings are a name value pair where the name is always an Object ID. While specifying a new event that has variable bindings, both the name and the value for each variable binding need to be specified. Variable bindings start with an index of 0.

Lab Exercises



LAB: Suppress Correlator

This Lab Session requires:

- Suppress Correlator
- Alarm Signature
- Advanced Filter
- Operators
- Constant Variable
- Lookup Variable

Correlation Requirement

Discard all interface-down events from a list of interfaces on the first day of January in all the

years. These interfaces will go down for maintenance during this day.

The list of interfaces will be stored in a datastore in the following format with the key "IF_Failure":

`Composer.ds` DataStore:

```
ADD DATA ("IF_Failure", ["First", "Second", "Third", "Fifth"])
```

Example Trap-PDU

```
Trap-PDU {
    enterprise {1 6 7 8},
    agent-addr internet : "\x7f\x00\x00\x02",
    generic-trap 6,
    time-stamp 0,
    variable-bindings {
{
    name {1 3 6 1 4},
    value simple : string : "Second"
},
{
    name {1 3 6 1 4 11},
    value simple : string : "01:01:03:01:00:01"
},
{
    name {1 2 6 5 8},
    value simple : number : 20
}
    }
}
```

Event Definition

An interface-down trap is identified by

- enterprise id set to 1.6.7.8
- generic-trap set to 6
- varBind [0] contains interface number
- varBind [1] contains the time-stamp in the format dd:mm:yy:hh:mi:ss

View Pre-Correlator Results

1. A sample event stream from this environment is available for testing. Copy the following file from `$OV_CONTRIB/OVTraining/NNM3` to `$OV_CONTRIB/ecs`:
   - `maintIF.evt`: sends the following events:
     — An event that matches the signature for this lab.

— An event with a different specific ID.

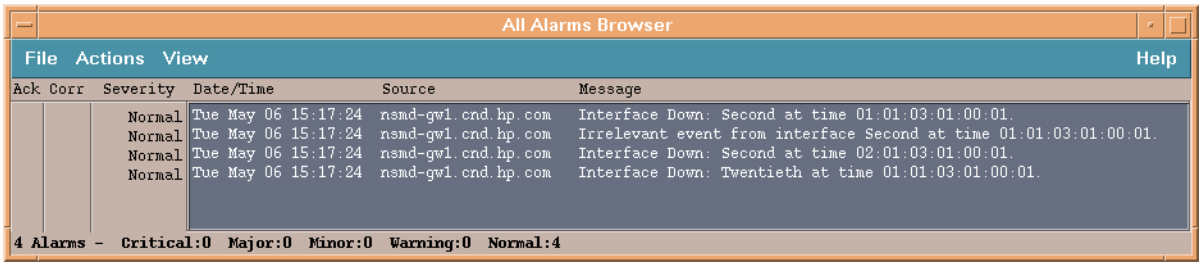
— An event from a different date.

— An event from a different interface.

2. Open the Composer Labs Alarms Browser.
3. Empty all old alarms by selecting `Actions:Delete->All Alarms in Category`.
4. Type the commands:

```
cd $OV_CONTRIB/ecs

ecsevgen -n maintIF.evt
```

5. Review the alarms in the Alarm Browser.



6. To mark which alarms came from this pre-test run, select `Actions:Acknowledge->All Alarms in Category`.

Directions

Execute the following steps to solve the Lab Session:

7. Start the Composer development interface.
8. Create a new correlator store with SNMP events.
9. Create a Suppress correlator.
10. In the Alarm Signature section, enter the following to filter the interface down events:

Field	Operator	Value
enterprise	=	1.6.7.8
generic-trap	=	6

11. The list of interfaces are available in the datastore with a key value “IF_Failure”. Use Lookup variable and pass the key value to look for in the datastore to get the list of interface failures.

    a. Create variables to pass the key and to get the interface failures from datastore.

Name	**Type**	**Value**
IFKey	Constant	“IF_Failure”
IFLookup	Lookup	IFKey

    b. Create a variable to filter the events with time stamp matches the 1st of January as shown below.

Variables

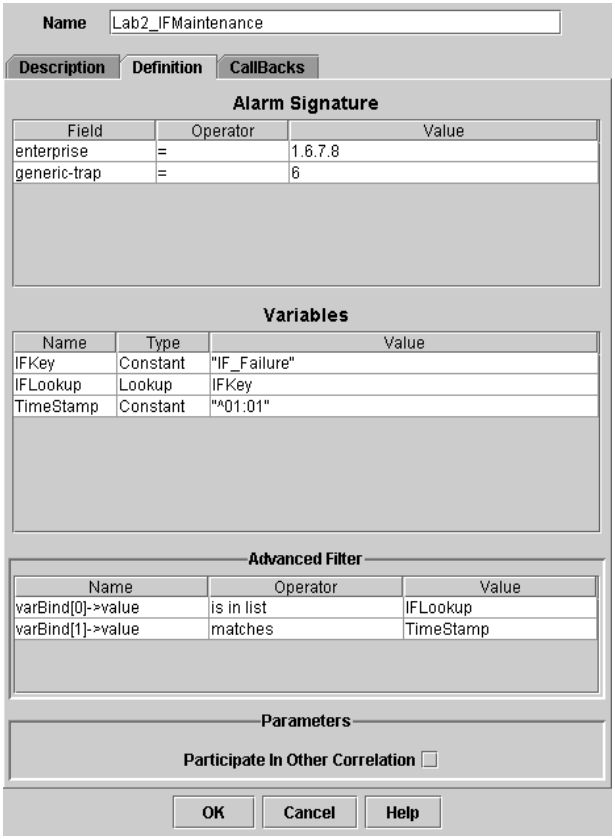
Name	Type	Value
TimeStamp	Constant	"^01:01"

12. In the Advanced Filter, evaluate the incoming event's varBind[0] value with datastore lookup value to check whether the interface is under maintenance. Also evaluate the incoming event's varBind[1] value matches the TimeStamp. Enter the values in the Advanced Filter as shown below.

    Advanced Filter

Name	Operator	Value
VarBind[0]->value	is in list	IFLookup
VarBind[1]->value	matches	TimeStamp

    If the incoming event passes the Alarm Signature and Advanced Filter, then it will be suppressed. In the Advanced Filter, the operator "matches" checks the incoming event's varBind [1] value for the exact pattern of first five characters specified in the TimeStamp value. For this the "^" is used. The first 5 characters of varBind[1] contains "01:01" then, only those events will be passed through this Advance filter and all those events are discarded(suppressed).



13. Save your Correlator Store in the `$OV_CONF/ecs/CIB` directory.

14. Close the file in the developer interface. You can leave the interface running.

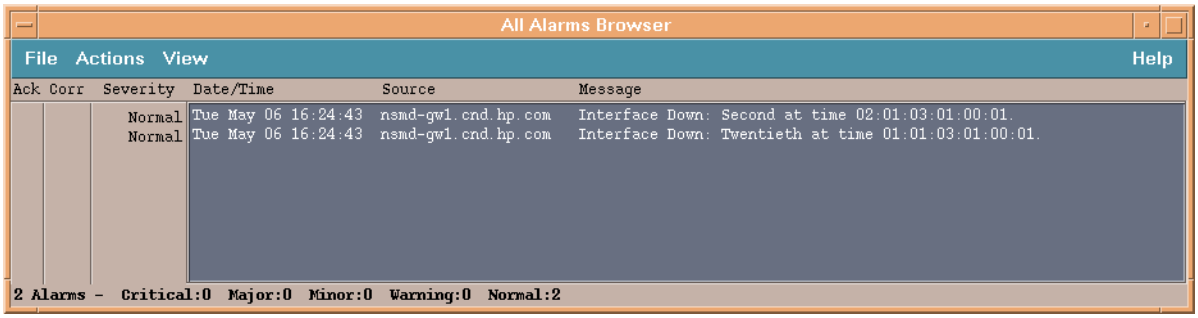
15. Add your data to the `Composer.ds` datastore.
16. Provide Operator access to your correlator.
17. Deploy your correlation from the command line.

Test Your Correlator

18. Type the commands:

```
cd $OV_CONTRIB/ecs

ecsevgen -n maintIF.evt
```

19. Examine the Composer Labs Alarms Browser. You should see a no indication of the first event. There should be no indication of the second event since the enterprise ID and generic ID still match the signature. The third event should not be correlated since it is not from January, so you see it in the Alarm Browser. The fouth event also shows in the Alarm Browser since the interface is not in your data store.



LAB: Enhance Correlator Template (Optional)

This Lab Session requires:

- Enhance Correlator
- Combine Variable
- Lookup Variable
- New Alarm Section
- Global Constants

Correlation Requirement

Assume a wide area network with many small networks interconnected by routers (primary and secondary routers to share the load). Some of the sub-networks are connected only by one router (primary) as shown.

If a Router down event is emitted, the correlator should check if the problematic router has a secondary router to share the load.

- If yes, the event should be enhanced with the information that “the problematic router is backed-up by secondary router and all the traffic will be routed through the secondary router”.
- If no, the event should be enhanced with warning information that “no backup router is available and the sub-network is isolated”.

The router is uniquely identified by a combination of enterprise ID + Router ID (varBind[0] value). The secondary router’s id is available in the datastore and can be retrieved by passing the primary router id.

The datastore looks like:

```
ADD DATA ("<enterpriseid + Rotuerid>", "<Secondary Router information>" )
```

Trap Definition

A router fail trap is identified by:

- enterprise id set to 1.5.6.7.8
- specific-trap set to 6
- var-bind0 contains router id

Example Trap-PDU

```
Trap-PDU {
    enterprise {1 5 6 7 8},
    agent-addr internet : "\x7f\x00\x00\x02",
    specific-trap 6,
    time-stamp 414746291,
    variable-bindings {
{
    name {1 3 6 1},
    value simple : number : 5
```

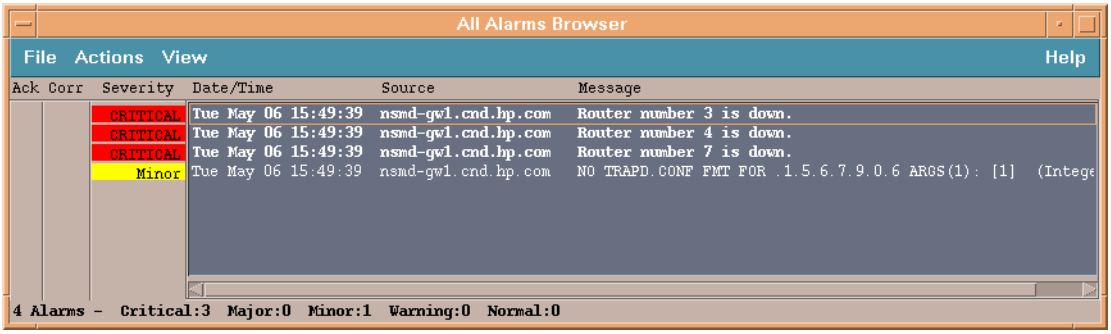
```
}
    }
}
```

View Pre-Correlator Results

1. A sample event stream from this environment is available for testing. Copy the following files from `$OV_CONTRIB/OVTraining/NNM3` to `$OV_CONTRIB/ecs`:
   - `routerBackup.evt`: sends the following events:
     - — From a router that has a secondary.
     - — From a router that does not have a secondary.
     - — The Router ID is not in the Lookup list.
     - — The enterprise does not match.
2. Open the All Alarms Browser. (Since one of these events is purposely not listed in trapd.conf, the configuration alarm for that only appears in the All Alarms Browser.)
3. Empty all old alarms by selecting `Actions:Delete->All Alarms in Category`.
4. Type the commands:

   ```
   cd $OV_CONTRIB/ecs
   ecsevgen -n routerBackup.evt
   ```

5. Review the alarms in the Alarm Browser. (Hint: The event with a different enterprise ID only shows in the All Alarm Browser.)



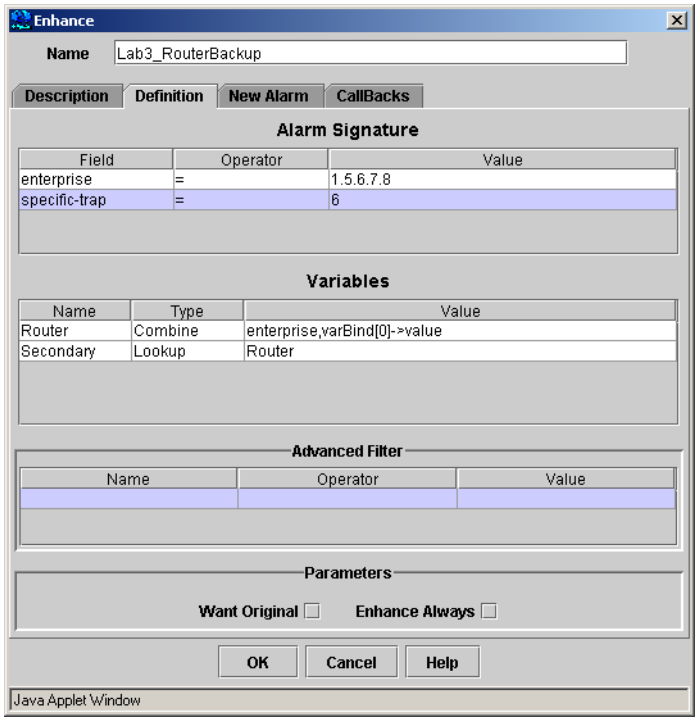
6. To mark which alarms came from this pre-test run, select `Actions:Acknowledge->All Alarms in Category`.

Directions

Execute the following steps to solve the Lab Session:

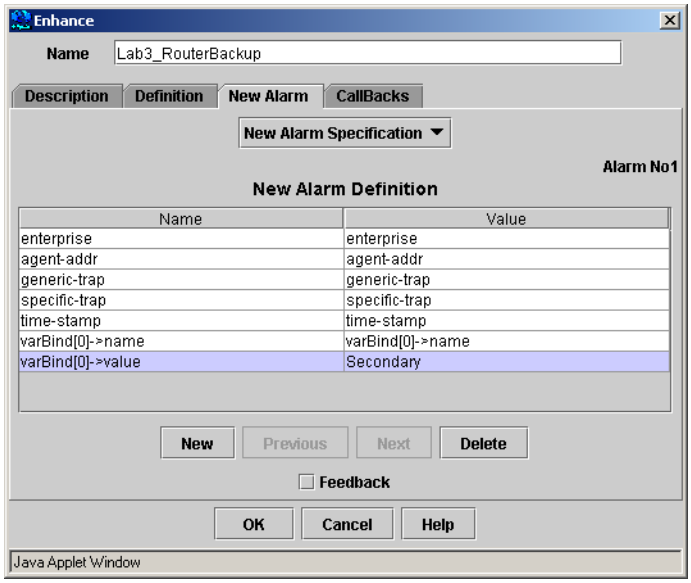
1. Start the Composer development interface.
2. Create a new correlator store with SNMP events.
3. Use Enhance Correaltor Template.
4. In Alarm Signature section, filter the router down events from a network.
5. The router is uniquely identified by the combination of enterprise id and router id (varBind[0] value). Create a Combine variable and add these two attributes.

6. The secondary router detail is available in the datastore. To get the secondary details, create a Lookup variable and pass the unique router created in the last step.



7. Use the New Alarm Section and enhance the event with the secondary router information. Add a varbind name and value to contain the Secondary variable information.

   If the incoming event passes the Alarm Signature and Advanced Filter, the variables which are used in the New Alarm section are evaluated and the enhanced event is output from the Correlator.



8. Save your Correlator Store in the `$OV_CONF/ecs/CIB` directory.
9. Exit the Composer developer interface.
10. Add your data to the datastore. Refer to the lab exercise slide for the connectivity diagram.

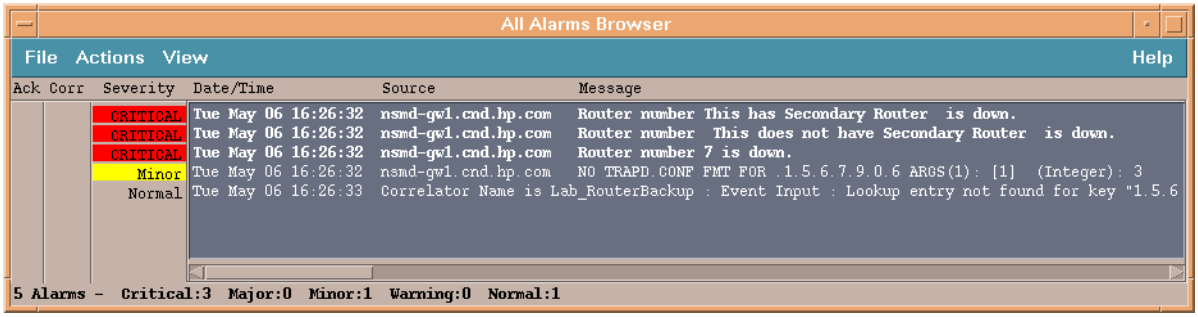
11. Provide Operator access to your correlator.
12. Deploy your correlation.

Test Your Correlator

13. Type the commands:

```
cd $OV_CONTRIB/ecs

ecsevgen -n routerBackup.evt
```

14. Examine the Composer Labs Alarms Browser. You should see a new event stating that there is a secondary, a new event stating that there is no secondary, an original event with a router ID not in the list, and an original event from a different enterprise. Composer also displays an alarm that the router ID not in the list failed the Lookup in the All Alarms Browser.



LAB: Extract Variable

This Lab Session requires:

- Enhance Correlator
- Alarm Signature
- Advanced Filter
- Operators
- Extract Variable

Correlation Requirement

A network spooler emits an event whenever a new job is submitted. This event can be ignored unless the event indicates that the specified output device is unavailable and the job has high priority (over 5). In this case, emit a new event indicating whom to call and why.

Example Trap-PDU

```
Trap-PDU{

enterprise {1 2 3 4},

agent-addr internet : "\x7f\x00\x00\x02",
generic-trap 6,

specific-trap 10,

variable-bindings{

{

name {1 2 3 4 1},

value simple : string : Location=Building 41, Roswell; Contact=John Bigboote, 729-315-4545

},

{

name {1 3 6 1 4 11 2 17 2 1 0},

value simple : string : Model=5317;MaxSpace=6420000K;MaxQueues=12;MaxJobs=512
},

{

name {1 3 6 1 4 1 11 2 17 2 1 0}

value simple : string : CurJobs=5;CurFreeSpace=23980;NonEmptyQueues=2;CurMaxDepth=3
},

{

name {1 3 6 1 4 1 11 2 17 2 1 0}

value simple : string :
JobID=345;Target=ljet1;Size=45135;Submit=20030519.114525;Prio=7;Type=Binary
},

{

name {1 3 6 1 4 1 11 2 17 2 1 0}

value simple : string : Type=printer;Model=Laserjet 5 MX;Avail=N;Status=TonerLow;Error=37
}

}
```

```
}
```

Event Definition

An interface-down trap is identified by

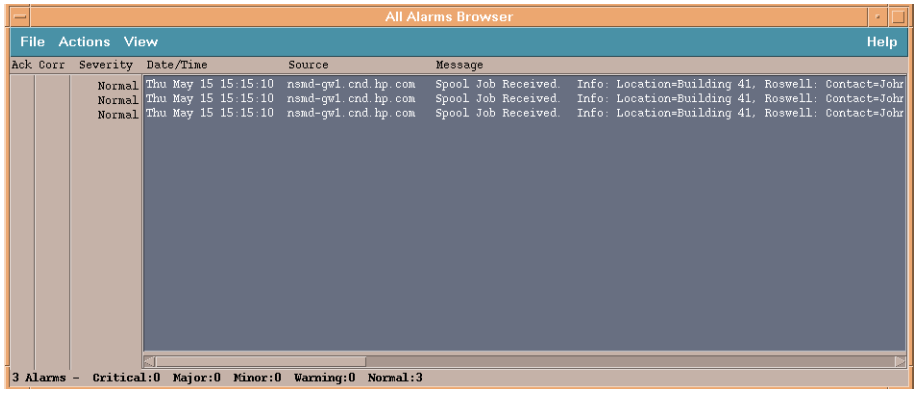
- enterprise is set to 1.2.3.4
- generic-trap is set to 6
- specific-trap is set to 10
- var-bind[0] of the event provides the location and contact of the spooler.
- var-bind[1] provides information about the spooler.
- var-bind[2] provides information about the spooler status.
- var-bind[3] provides information about the job.
- var-bind[4] provides information about the status of the output device.

View Pre-Correlator Results

1. A sample event stream from this environment is available for testing. Copy the following file from `$OV_CONTRIB/OVTraining/NNM3` to `$OV_CONTRIB/ecs`:
   - `spooler.evt`: sends the following events:
     — high priority job with printer down
     — high priority job with printer Normal
     — low priority job with printer down
2. Open the Composer Labs Alarms Browser.
3. Empty all old alarms by selecting `Actions:Delete->All Alarms in Category`.
4. Type the commands:

   ```
   cd $OV_CONTRIB/ecs
   ecsevgen -n spooler.evt
   ```

5. Review the alarms in the Alarm Browser.



6. To mark which alarms came from this pre-test run, select `Actions:Acknowledge->All Alarms in Category`.

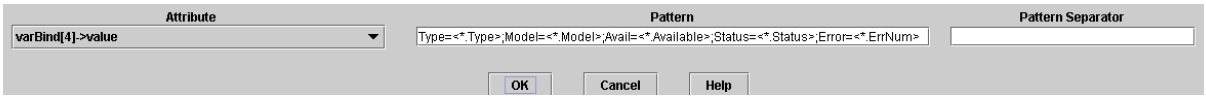
Directions

Execute the following steps to solve the Lab Session:

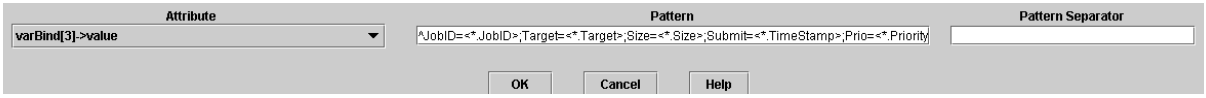
7. Start the Composer development interface.
8. Create a new correlator store with SNMP events.
9. Create an Enhance correlator.
10. In the Alarm Signature section, enter the following to filter the interface down events:

Field	Operator	Value
enterprise	=	1.2.3.4
generic-trap	=	6
specific-trap	=	10

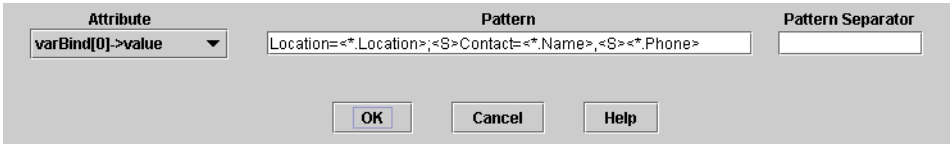
11. Create a variable named `OutputDevice`. Extract the output device information into the variables `OutputDevice.Status` and `OutputDevice.Name` from varbind[4].



12. Create a variable named `JobInfo`. Extract the job priority from varbind[3] into `JobInfo.Priority`. and `JobInfo.Target` and `JobInfo.JobID`.

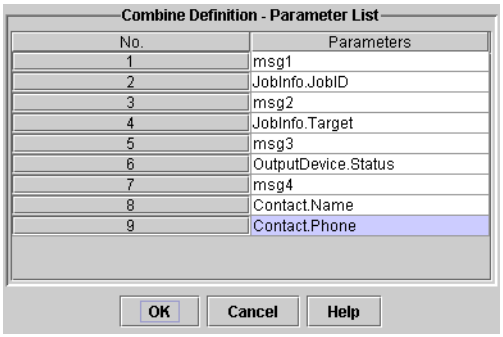


13. Create a Constant variable, `HiPri`, to hold the minimum interesting priority, “5”. Create a variable to hold the string “Normal”.
14. Extract the contact information from varbind [0] into `Contact.Name` and `Contact.Phone`.



15. Create constant variables to hold the parts of the message string. The final message string should read, “Cannot print job 345 to target ljet1because Toner Low. Contact John Bigboote, 729-315-4545.” The strings are “Cannot print job ”, “ to target ”, “ because ” “. Contact ”.

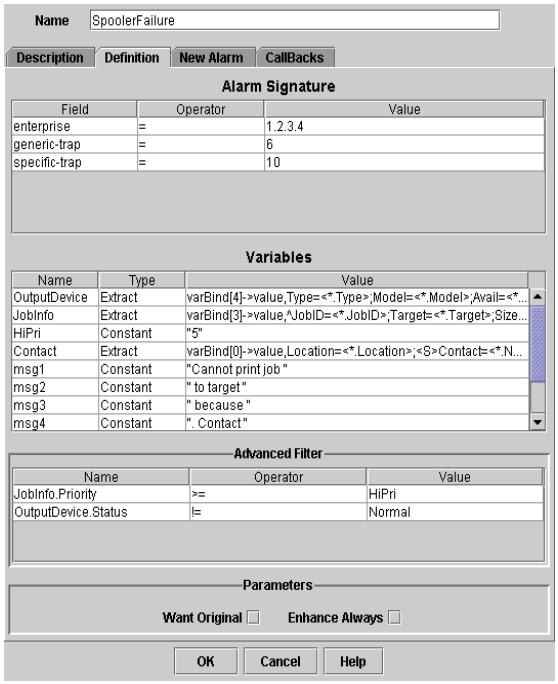
16. Create a Combine variable named ErrStr to build the message string.



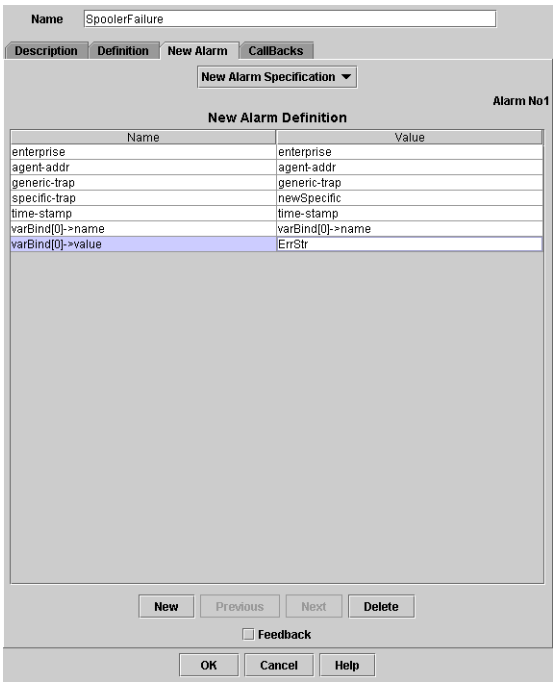
17. Create a variable `NewSpecific` to hold the specific ID for the new event, 11.
18. In the Advanced Filter, evaluate the incoming event's priority against `HiPri`. Also evaluate the incoming event's device status. Enter the following values:

Name	**Operator**	**Value**
JobInfo.Priority	>=	HiPri
OutputDevice.Status	!=	Normal

19. Allow the incoming event to be discarded.



20. Create the new event, placing the message string in varbind[0].



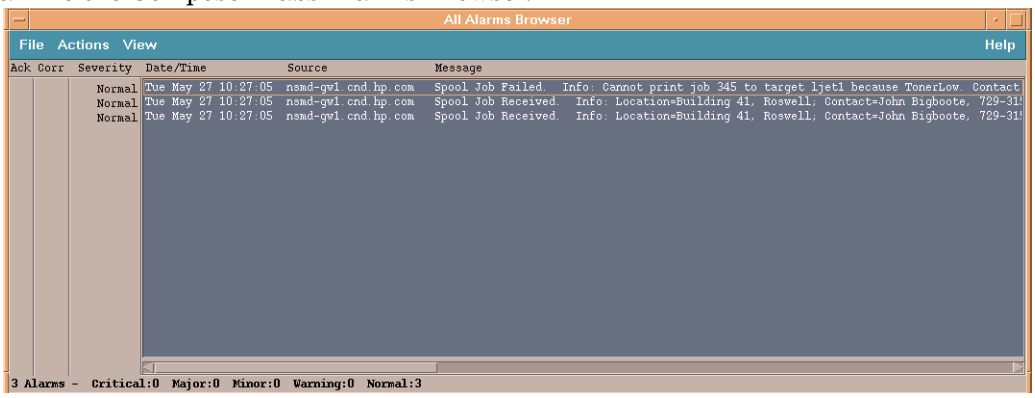
21. Save your Correlator Store in the `$OV_CONF/ecs/CIB` directory.
22. Exit the Composer developer interface.
23. Provide Operator access to your correlator.
24. Deploy your correlation.

Test Your Correlator

25. Type the commands:

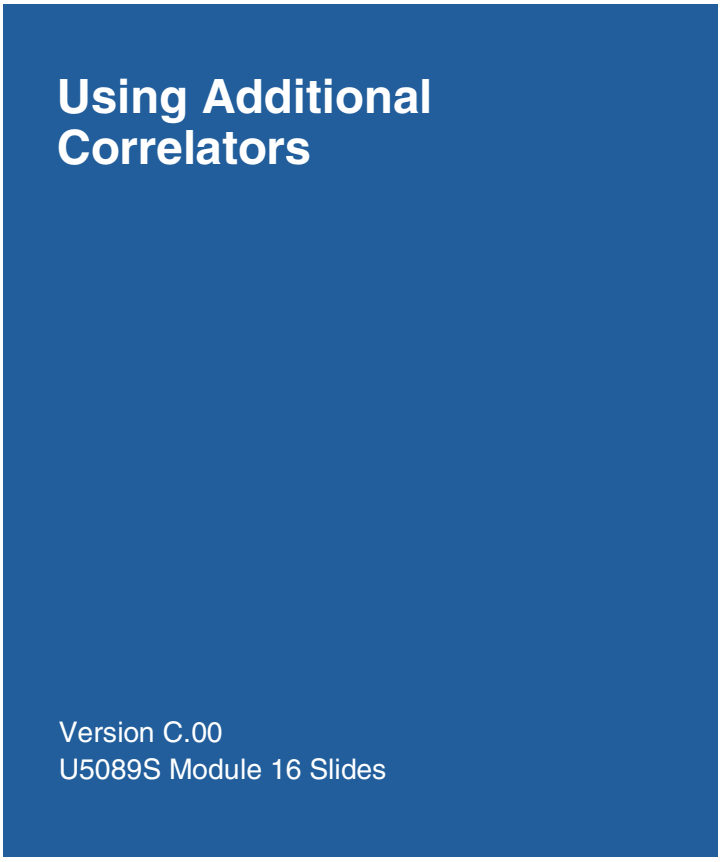
```
cd $OV_CONTRIB/ecs

ecsevgen -n spooler.evt
```

26. Examine the Composer Labs Alarms Browser.



16 Using Additional Correlators

Module Objectives



At the completion of this module, you will be able to:

- Define a Rate correlator.
- Describe the purpose of a message key.
- Design and specify a message key.
- List and access automatic variables.
- Define a Repeated correlator.
- Define a Transient correlator.
- Specify a Clear Alarm.

Rate Correlator Template

Rate Correlator Template

- The Rate Correlator Template can be used to count the number of events occurring within a specified time period.
- If the count equals the value specified within the time period, the threshold is considered breached and a new event is created.

©2004 Hewlett-Packard Development Company, L.P.

Rate Correlators are used to measure the number of incoming events within a defined time period. When a specified number is received, you can choose to discard the events and generate a new more meaningful event.

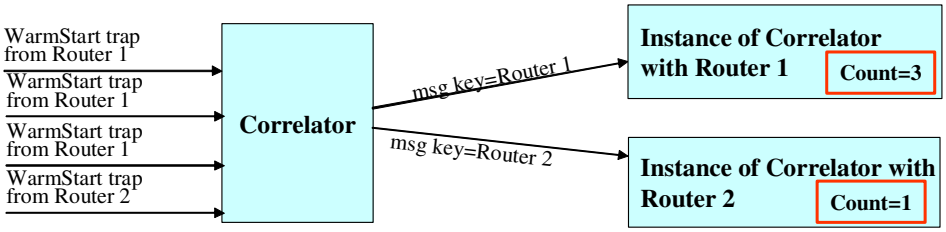
If the count equals the value specified within the time period, the threshold is considered breached and a new event is created. When the threshold is breached, the rule can be configured to:

- discard all events (regardless of rate) and emit only the newly created event.
- emit all events as they arrive and the newly created event (if any).

Message Key

Message Key

- Events whose Message Keys match are counted/correlated together.
- An implicit relationship between the various events in the group
  - Physical entity like an interface, a router
  - Logical entity like a service or customer
- Evaluated for each incoming event that passes the Alarm Signature and Advanced Filter
- Necessary only when two or more events need to be related
  - Used in Rate, Repeated, Transient and Multi-Source



U5089S ©2004 Hewlett-Packard Development Company, L.P. 3

When an event enters the correlator (the event has passed both the primary and secondary filters for the correlator), the Message Key is evaluated.

For example we could have a Multi-Source Correlator that has two events defined - a router down and an interface down event.

The idea here is that if a router fails, then discard individual interface (component) failures from the *same* router. The correlator specified however, is generic in the sense that it applies to all router down and interface down events. It is obvious that an interface down event should be suppressed only if the router to which this interface belongs to has also failed.

The mechanism to tie events together, in this case the router down event to the interface event from the same router, is the Message Key. Events that evaluate to the same value of the Message Key are correlated together.

Taking the above example, the name of the router could be used as a Message Key assuming that the router name can be extracted from both the router and the interface down events. The Message Key could be a physical entity like an interface or a router or a logical entity like a service, customer or PVC.

If there does not exist an instance of the correlator with the evaluated Message Key, an instance of the correlator is created with the Message Key. This is referred to as Correlator Creation.

On the other hand, when the event comes in, if there exists an instance of the correlator for the

same Message Key then the incoming event is correlated under the correlator for this Message Key. In other words this event is correlated with the other event(s) that evaluated to the same Message Key.

The Message Key is necessary only when two or more events need to be related. For example in Suppress there is no requirement for a Message Key as the correlator is applied to all events that meet the Alarm Signature and Advanced Filter conditions.

More Examples of Message Keys

- When the number of events received from the same router crosses 20 within 1 hour, trigger a new event indicating that the event rate is too high.

  All events matching the Alarm Signature, are categorized as Router events. But there has to be a mechanism by which to examine Router events emitted from the same router. To solve this problem, assign the agent-address as the Message Key.

- Monitor the rate at which interfaces on a router are failing. Configure a correlator called IF_Rate where the Message Key = x+Interface Number (varbind0), where x can be some attribute.

Automatic Variables

Automatic Variables

•Maintained by Composer

•**AlarmCnt** - the count of the number of events that entered the Correlator.

- Accessible while creating new events

•**CorrelationDuration** - the actual time taken for the Correlator to be applied.

- For example, while using Repeated correlation, a new event can be triggered when the window period is finished at 3 minutes. Then the CorrelationDuration has the value 180 seconds bound to it.

©2004 Hewlett-Packard Development Company, L.P.

Apart from the standard event attributes and the user-defined variables, Composer maintains some automatic variables. The variables maintained by the Composer are:

- `AlarmCnt` - The `AlarmCnt` attribute maintains the count of the number of events that entered the correlator. For example, if the correlation being used is Rate correlation, the attribute `AlarmCnt` maintains the count of the number of events arriving. The `AlarmCnt` variable is accessible while creating new events and defining Callback functions.
- `CorrelationDuration` - `CorrelationDuration` is the actual time taken for the correlator to be applied. For example, while using Rate correlation, a new event can be triggered when the rate exceeds 5 in 30 minutes. But if the rate has been breached in the 10th minute, then the `CorrelationDuration` has the value 600 bound to it.

Rate Example

Rate Example

•Problem Statement

- When a router is re-intialized a Start trap is emitted.
- Monitor the rate at which the Start traps come in and discard the Start traps.
- If more than 3 traps arrive from a given router in a 30 minute period, create a new event indicating the instability in the router.

•Solution Design

- Recognize traps emitted from the same router
  - Use the Message Key to identify traps to be correlated together.
- Maintain a count of the traps that arrived and the time in which the rate was breached.
  - Use automatic variables - **`AlarmCnt`**, **`CorrelationDuration`**
- Create the new event.

©2004 Hewlett-Packard Development Company, L.P.

Before you begin to configure this correlator, what do you need to know?

1. How do you recognize a Start trap?

   By the specific trap ID (this example uses 8).

2. How do you recognize traps from the same router?

   Traps with the same `agent-address` are correlated together by using it as the Message Key.

3. How is the count of the number of traps maintained?

   The automatic variable `AlarmCnt` maintains a count of events arrival.

4. How is the correlation time maintained?

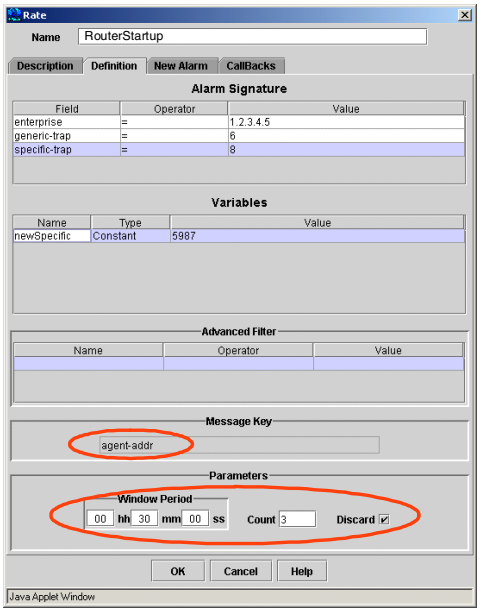
   The automatic variable `CorrelationDuration` maintains a duration of traps arrival.

5. What are the parameters of the new event?

   - `Specific-trap=newSpecific`. `newSpecific` must already been defined previously in the Variables section. (This example uses 5987.)
   - `variable-binding[0].value = AlarmCnt`.
   - `variable-binding[1].value=CorrelationDuration`.
   - Map all other attributes to the original trap's attributes.

Rate Example - Definition Window

Rate Example - Definition Window



- Count Start traps from the same router together
  - Message key differentiates the Start traps from specific routers.
  - The source of the Start trap is identified by the attribute “agent address”.
  - The “agent address” is used as Message Key, since the agent address contains the IP address of the router.
- Set threshold in Parameters section
  - 30 minute window
  - 3 occurrences for each router
- Discard original traps, just count them

U5089S ©2004 Hewlett-Packard Development Company, L.P. 6

Follow the procedure given below to configure this correlator

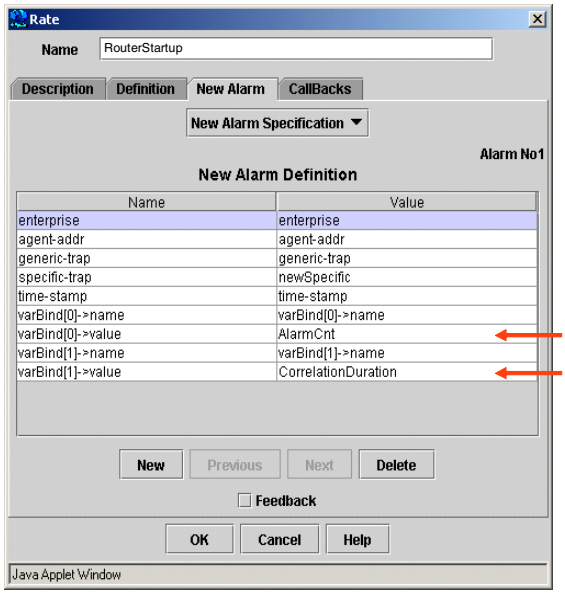
1. Select `Correlations:Correlator Templates->Rate` or click the Rate toolbar icon.
2. Enter the name of the correlator in the Name text field.
3. Enter the description of the correlator.
4. Click the Definition tab to display the Alarm Definition panel.

   Set the following values to define the Alarm Signature:

   - `enterprise = 1.2.3.4.5`
   - `generic-trap = 6`
   - `specific-trap = 8`
5. Create a Constant variable, `newSpecific`, with a value of 5987 for the new event ID.
6. In the Message Key section, select the `agent-addr` from the pop-up list.
7. Set the `Window Period` to 30 minutes.
8. Set the `Count` to 3.
9. Click the `Discard` button.

Rate Example - New Event Creation

New Event Creation



- When threshold is breached, what needs to be done?
  - Create new event.
  - For example, if 3 Start traps arrived from Router 1 within 30 minutes, a new event is generated indicating instability in Router 1.
  - Specific ID is from variable
  - AlarmCnt and CorrelationDuration are automatic variables

U5089S ©2004 Hewlett-Packard Development Company, L.P. 7

Follow the procedure given below to create a new event:

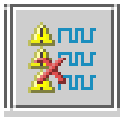
1. Select New Alarm Specification from the drop down menu in the New Alarms panel.
2. The New Alarm Definition table displays all the mandatory attributes that must be filled in. Select values for the fields from the pop-up list displayed.
   a. Set the `enterprise`, `agent-addr`, and `generic-trap` to match the incoming event.
   b. Set the `specific-trap` to use the variable `newSpecific`.
   c. Set the `time-stamp` to match the incoming event.
   d. Add a row for `varbind[0]->value` and set it to `AlarmCnt`.
   e. Add a row for `varbind[1]->value` and set it to `CorrelationDuration`.
3. Do not mark the event for feedback.
4. Click [OK].

Repeated Correlator Template

Repeated Correlator Template

•The Repeated Correlator Template can be used to:

- Discard duplicate events received within the window period of the first event
- Optionally send an update event at the end of the window period



U5089S ©2004 Hewlett-Packard Development Company, L.P. 8

Repeated correlators can be used to either discard duplicate events within a defined time period, or generate a new event each time an additional event is received so that the current number of events received can be specified in the event message text.

Repeated Correlation can operate in one of the following two modes:

- Mode 1: Duplicate events received within the window period of the first event are discarded.

  If the incoming event is to be discarded, you can configure it to participate in other correlations before it is discarded. You can also choose to send an update event at the end of the window period. This is typically used to create a new event indicating the number of events discarded by the first event in the window.

  Repeated correlation operates in this mode when the Discard Duplicate button is checked.

- Mode 2: Duplicate events are not discarded. If there is a specification for a new event to be created, a new event is created for every incoming event.

  This mode is typically used to send a new event to replace the previously sent one, with the count of duplicate events received so far.

Duplicate events can enter one of the following states:

- Output - The event is to be output.
- Discarded - The events are discarded from ECS. You can further decide if this event should

take part in other correlations

You can choose to discard or output events whenever required by using the `Discard Duplicate` button and the `Discard Immediately` button.

Discard Immediately	Yes	This is applicable only if the [`Discard Duplicate`] button is chosen. The effect of this is that all duplicate events will be discarded without participating in other correlators.
	No	Duplicate events will be discarded only after participating in other correlators.

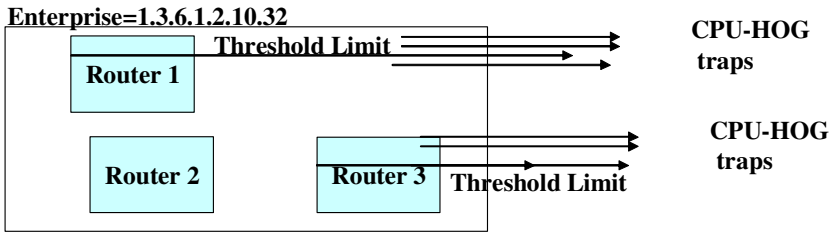
Repeated Example

Repeated Example

•Routers generate a CPU-HOG trap when the utilization exceeds a threshold.

•The requirement is to

- Pass only the first trap for a given router in a 30 minute window
- Discard all other traps received within same window period
- At the end of the 30 minutes emit a new event indicating the number of such traps received (and discarded).



©2004 Hewlett-Packard Development Company, L.P.

In general terms, duplicate events are messages that report the same activity. You can use Repeated Correlation to suppress duplicate messages based on a variety of suppression types. Repeated Correlation is used to monitor duplicate events arriving within the specified Window Period.

Routers generate a CPU-Hog trap when the utilization exceeds the threshold. The requirement is to pass only the first trap for a given router in a 30 minute time window and discard all other traps received in the same window. Additionally, at the end of the 30 minute period a new event must be generated indicating the number of such traps received (and discarded).

Before you begin to configure this correlator, what do you need to know?

A sample trap PDU appears in a log as:

```
Trap PDU{
enterprise {1 3 6 1 2 10 32},
agent-addr internet:"\x0A\x00\x01\x7F",
generic-trap 6
specific-trap 92
time-stamp 41474291
variable-bindings { }
```

```
}
```

1. How do you identify which traps are duplicate?

   All traps with the following attributes are identified as Duplicate traps.

   - enterprise is set to .1.3.6.1.2.10.32
   - generic-trap is set to 6
   - specific-trap is set to 92

2. How do you identify traps from the same router?

   Two traps are said to be coming from the same router if the agent address of the router is the same.

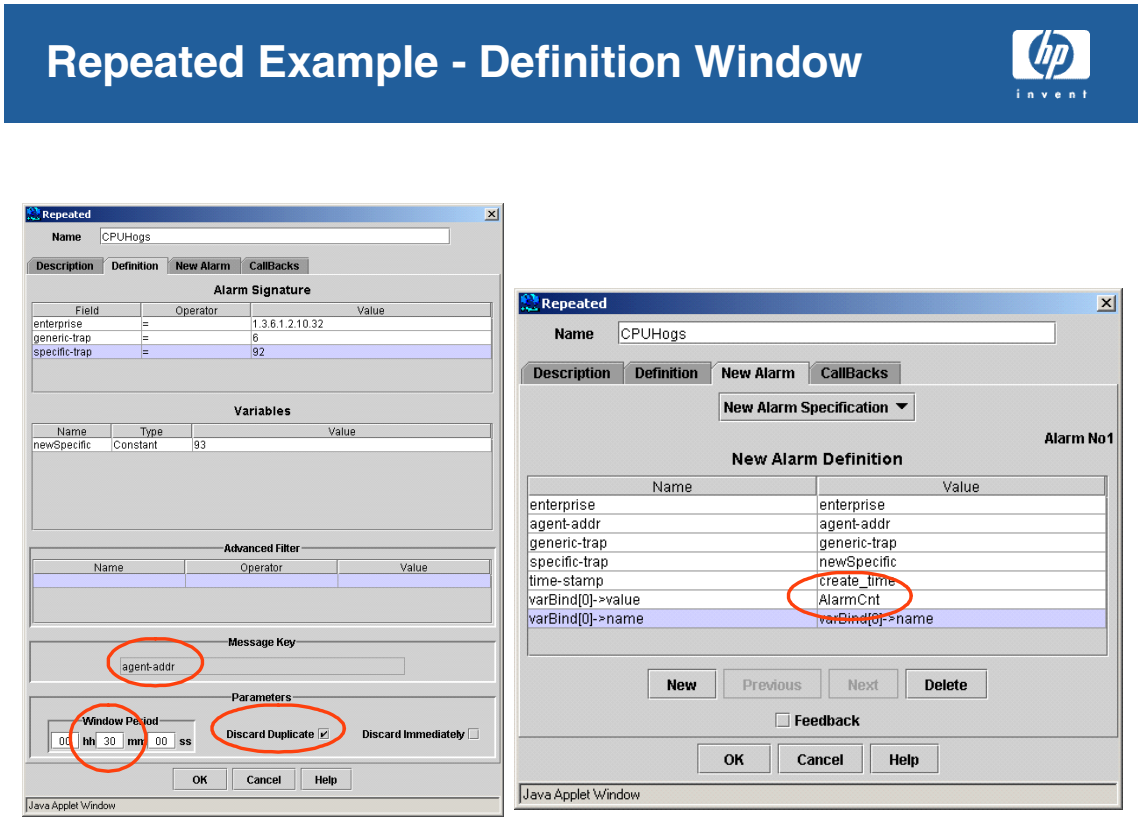
3. What do you want to do with the duplicate traps?

   In this example, discard them.

4. When should the traps be discarded or output?

   In this case, they do not participate in other correlators, so discard immediately.

Repeated Example - Definition Window



Follow the procedure given below to define the Repeated Correlator Template:

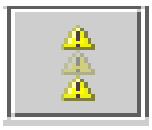
1. Select `Correlations:Correlator Templates->Repeated` from the Correlator Store window. You can also click the Repeated Correlator Template icon in the Correlator Templates toolbar.
2. Enter the following values to define the Alarm Signature:
   - `enterprise = 1.3.6.1.2.10.32`
   - `generic-trap = 6`
   - `specific-trap = 92`
3. Declare the following variables
   - `newSpecific` constant 93.
4. Select the `Message Key = agent-addr`.
5. Enter 30 minutes in the Window Period field.
6. Check the `Discard Duplicate` and `Discard Immediately` buttons, as duplicate events are to be discarded as soon as they have taken part in the this correlation and you do not wish for it to take part in other correlations.
7. Define the new event. Click the New Alarm tab.

8. Select `New Alarm Specification` from the drop down menu. The Create Alarm Definition table is displayed.
9. The table lists the mandatory fields to be filled for an event. Set the following values:
   - `enterprise = enterprise`
   - `agent-addr = agent-addr`
   - `generic-trap = generic-trap`
   - `specific-trap = newSpecific`
   - `time-stamp = create-time`
   - `varBind[0]->value = AlarmCnt`
10. Click [`OK`] to complete the definition of the Correlator. Notice that the Correlator you have just defined is displayed in the Correlator Store table.

Transient Correlator Template

Transient Correlator Template

- Transient Correlation is used to
  - Detect transient failures
  - Discard associated events
  - Monitor the rate of such transient failures and create a new event if a configured threshold is breached



©2004 Hewlett-Packard Development Company, L.P.

Transient correlator is used to detect a defined number of paired event occurrences within a defined time period, such as node up/node down. The paired events can be discarded, and a new more meaningful event can be generated.

A transient failure is when the state of a managed entity changes to abnormal and then reverts to normal, in a small period of time. Transient Correlation is typically used to detect transient failures and when a transient failure is detected, associated events are discarded. Additionally you can use this model to monitor the rate of such transient failures and create a new event if a configured threshold is breached. (The threshold is considered breached if the number of transient pairs in a time window equals the configured breach value.)

Transient Example

Transient Example

•Problem Statement

- PCM links go out-of-sync and a trap is generated indicating a link failure. Usually, however, the two ends of the link re-synch in 1 second.
- The requirement is:
  - to discard PCM_Down traps if PCM_UP trap arrives 2 seconds.
  - if 3 such link pair failures are detected in a 30 minute period, emit a new event with specific-trap set to 14 (all other fields are identical to the PCM trap).

•Solution Design

- Distinguish the CLEAR and FAIL traps. The [CLEAR] button in the parameter section of the Transient Correlator is used to make this distinction.
- Ensure they are coming from same PCM_Link.

©2004 Hewlett-Packard Development Company, L.P.

PCM Link trap PDU looks like:

```
Trap-PDU{
enterprise {1 2 3 4},
agent-addr internet : "\x0A\x00\x01\x7F",
generic-trap 6,
specific-trap 12,----------> Down trap. Up is 13.
time-stamp 414746291,
variable-bindings{
{
name {1 3 6 1 4 1 11 2 17 2 1 0}
value simple : number : 40 ----------> Link ID
}
}
}
```

1. How do you identify the PCM Link failure trap?

   All traps with the following attributes are identified as PCM Link failure traps.

- enterprise is set to 1.2.3.4
- generic-trap is set to 6
- specific-trap is set to 12

2. How do you identify traps from the same device?

   Two traps are said to be coming from the same device if the agent address and link-id both match.

3. How do you find the link-id?

   varbind[0] in the PCM trap contains PCM link.

4. What do you want to do with the original traps?

   Hold them for 2 seconds. If the PCM_UP trap arrives, discard the PCM_Down.

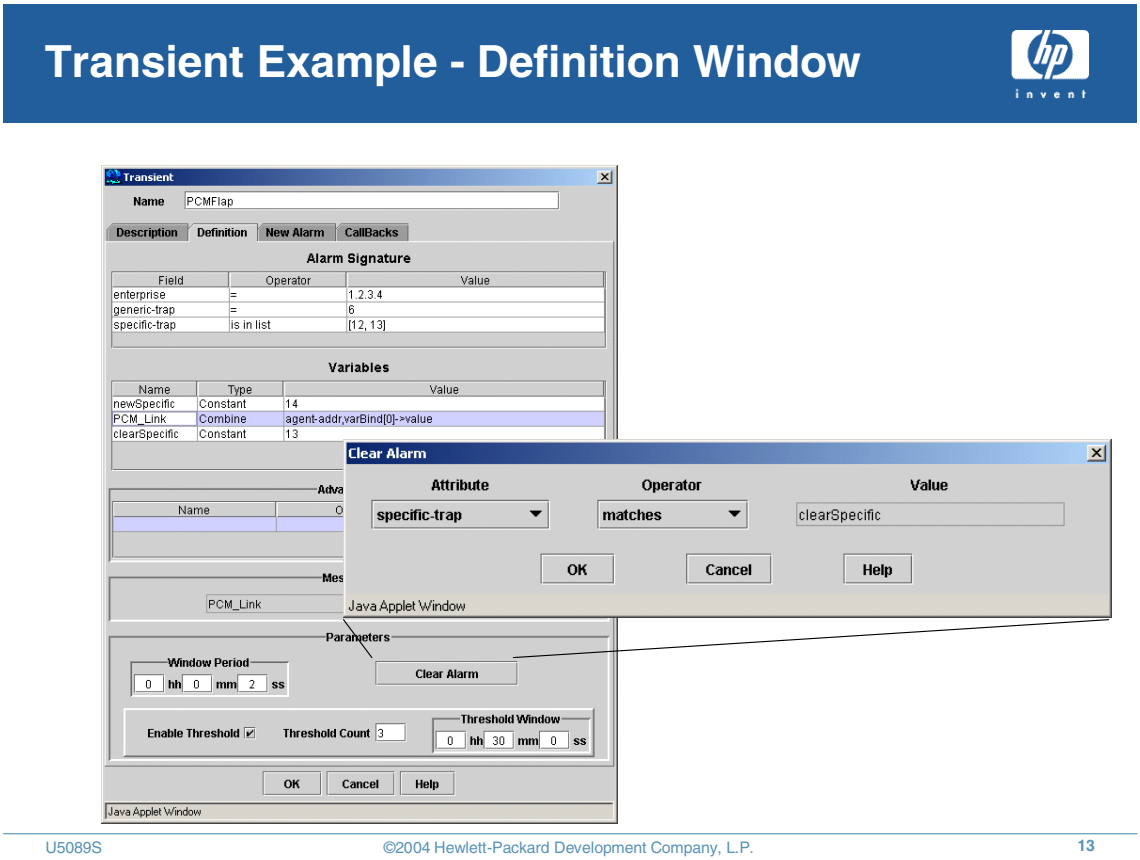
5. When should a new event be output?

   If 3 traps are counted in a 30 minute window.

6. What goes in the new event specification?

   Specific ID is 14. All other fields are identical to the PCM_UP trap.

Transient Example - Definition Window



 ©2004 Hewlett-Packard Development Company, L.P.

Follow the procedure given below to configure this correlator:

1. Select `Correlations:Correlator Templates->Transient` from the Correlator Store window. You can also click the Transient icon in the Correlator Templates toolbar.
2. Enter the name of the Correlator in the Name text field.
3. Enter the description of the Correlator. The Description can briefly state what the Correlator is expected to do.
4. Click the Definition tab to display the Alarm Definition panel.

   Set the following values to define the Alarm Signature:

   - `enterprise = 1.2.3.4.`
   - `generic-trap = 6`
   - `specific-trap matches [12,13]`
5. Create a Combine variable `PCM_Link`.

   a. In the Name cell, enter the name of the variable `PCM_Link`.

   b. Click in the Value cell to open the Combine Definition dialog.

   c. Select `agent-addr` as the first parameter.

d. Right-click and Add an additional row.

e. Select the varbind[0]->value.

f. Click [OK].

6. Create a Constant variable `newSpecific` to hold the event ID to be created.

   a. Give the variable the value of `14`.

7. Create a Constant variable `clearSpecific` to designate which incoming trap is the clear trap. Give it the value `13`.

8. In the Message Key section, select the `PCM_Link` from the pop-up list.

9. Set the Window Period to 2 seconds.

10. Click [`Clear Alarm`] to open the Clear Alarm dialog.

    a. Select `specific-trap` as the Attribute.

    b. Set the Operator to `matches`.

    c. Set the Value to `clearSpecific`.

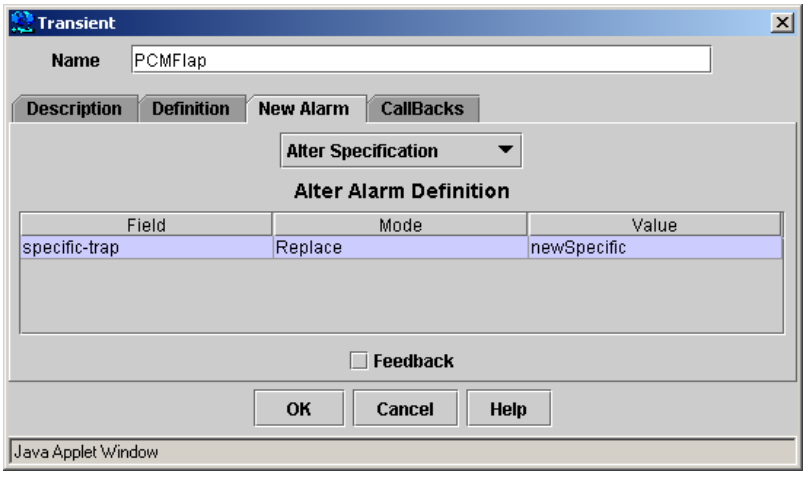
    d. Click [OK].

11. Enable Thresholds.

12. Set the Threshold Count to 3.

13. Set the Threshold Window to 30 minutes.

14. Select Alter Specification from the drop down menu in the New Alarms panel.



15. Change the trap number and leave the rest the same as the clear trap.

    a. Set the `specific-trap` to replace with the variable `newSpecific`.

16. Do not mark the event for feedback.

17. Click [OK].

Lab Exercises

- Transient correlator
- Rate correlator
- Repeated correlator

 ©2004 Hewlett-Packard Development Company, L.P.

Lab Case: Rate Correlation

Robotic arms on production lines may report failures during high volume conditions. The requirement is to discard all robotic arm failures if the rate of failure is below 5 failures in 30 minutes. If the rate exceeds this threshold, forward the last event to the browser after annotating the event with the rate.

Requirements

A sample SNMP trap PDU for an arm failure could appear in an event log as below:

```
Trap-PDU{
enterprise {1 2 3 4 }
agent-addr internet : "\x0A\x00\x01\x7F",
generic-trap 6,
specific trap 80,
```

```
time-stam 414746291,
variable-bindings{
{
name{1 3 6 1 4 11 2 17 2 1 0},
value simple:number : 2
},
{
name { 13 6 1 4 11 2 17 2 2 0},
value simple : string : "Arm#10#CTRL#20"
}
}
}
```

What do you need to know?

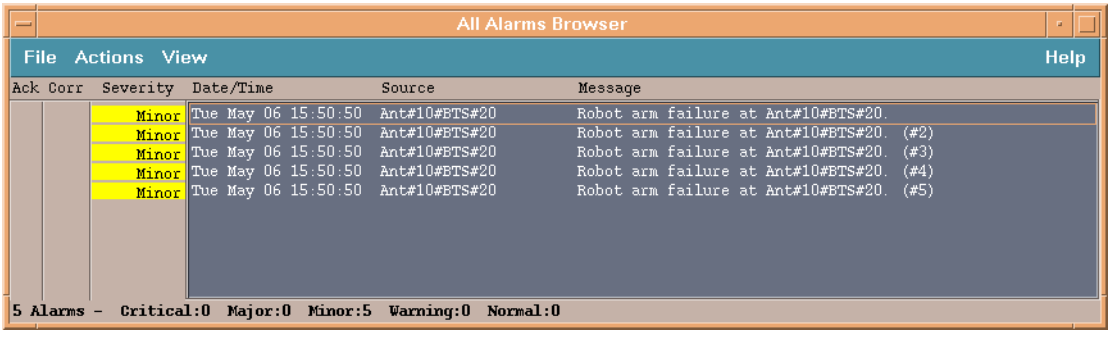
1. How do you identify the events for which the count will be maintained?
2. What do you with the events?
3. How will you match the traps?

View Pre-Correlator Results

1. A sample event stream from this environment is available for testing. Copy the following file from `$OV_CONTRIB/OVTraining/NNM3` to `$OV_CONTRIB/ecs`:
   - `robot_5.evt`: sends 5 traps.
2. Open the Composer Labs Alarms Browser.
3. Empty all old alarms by selecting `Actions:Delete->All Alarms in Category`.
4. Type the commands:

   ```
   cd $OV_CONTRIB/ecs
   ecsevgen -n robot_5.evt
   ```

5. Review the alarms in the Alarm Browser.



6. To mark which alarms came from this pre-test run, select `Actions:Acknowledge->All Alarms in Category`.

Directions

Follow the procedure given below to define the Rate Correlator Template:

7. Select `Correlations:Correlator Templates->Rate` from the Correlator Store. The Rate Correlator Template window opens. You can also click on the Rate Correlator icon in the Correlator Templates toolbar.
8. Enter the Name and Description for the Correlator.
9. Enter the following values to identify the Alarm Signature
   - enterprise
   - generic-trap
   - specific-trap
10. Declare the following variables
    - arm - This is a variable that in combination of the extracted pattern will specify the Arm ID and Controller ID.

      — Extract the Arm ID and Controller from variable-bindings[1].value. In the extract pattern window enter `Arm#<*.armid>#CTRL#<*.ctrlid>`

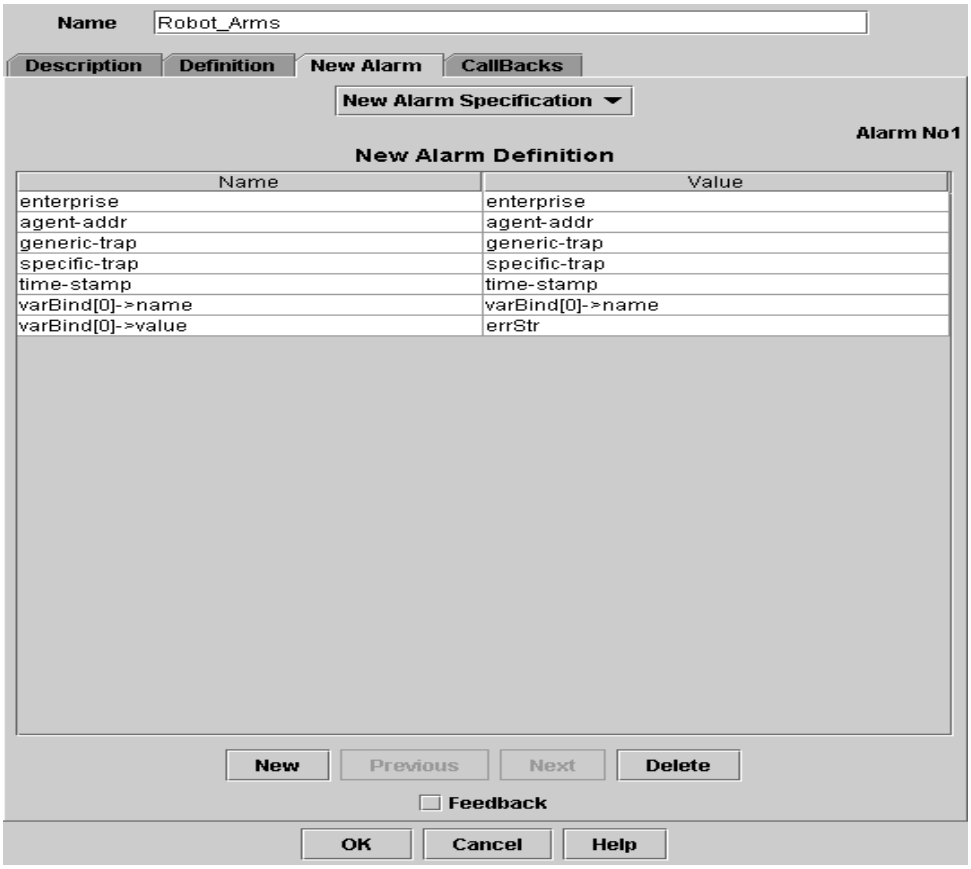
      — Leave the Pattern Separator field blank.
    - `mkey` - This is the unique field that will combine all the above attributes into one and constitute the Message Key.

      — Combine the attributes `arm.armid`, `arm.ctrlid`, `agent-addr`.
11. Select the Message Key. Click in the MessageKey window. A pop up menu displays all attributes and pre-defined variables. Select `mkey` from the menu.
12. Define the parameters for the correlation.
    - `Window Period` = 30 minutes
    - `Count` = 6
13. Select the Discard button. Although the events are discarded, the count of event arrival is maintained.
14. Before a new event is created, it is necessary to define the error string that declares the problem. Define the following variables in the Variable table.
    - `str1` constant “The threshold has been breached for the robotic arm ”
    - `str2` constant “ from Controller ”
    - `errstr` combine of `str1`, `arm.armid`, `str2`, `arm.ctrlid`

    The above definition creates an errstr which will look like “The threshold has been breached

for the robotic arm 10 from Controller 20".

15. Define the new event. Click on the New Alarms tab to alter the event. The New Alarm panel opens.
16. Select New Alarm Specification from the drop down menu. The New Alarm Definition table is displayed.
17. Select the following to define the change
    - enterprise = enterprise
    - agent-addr = agent-addr
    - generic-trap = generic-trap
    - specific-trap = specific-trap
    - time-stamp = time-stamp
    - varBind[0]->name=varBind[0]->name
    - varBind[0]->value = errstr
18. Click on [OK] to complete the definition of the Correlator. Notice that the Correlator you have just defined is displayed in the Correlator Store table.



19. Save your Correlator Store in the `$OV_CONF/ecs/CIB` directory.
20. Exit the Composer developer interface.
21. Provide Operator access to your correlator.
22. Deploy your correlation.

Test Your Correlator

23. Demonstrate that your correlation holds traps during the window. Type the commands:

```
cd $OV_CONTRIB/ecs

ecsevgen -n robot_5.evt
```

24. Demonstrate that your correlation creates a new event when too many arrive. Type the `ecsevgen` command one more time.

LAB: Using Transient Correlator

This Lab Session requires:

- Transient Correlator
- New Alarm Section
- Global Constant `AlarmCnt`

Correlation Requirement

One of the A/C machines in a floor is having problems. It runs for a while and stops and starts again after a while. The facility management system manages all this equipment. It emits up and down events for the A/C machine.

- The varBind [0] value of the up and down events provides the floor number.
- The varBind [1] value of the up and down events provides the room temperature.

Create a correlator that suppresses transient events indicating ups and downs, when the room temperature is with in an acceptable range. If the temperature is not in the acceptable range, then send a new event for every 10 minutes with information of number of ups and downs in that window and suppress all the ups and downs events. If 3 up and down events emitted within 10 minutes, then create a new event.

Alarm Definition

Example Trap-PDU

```
Trap-PDU {
    enterprise {1 2 3 4},
    agent-addr internet : "\x7f\x00\x00\x02",
    generic-trap 6
    specific-trap 12,
    time-stamp 414746291,
    variable-bindings {
{
    name {1 3 6 1},
    value simple : number : 2
}
{
    name {1 3 6 8},
    value simple : number : 27
}


    }
}
```

AC Up and Down events are identified by:

- enterprise id set to 1.2.3.4

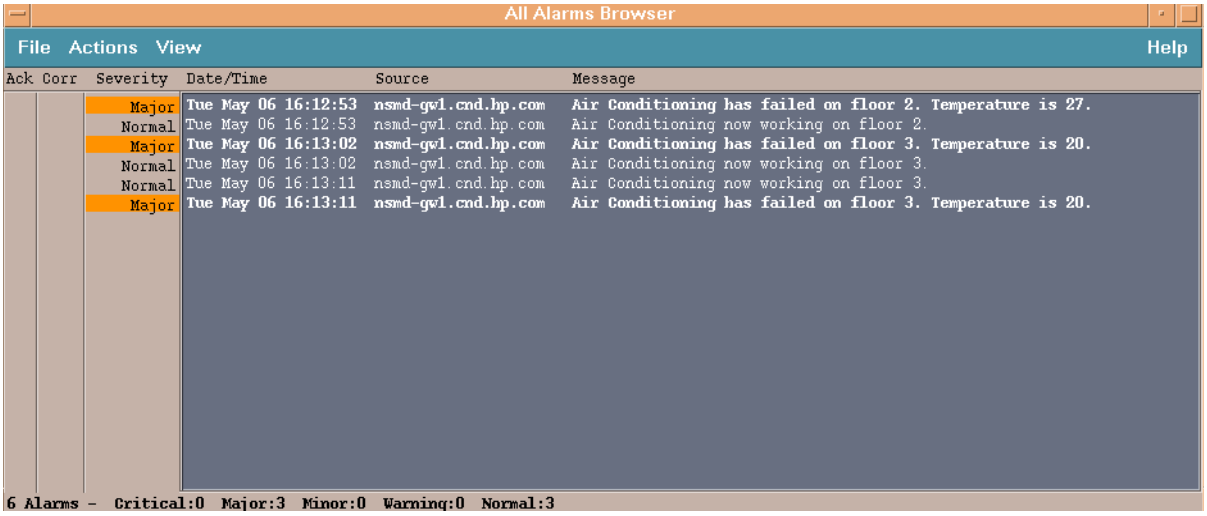
- generic-trap set to 6
- specific-trap set to 12 or 13 (12 is down event and 13 is up event)
- var-bind[0] contains Floor number
- var-bind[1] contains temperature

View Pre-Correlator Results

1. A sample event stream from this environment is available for testing. Copy the following files from `$OV_CONTRIB/OVTraining/NNM3` to `$OV_CONTRIB/ecs`:
   - `AC_hot.evt`: send a down, then an up trap with the temperature set to 27.
   - `AC_cool.evt`: send a down, then an up trap with the temperature set to 20.
   - `AC_reverse.evt`: send an up then a down trap.
2. Open the Composer Labs Alarms Browser.
3. Empty all old alarms by selecting `Actions:Delete->All Alarms in Category`.
4. Type the commands:

```
cd $OV_CONTRIB/ecs

ecsevgen -n AC_hot.evt

ecsevgen -n AC_cool.evt

ecsevgen -n AC_reverse.evt
```

5. Review the alarms in the Alarm Browser.



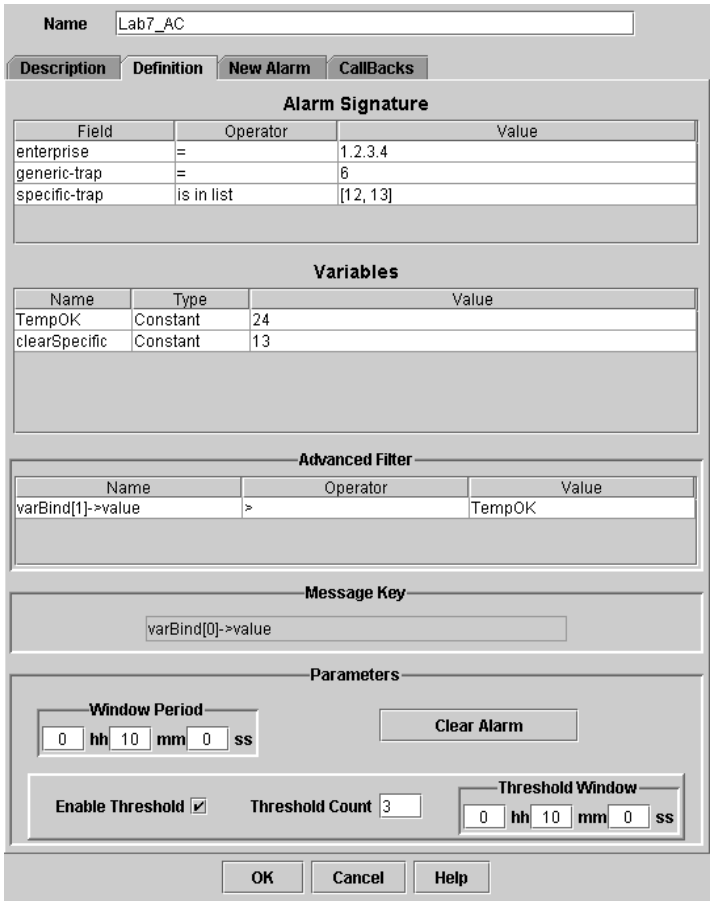
6. To mark which alarms came from this pre-test run, select `Actions:Acknowledge->All Alarms in Category`.

Directions

Execute the following steps to solve the lab:

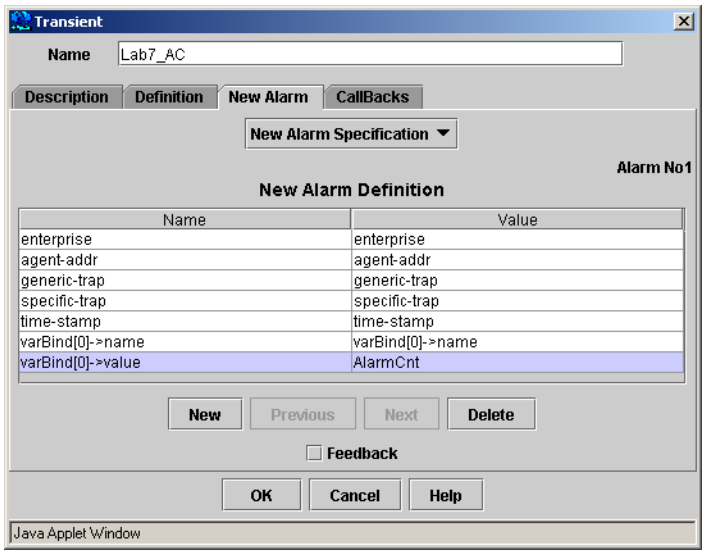
7. Use the Transient Correlator Template.
8. In the Alarm Signature section, filter the AC up/down events.
9. Create a variable to hold the acceptable temperature value. For example, if the temperature is >= 24, then it is considered outside the acceptable range.

10. The varbind[1].value contains the temperature value. In the Advanced Filter, compare the varbind[1].value with the temperature variable.
11. The varbind[0].value contains the floor number. To correlate all events coming from the same floor, use varbind[0].value as the Message Key.
12. Specify the Window Period.
13. Specify the Clear Alarm. In this case, specific trap 13 is the clear event.
14. Enable the Threshold Count and specify count and Threshold Window period.



15. Create a new event and map the required attributes and bind automatic variable `AlarmCnt` to

any attribute in the new event.



16. Save your Correlator Store in the `$OV_CONF/ecs/CIB` directory.
17. Exit the Composer developer interface.
18. Provide Operator access to your correlator.
19. Deploy your correlation.

Test Your Correlator

20. Demonstrate that your correlation suppresses an initial down event when an up event arrives. Type the commands:

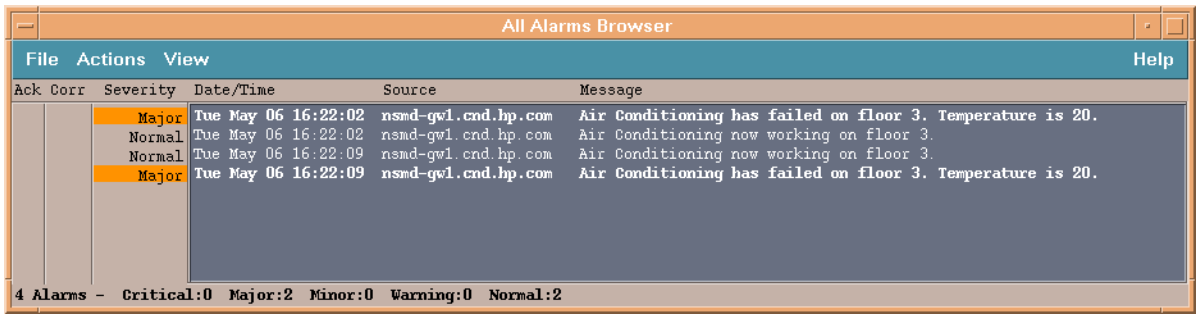
```
cd $OV_CONTRIB/ecs

ecsevgen -n AC_hot.evt
```

21. Demonstrate that your correlation creates a new event if the A/C unit goes up and down several times. Type the `ecsevgen` command two more times. (Not shown in results below.)
22. Demonstrate that your correlation does not suppress the down event if the room is not overheating. Type the command:

```
ecsevgen -n AC_cool.evt
```

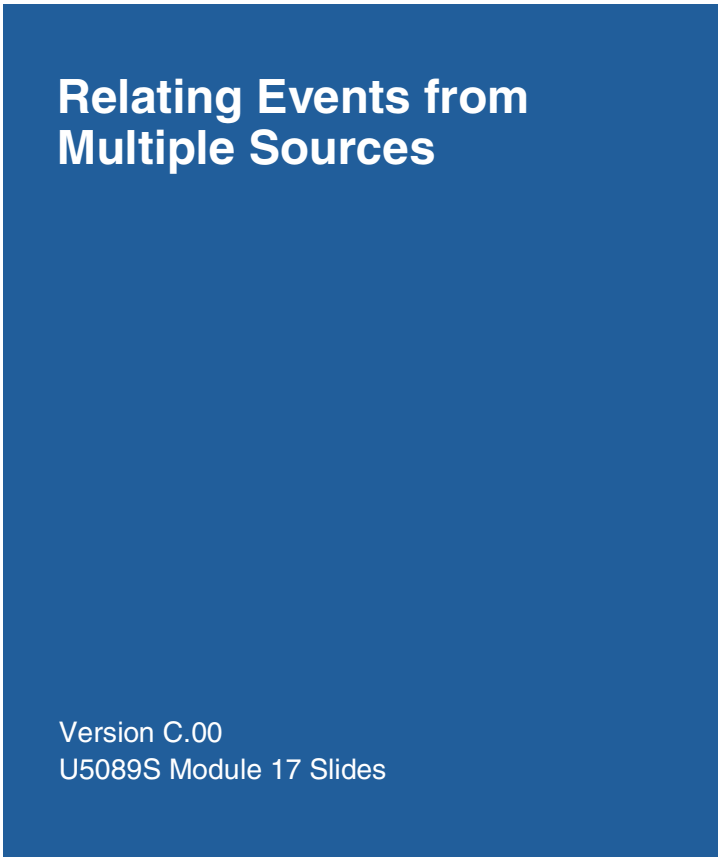
23. Demonstrate that your correlation does not suppress the events if the down arrives before the up event. Type the command:

```
ecsevgen -n AC_reverse.evt
```



17 Relating Events from Multiple Sources

Module Objectives



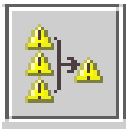
At the completion of this module, you will be able to:

- Define a Multi-Source correlator.
- Differentiate between a set of events and events that do not require a set.

Multi-Source Correlator Template

Multi-Source Correlator Template

- The Multi-Source Correlator Template is used to:
  - Define a relationship among an arbitrary number of event types that together form a logical set that identifies the problem.
- The flexibility of Multi-Source correlator is that events from different sources can be correlated.
- Each trap definition has its own Message Key section.
  - Traps with the same Message Key value are correlated together.



©2004 Hewlett-Packard Development Company, L.P.

A Multi-Source Correlator can be used to define a relationship among an arbitrary number of events, potentially from different sources, that together form a logical set that identifies a problem. The set of events must all arrive within a defined time period.

Set Definition

Set Definition

- At least one of each listed event type
- With the same Message Key
- Arrives within the window period
- The order of arrival is NOT important
- Choose Mode
  - Correlate all matches together
    - Collect related events
  - Require exactly one of each type
    - Duplicate starts new set
- Multi-Source Correlation can be used on set completion to:
  - Discard a subset of events (of specific type)
  - Modify a subset of events
  - Create one or more new events
    - Attributes from all events of the set can be used to define or modify the event.

U5089S ©2004 Hewlett-Packard Development Company, L.P. 3

Multi-Source correlators look for sets of related events. Set completion can operate in one of two modes:

- Mode 1: Collect all related events: The Multi-Source correlation operates in this mode by default. When the set is deemed complete, the instance of the set remains in a completed state for the duration of the time window. This is typically used in a situation where all events from a source can be discarded if caused by the failure of another entity. The Multi-Source correlation works in this mode when the Set button is NOT checked.
- Mode 2: Collect precisely one of each event type: When Multi-Source correlation is configured to operate in this mode, it is expected that events will arrive in pre-defined sets. In this case the requirement is that the correlation rule is applied as soon as the set is completed.

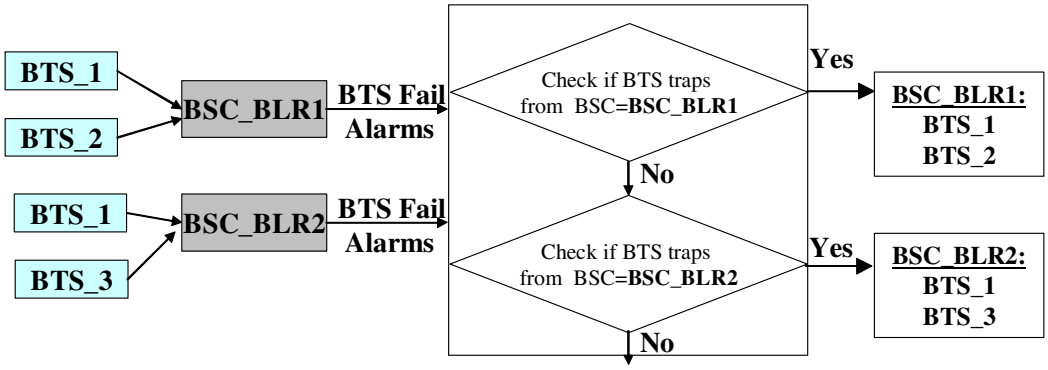
Multi-Source Correlation can be used on set completion to:

- discard a subset of events.
- modify a subset of events with attributes defined from any or all of the other events in the set.
- create one or more new events with values called from attributes or pre-defined variables from the other events in the set.

MultiSource Example

MultiSource Example

- When a BSC fails, all BTSs connected to the BSC emit traps.
- The requirement is to discard all BTS-fail traps for 10 seconds.
- BTS traps emitted from different BSCs have the same BTS names.
- BTS traps must be differentiated based on the parent BSC.
- The BSC and the BTS traps are generated by different sources and look different. (The alarm signatures are different.)



U5089S ©2004 Hewlett-Packard Development Company, L.P. 4

BSC stands for Base Station Controller. BTS stands for Base Transceiver Station. They are used in cell phone tower communication.

No new events are required.

How do you identify BSC and BTS traps?

A BSC trap PDU looks like:

```
Trap-PDU{
enterprise {1 2 3 4},
agent-addr internet : "\x0A\x00\x01\x7F",
generic-trap 6,
specific-trap 15,
time-stamp 414746291,
variable-bindings{
{
name {1 2 3 4 1},
value simple : string : "BSC_BLR_1" -------> varbind[0] string contains BSC name
```

```
}
 }
}
```

A BTS trap PDU looks like:

```
Trap-PDU{
enterprise {1 2 3 4},
agent-addr internet : "\x0A\x00\x01\x7F",
generic-trap 6,
specific-trap 16,
time-stamp 414746291,
variable-bindings{
{
name {1 2 3 4 1},
value simple : string : "BTS_1:BSC_BLR_1" ------> varbind[0] part of string contains BSC name

}
 }
}
```

What do you need to know?

1. How do you identify a BSC failure trap?

   The specific ID is 15.

2. How do you identify a BTS failure trap?

   The specific ID is 16.

3. How do you know from which BSC a trap is emitted?

   The BSC name is specified in the BSC trap's variable-binding[0].value.

4. How do you know the BSC and BTS are connected?

   Studying the traps alone does not help identify how the traps are related to each other.

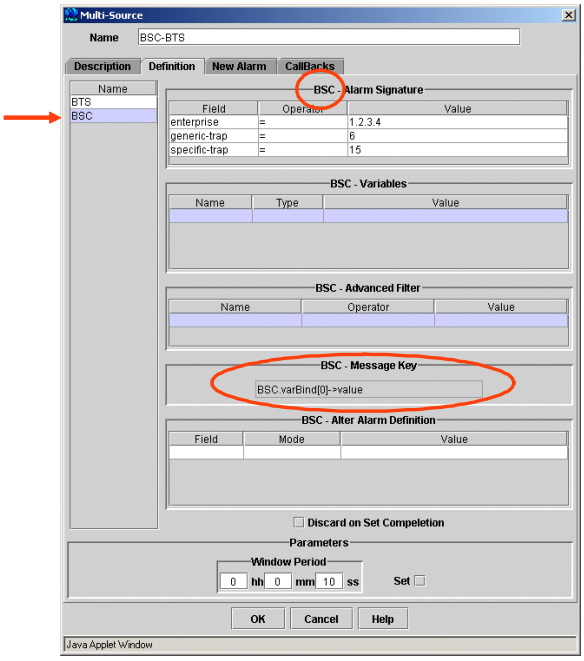
   - In the BSC trap, Message Key = `variable-binding[0].value` which is the name of the BSC.
   - In the BTS trap, Message Key = extracted BSC name from the BTS trap (again from `variable-binding[0].value`).

5. How do you know that both traps are correlated under the same instance?

   The Message Keys must be the same. Only then do the traps get correlated under the same instance.

MultiSource Example - BSC Definition

BSC Definition Window



- Identify incoming traps in left-hand Name field.
- Configure Alarm Signature for each trap type.
- Message Key must be available for all incoming trap types.

©2004 Hewlett-Packard Development Company, L.P.

Follow the procedure given below to configure this correlator:

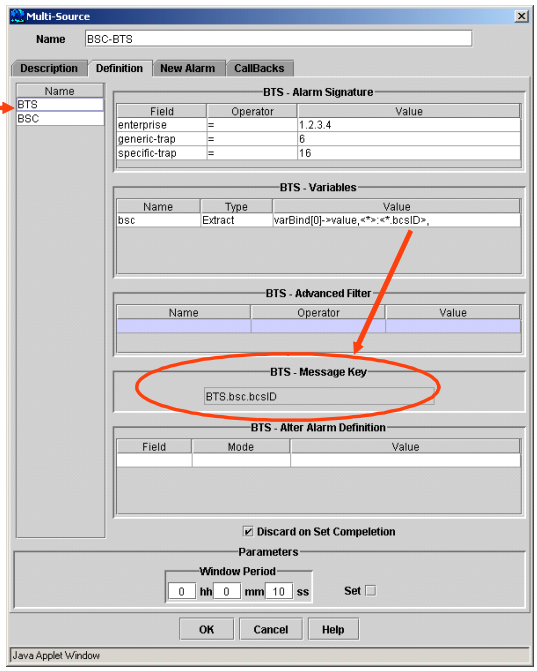
1. Select `Correlations:Correlator Templates->Multi-Source` or click the Multi-Source toolbar icon.
2. Enter the name of the Correlator in the Name text field.
3. Enter the description of the Correlator.
4. Click the Definition tab to display the Alarm Definition panel.
5. In the left pane, identify the first incoming trap type as BSC.
6. Set the following values to define the Alarm Signature:
   - `enterprise = 1.2.3.4`
   - `generic-trap = 6`
   - `specific-trap = 15`
7. You identify the incoming source events individually and link them using the Message Key field. The Message Key you design must be available for all incoming event types.

   In the Message Key section, select `BSC`, then select the `varbind[0]->value` from the pop-up list.

8. Set the Window Period to 10 seconds.
9. Do not click the `Set` button.

MultiSource Example - BTS Definition

BTS Definition Window



©2004 Hewlett-Packard Development Company, L.P.

Follow the procedure given below to configure this correlator.

1. Identify the incoming source events for the BTS.

   a. In the left pane, identify the next incoming trap type as BTS. Notice that the field labels change.

2. Set the following values to define the Alarm Signature:

   - `enterprise = 1.2.3.4`
   - `generic-trap = 6`
   - `specific-trap = 16`

3. Create an Extract variable `bsc` to get the BSC identifier from the BTS trap.

   Variables declared under the same multi-source correlator name are visible to all the event instances under that correlator. For example, if there are event instances called e1 and e2 under multi-source correlator multi_1, then all variables of e1 are visible to e2 and vice versa. But variables of e1 are not visible to, say event instance e3, under a different correlator instance say multi_2. For cross correlator visibility, global constant mechanism should be used, or you can use the built-in functions Store and StoreStr.

   a. In the Name cell, enter the name of the variable `bsc`.

b. Click in the Value cell to open the Extract Pattern dialog.

c. Select `varbind[0]->value` as the Attribute.

d. Type the Pattern `BTS_<#.btsID>:BSC_<#.bscID>`.

e. Type `#` for the Pattern Separator.

f. Click [`OK`].

4. In the Message Key section, select `BTS`, then select the `bsc`, then select `bcsID` from the pop-up list.
5. Mark the BTS traps to be discarded on set completion.
6. Set the Window Period to 10 seconds.
7. Do not click the `Set` button.
8. Click [`OK`] to complete the Correlator. Notice that the correlation you have just defined is displayed in the Correlator Store table.

Another MultiSource Example

Another MultiSource Example

•Traps A, B and C are said to be part of the same set if the agent-addr are the same.

•If traps A, B and C come within a 10 second interval then:

- Discard A and B
- Modify C's varbind1 to the string “C got modified”
- Create a new event D with
  - specific trap set to 20
  - varbind[0] value is varbind[0] of trap A
  - varbind[1] is varbind[0] of trap B
  - varbind[2] is varbind[0] of trap C.

©2004 Hewlett-Packard Development Company, L.P.

1. How do you identify Type A, Type B and Type C traps?

   A Type A trap is identified by:

   - enterprise oid is set to 1.2.3.4
   - specific trap is set to 17

   A Type B trap is identified by:

   - enterprise oid is set to 1.2.3.4
   - specific trap is set to 18

   A Type C trap is identified by:

   - enterprise oid is set to 1.2.3.4
   - specific trap is set to 19

   You need exactly one of each type of trap to complete a set.

2. What Message Key do you use?

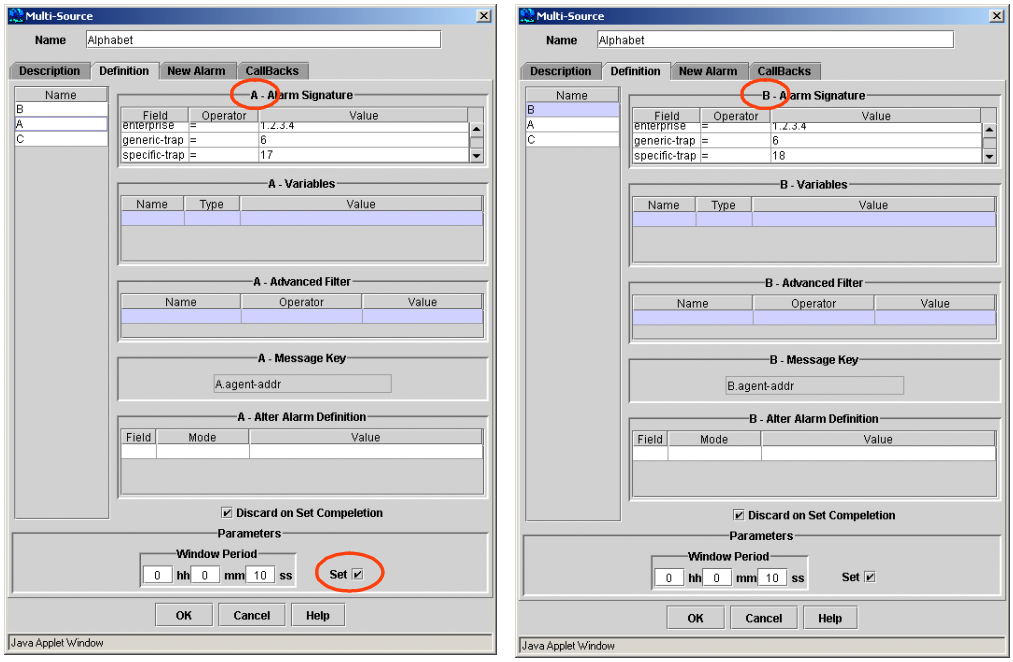
   The agent-address provides the Message Key.

3. What do you do when a set is complete?

- Discard Trap A.
- Discard Trap B.
- Modify C's varbind[1].value to "C got modified".
- Create a new event as described.

Another MultiSource Example - Trap A & B Definitions

Traps A & B Definition Windows



 ©2004 Hewlett-Packard Development Company, L.P.

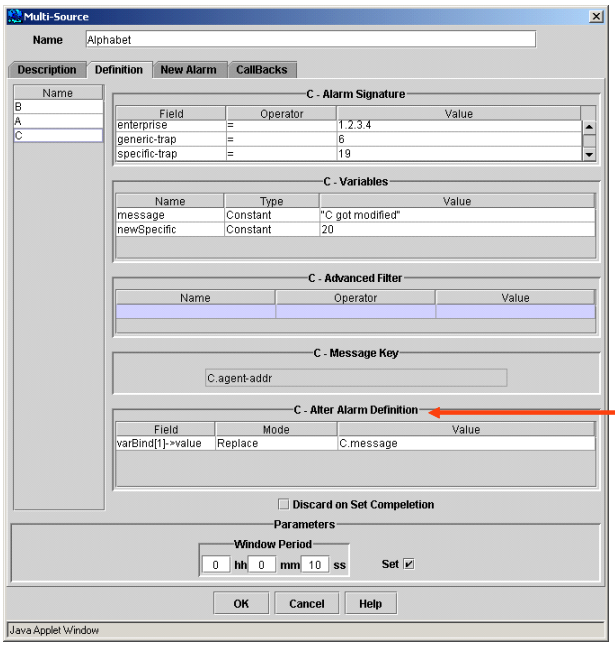
Follow the procedure given below to configure this correlator:

1. Select `Correlations:Correlator Templates->Multi-Source` from the Correlator Store window. You can also click the Multi-Source icon in the Correlator Templates toolbar.
2. Enter the name of the Correlator in the Name text field.
3. Enter the description of the Correlator. The Description can briefly state what the Correlator is expected to do.
4. Click the Definition tab to display the Alarm Definition panel.
5. Identify the incoming source events.
   a. In the left pane, identify the first incoming trap type as A.
6. Set the following values to define the Alarm Signature:
   - `enterprise = 1.2.3.4`
   - `generic-trap = 6`
   - `specific-trap = 17`
7. In the Message Key section, select the `agent-addr` from the pop-up list.

8. Mark A's to be discarded on set completion.
9. Set the Window Period to 10 seconds.
10. Click the `Set` button. This ensures that only one of each trap type gets correlated together.
11. Identify the next incoming source events.
    a. In the left pane, identify the next incoming trap type as B.
12. Set the following values to define the Alarm Signature:
    - `enterprise = 1.2.3.4`
    - `generic-trap = 6`
    - `specific-trap = 18`
13. In the Message Key section, select the `agent-addr` from the pop-up list. Note that it is listed as `B.agent-addr`.
14. Mark B's to be discarded on set completion.
15. Set the Window Period to 10 seconds.
16. Click the `Set` button.

Another MultiSource Example - Trap C Definition

Trap C Definition Window



- Define variables to use in new event creation
- Modify C's event definition

©2004 Hewlett-Packard Development Company, L.P.

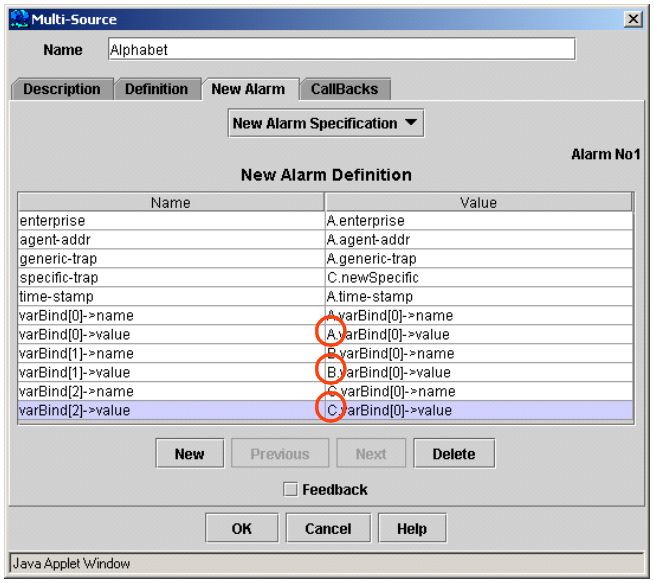
Follow the procedure given below to configure this correlator

1. Identify the incoming source events.
   a. In the left pane, identify the "last" incoming trap type as C. (Remember that the actual traps may arrive in any order; this is just the order of defining them in this example.)
2. Set the following values to define the Alarm Signature:
   - `enterprise = 1.2.3.4`
   - `generic-trap = 6`
   - `specific-trap = 19`
3. In the Message Key section, select `agent-addr` from the pop-up list.
4. Create the following variables to be used in the new event creation:
   a. A Constant variable named `message` which contains the string `"C got modified."`
   b. A Constant variable named `newSpecific` which contains the event ID `20`.
5. Do not mark the C traps to be discarded on set completion.

6. Modify the Alter Alarm Definition for C:
   a. In the Field, select `varbind[1]->value`.
   b. Set the mode to `Replace`, not `Append`.
   c. For the value, select configuration `C`, then select the variable `message`.
7. Set the Window Period to 10 seconds.
8. Click the `Set` button.

Another MultiSource Example - New Event

Creating a New Event



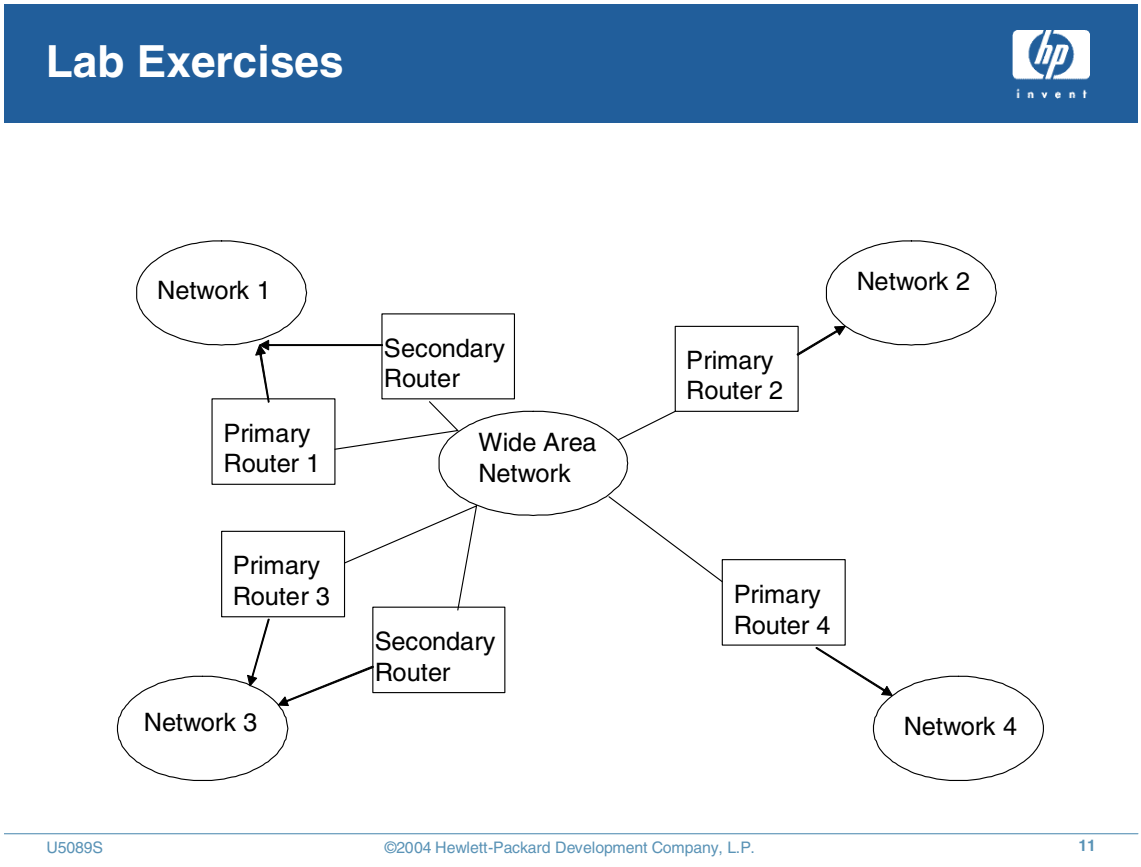
- Attributes from all traps are available in the new event definition

©2004 Hewlett-Packard Development Company, L.P.

Follow the procedure given below to create a new event.

1. Select New Alarm Specification from the drop down menu in the New Alarms panel.
2. The New Alarm Definition table displays all the mandatory attributes that must be filled in. Select values for the fields from the pop-up list displayed.
   a. Set the `enterprise`, `agent-addr`, and `generic-trap` to match trap A's values by selecting trap `A`, then selecting it's attributes.
   b. Set the `specific-trap` to match trap `C`.
   c. Set the `time-stamp` to match trap `A`'s.
   d. Add a row for `varbind[0]->value` and set it to trap `A`'s `varbind[0]->value`.
   e. Add a row for `varbind[1]->value` and set it to trap B's `varbind[0]->value`.
   f. Add a row for `varbind[2]->value` and set it to trap C's `varbind[0]->value`.
3. Do not mark the event for feedback.
4. Click [OK].

Lab Exercises



Lab requires:

- Multi Source Correlator

Requirements

Assume a wide area network with many small networks interconnected by routers (primary and secondary routers to share the load). Some sub-networks are connected by one router (primary) as shown in the slide.

If primary router is down and if a secondary router is connected, then all the traffic is diverted through secondary router. Due to heavy traffic, the secondary router will generate many “Traffic-Threshold-Breached” traps. The requirements are:

- When the Primary Router down trap is emitted, discard all the “Traffic-Threshold-Breached” traps from the secondary router for 30 seconds.

- Output the original Primary Router Down trap.
- Make sure that the trap sent is from the same primary and secondary router connecting the same 2 networks.

The router is uniquely identified by a combination of enterprise ID + Router ID (varBind[0] value). If the secondary router is connected, then only from those routers, the threshold alarms should be discarded.

Whether the secondary router is available or not is stored in the datastore. Passing the unique router as a key to the datastore can identify it.

The dataStore looks like:

```
ADD DATA ("<enterpriseid + Routerid>", "<Yes/No>" )
```

Trap Definitions

A router down trap is identified by

- enterprise id set to 1.5.6.7.8
- specific-trap set to 8
- var-bind0 contains router id

Example Trap-PDU

```
Trap-PDU {
    enterprise {1 5 6 7 8},
    agent-addr internet : "\x7f\x00\x00\x02",
    specific-trap 8,
    time-stamp 414746291,
    variable-bindings {
{
    name {1 3 6 1 4},
    value simple : number : 10
}
}
}
```

A Threshold breached trap is identified by

- enterprise id set to 1.5.6.7.8
- specific-trap set to 9
- var-bind0 contains router id of the primary router it is backing up

Example Trap-PDU

```
Trap-PDU {
    enterprise {1 5 6 7 8},
    agent-addr internet : "\x7f\x00\x00\x03",
    specific-trap 9,
    time-stamp 414746291,
```

```
    variable-bindings {
{
    name {1 3 6 1 4},
    value simple : number : 10
}
}
}
```

View Pre-Correlator Results

1. A sample event stream from this environment is available for testing. Copy the following file from `$OV_CONTRIB/OVTraining/NNM3` to `$OV_CONTRIB/ecs`:
   - `secondaryRouter.evt`: sends the following events:

     Router down

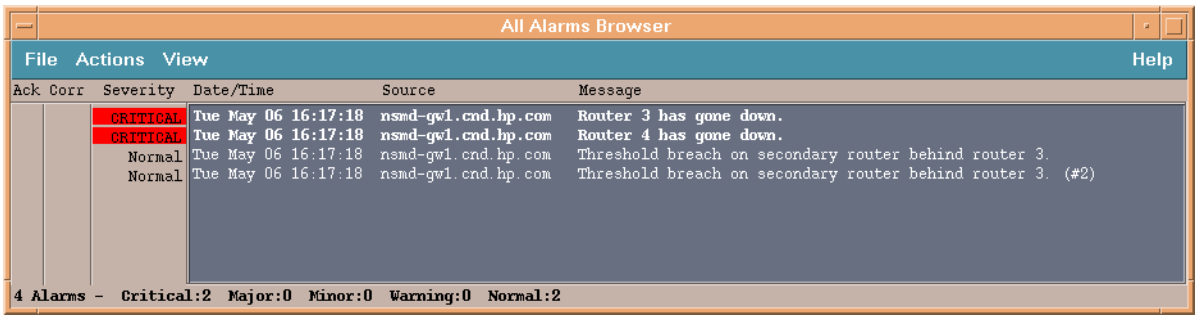
     f

     — rom a router that has a secondary.

     — Router down from a router that does not have a secondary.

     — Threshold breach from secondary router (2 times).
2. Open the Composer Labs Alarms Browser.
3. Empty all old alarms by selecting `Actions:Delete->All Alarms in Category`.
4. Type the commands:

   ```
   cd $OV_CONTRIB/ecs
   ecsevgen -n secondaryRouter.evt
   ```
5. Review the alarms in the Alarm Browser.



6. To mark which alarms came from this pre-test run, select `Actions:Acknowledge->All Alarms in Category`.

Directions

Execute the following steps to solve the exercise:

7. Use the Multi-Source Correaltor Template.
8. Create two types of events to filter:
   a. Router Down traps

b. Threshold Breach traps

9. Specify the Alarm Signature for each type of event.
10. A router is uniquely identified by the combination of enterprise ID + Router ID (varBind[0] value). Use a Combine variable and combine these attributes for each type of event.
11. Create a Lookup variable and pass the unique id (created in last step), to check for secondary router.
12. In the Advanced Filter section, check for secondary router's availability.
13. In the Message Key section, specify the unique router id (which is created in step 10) for each type of event. This will correlate the alarms from the same router together.
14. To discard threshold traps when the set is completed, check `Discard on Set Completion` box for Threshold Breach events .
15. Specify the Window Period of 30 seconds.

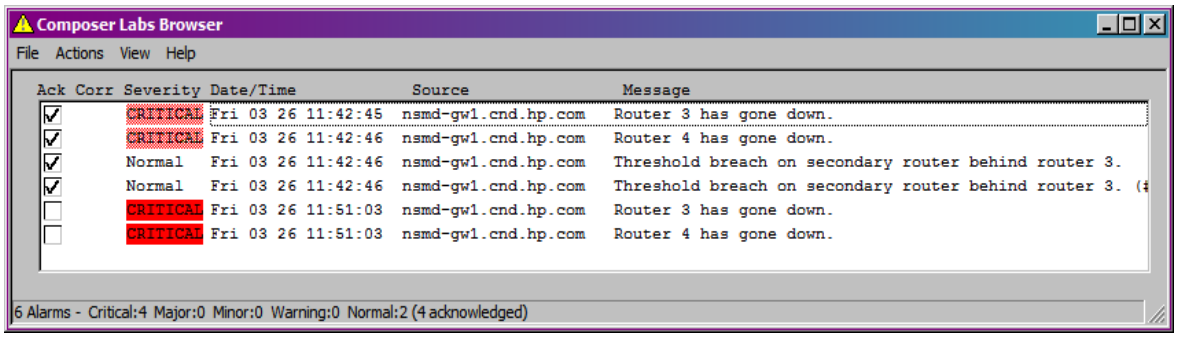
16. Save your Correlator Store in the `$OV_CONF/ecs/CIB` directory.
17. Exit the Composer developer interface.
18. Add your data to the datastore.
19. Provide Operator access to your correlator.
20. Deploy your correlation.

Test Your Correlator

21. Type the commands:

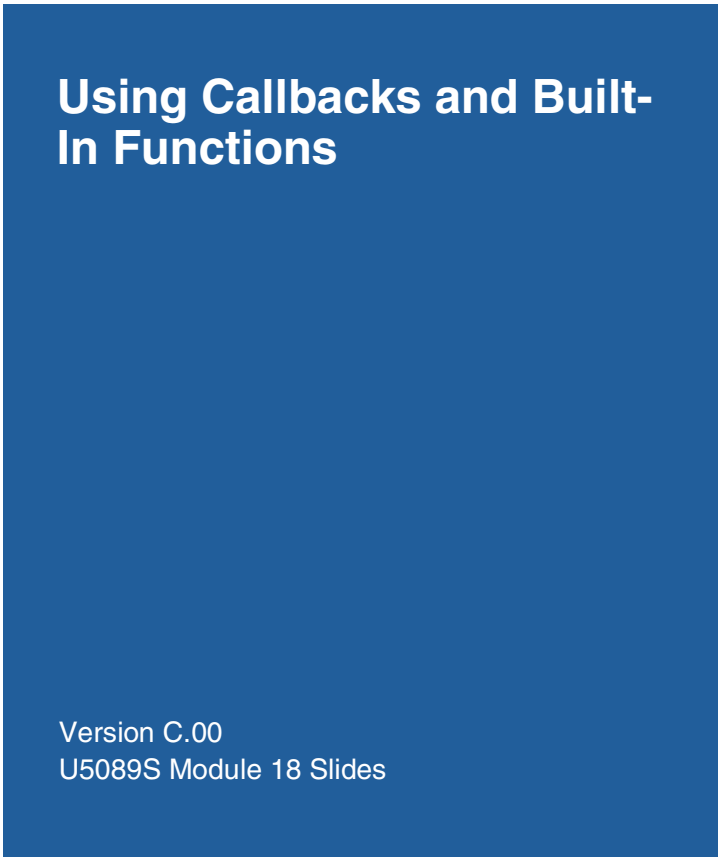
```
cd $OV_CONTRIB/ecs

ecsevgen -n secondaryRouter.evt
```

22. Examine the All Alarms Browser. You should see only the first two events come through. .



18 Using Callbacks and Built-In Functions

Module Objectives



At the completion of this module, you will be able to:

- Use a discard callback.
- Pass parameters to a function.
- Specify a Perl or C callback function.
- Access a built-in function.

Variables that Access Functions

Variables that Access Functions

- Gather external information for
  - Message Key
  - Advanced Filter
  - Count or store incoming event data
- Functions can be written in C, Perl or Composer's Built-In functions
- Parameters to the function:
  - Global Constants
  - Variables defined within the correlator
  - Attributes of the event

U5089S ©2004 Hewlett-Packard Development Company, L.P. 2

The variable type can be a function whose return value is bound to the name of the Variable.

To define the function:

1. Enter the Function Name. This is the name of the function to be called.
2. Provide a brief description of the function in the Description tab. To view the entire description window, click the [...] button.
3. Select the function type from the drop down menu. The function types are
   - C function
   - Perl function
   - Built-in function

   While Perl and C functions are external functions to be supplied by the user, built-in functions are pre-packaged with the Composer.

NOTE If you wish to use external calls, contact HP to purchase a Partner Care Extended support contract before beginning.

NOTE You can also specify the library name while defining the function. The function name can be prepended with the C library name, where this function resides.

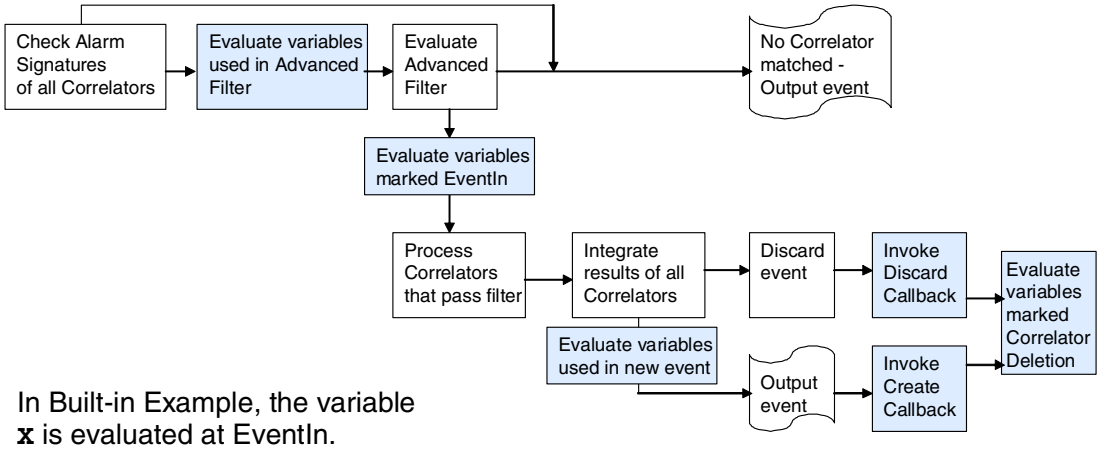
For example, if a function named BSCName() resides in a library called SNMPlib, you could mention function name as SNMPlib:BSCName() in the Function Name text box.

4. Select the time at which the external function has to be called. The function can be called at the:
   - Default - The external function (written in Perl or C) is called.
   - Event In - The function is called when each event enters ECS.
   - Correlator Creation - The function is called when the Correlator is instantiated (the first event enters).
   - Correlator Deletion - The function is called when the instance of the Correlator is deleted (the last event leaves).
5. Select the mode in which the function must be called, synchronously or asynchronously. This option is not valid for Composer Built-in functions. Built-In functions are always invoked synchronously. Asynchronous functions are executed by `genannosrv`. Synchronous functions are executed by `pmd`. All of them execute with `root` privilege. The following functions are automatically called synchronously:
   - Callbacks, both create and delete
   - Event In
   - Correlator Creation
   - Correlation Deletion
   - Built In
6. The parameters for the function must now be provided. Select the parameters from the Parameter list's pop up menu.
7. Click [OK] to complete the function definition.

Variable and Function Evaluation

Variable and Function Evaluation

- Variables are evaluated only when they are used.
- True of lookups, combines, extracts, and functions



In Built-in Example, the variable **`x`** is evaluated at EventIn. **`variableList`** is evaluated when used in the new event creation.

©2004 Hewlett-Packard Development Company, L.P.

Rules for variable evaluation

By default, a variable is evaluated (via a function) when it is used. To override this default behavior, select the point at which the variable should be evaluated. Following are the different times at which the variable can be evaluated:

- Default: The variable is evaluated when it is used.
- Event In: The variable is evaluated when each event participates in the Correlator after having passed the primary and secondary filter conditions.
- Correlator Creation: The variable is evaluated when the Correlator is instantiated.
- Correlator Deletion: The variable is evaluated when the instance of the Correlator is deleted.

Note that all parameters other than the standard attributes displayed in the pop up menu must be previously defined as variables.

NOTE A variable is evaluated ONLY once. For example, if a variable has been flagged to be evaluated at EventIn but the variable is used in the Advance Filter, then the variable gets evaluated when the Advanced Filter is processed and is not NOT

re-evaluated when the event enters the Correlator.

Functions flagged for evaluation at EventIn, Correlator Creation, and Correlator Deletion will ALWAYS be invoked synchronously.

An Asynchronous function whose parameters have not yet been evaluated at the point at which the function has been invoked cannot depend on a parameter that is evaluated through another Asynchronous function.

Callback Example - Problem Statement

- When a BSC fails, connectivity to all the BTSs connected to it are lost (MultiSource example).
- The requirement is
  - discard all BTS_FAIL traps when the parent BSC fails in a 10 second window
  - for every BTS trap discarded, invoke a Perl function someFunc() with two parameters
    - varBind[0] of the BSC trap
    - varBind[1] of the BTS trap being discarded

©2004 Hewlett-Packard Development Company, L.P.

In addition to the MultiSource example, what do you need to know?

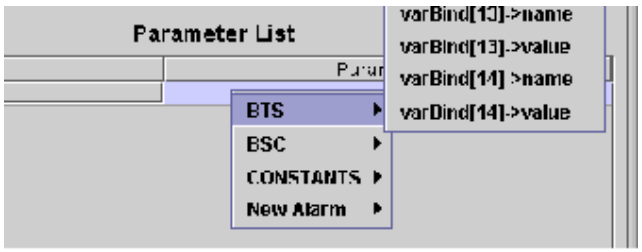
1. What do you do when a BTS failure is discarded?

   Call a function named `someFunc`, passing varbind[0] from the BSC trap and varbind[1] of the BTS trap.

Passing Parameters to Functions

Passing Parameters to Functions

- Attributes of different type of events can be selected from the list.
- The function is called only after the event is created or discarded.
- The new event's attributes can also be passed to a function, by selecting attributes from New Alarm-> <attributes list>



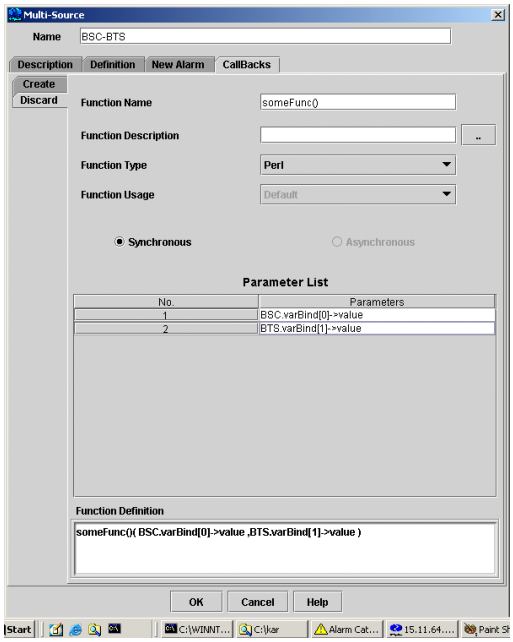
©2004 Hewlett-Packard Development Company, L.P.

Correlator Template	Available Attributes for Create Callback	Available Attributes for Discard Callback
Suppress	- All Attributes - Constants	- All Attributes - Constants
Enhance	- All Attributes - Constants - New Alarm attributes	Not Applicable
Rate	- All Attributes - Constants - New Alarm attributes	Not Applicable
Repeated	- All Attributes - Constants - New Alarm attributes - Suppressor events attributes	- All Attributes - Constants - New Alarm attributes - Suppressor event attributes

Transient	- CLEAR event Attributes - Constants - New Alarm attributes	- CLEAR event Attributes - Constants - Suppressor event attributes
Multi-Source	- Different set of event Attributes - Constants - New Alarm attributes	- Different set of event Attributes - Constants - Discarded events attributes
User Defined	- All Attributes - Constants - New Alarm attributes	- All Attributes - Constants - New Alarm attributes

Callback Example - Callback Specification

Callback Specification



- Callbacks - call a user specified function
  - Create Callback: when the new event is created. Applies within the scope of the correlator
  - Discard Callback: when the event is discarded by a correlator

U5089S ©2004 Hewlett-Packard Development Company, L.P. 6

When a Correlator either discards or creates a new event, the user can optionally choose to invoke a user-defined function, implemented in either 'C', PERL or the built-in functions. Parameters to these functions can be chosen from the set of variables defined for the Correlator.

Typically the callback functions are used to create audit trails. For example when an event is deleted, a logging function can be invoked.

Follow the procedure given below to configure a function call:

1. Click the Callbacks tab in the Correlator window. The Callbacks panel is displayed.
2. Enter the Function Name. This is the name of the Callback function provided by the user.
3. Enter the Function Description. Provide a description that will help you identify what the function does.
4. Select the Function Type from the drop down menu and the mode in which the function must be called.
5. Select the time at which the Function is to be called from the Function Usage drop down menu.
6. To select parameters to the external function, select the attributes from the pop up menu displayed in the Parameters table.
7. To add more parameters to the function, right click the mouse button and select the attributes

from the pop up menu.

8. Click [OK] to complete the Callback function definition.

Variables available to be used in Callbacks

All events have their attributes available via their corresponding event names. Automatic variables are available for access by the Create and Discard Callback functions. The Create Callback can access the attributes of the new event just created via the automatic variable NewAlarm. The Discard Callback function's automatic variables are dependent on the Correlation Model chosen. See the *HP OpenView Correlation Composer's Guide* for details.

NOTE The Create and Discard Callback functions can only be called synchronously.

TIP Only write functions that have side effects for the callback phase of Composer.

Callback Example Update - Problem Statement

Callback Example Update - Problem Statement

•What if the Operator wants access to the BTS traps?

- Correlate the discarded trap under the parent BSC alarm.
- Change Perl function **`someFunc()`** to the C function **`libOrchNNM:Orch_log_correlation.`**

•This function displays only the root cause in the Alarm Browser and all the discarded alarms in a correlated events window.

•This function requires two parameters:

- **`uuid`** of the root cause alarm (in this case, BSC is root cause alarm).
- discarded **`uuid`** (in this case, BTS alarms). The discarded alarms attributes are available by selecting: Discarded->attribute list.

U5089S ©2004 Hewlett-Packard Development Company, L.P. 7

When you create a new event, you may want to correlate all the input events under it in the Alarm Browser. This gives the operator access to the event data without cluttering the Alarm Browser. You can use the external function call `libOrchNNM:Orch_log_correlation`. This will be called for every discarded BTS trap and all of them will be correlated with the same parent.

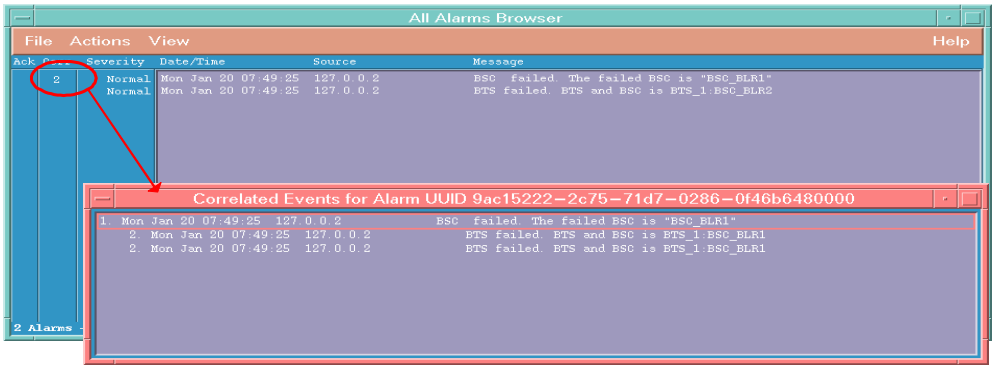
The function `libOrchNNM:Orch_log_correlation` requires two parameters. The first is the event uuid (universal identifier) for the event to be the root cause or parent alarm. The second is the uuid for the event to be correlated under it.

To see how to correlate multiple child events, refer to the OV_MultipleReboots correlator.

Callback Example Update- Display Correlated Events

Callback Example Update - Display Correlated Events

- The NNM Alarm Browser displays only the root cause alarm (BSC alarm) and the total number of discarded alarms in “corr” column.
- Select **`Actions:Show Correlated Events`** to show all the discarded alarms.



 ©2004 Hewlett-Packard Development Company, L.P.

Once the events are correlated, the Operator can review them by selecting the root cause alarm, then selecting `Actions:Show Correlated Events`.

Built-In Functions

Built-in Functions

•Composer is bundled with the following built-in functions to perform simple logging, storing, retrieving and manipulation of event data:

add	getCounter	mul
bitand	getHour	retrieve
bitinv	getMinute	retrieveStr
bitor	getMonth	setCounter
bitxor	getTime	store
div	makeList	storeStr
getByIndex	mod	sub

©2004 Hewlett-Packard Development Company, L.P.

The Composer comes bundled with built-in functions to perform simple logging, retrieving and manipulation of event data. The *HP OpenView Correlation Composer's Guide* documents all the built-in function syntax.

- `add` Returns the sum of values that are passed to it
- `bitand` The bitwise and operation on its arguments
- `bitinv` The bitwise inverse of the argument
- `bitor` True if either argument is true
- `bitxor` The bitwise exclusive-or of the two arguments
- `div` Integer divide
- `getByIndex` Returns from the specified element from the list
- `getCounter` Returns the value stored in a counter
- `getHour` Returns the current hour
- `getMinute` Returns the current minute
- `getMonth` Returns the current month
- `getTime` Returns the time(in seconds) since epoch

- `makeList` Returns a list that contains the set of arguments passed to it
- `mod` Returns the first integer modulus the second integer
- `mul` Return the product of values passed to it
- `retrieve` Retrieves a value stored previously
- `retrievstr` Retrieves a string stored previously
- `setCounter` Stores the incremented value
- `store` Stores a value based on a key
- `storeStr` Stores the string value based on a key
- `sub` Returns the difference of the values passed to it

Built-in Example - Problem Statement

Built-in Example - Problem Statement

•When a router is reinitialized a Start trap is emitted. The requirement is

- to discard the Start traps but monitor the rate at which they arrive.
- if more than 3 traps are emitted from the same router in 30 minutes, emit a new event indicating the instability of router.
- the new event created must contain the varbind[0].values of all traps that were discarded.

©2004 Hewlett-Packard Development Company, L.P.

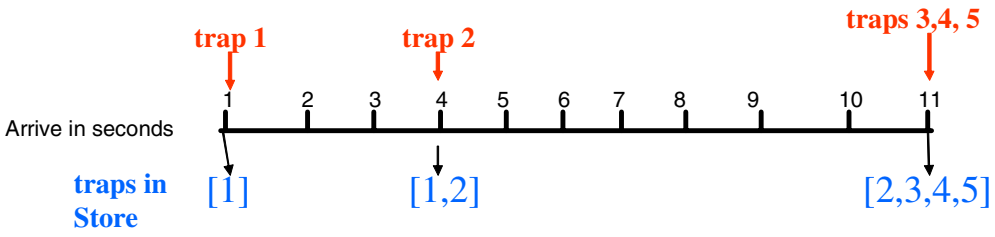
Built-in Example - Design

Built-in Example - Design

•Pass the varbind[0].value of all the traps that contributed to the breach to an external function when the breach occurs.

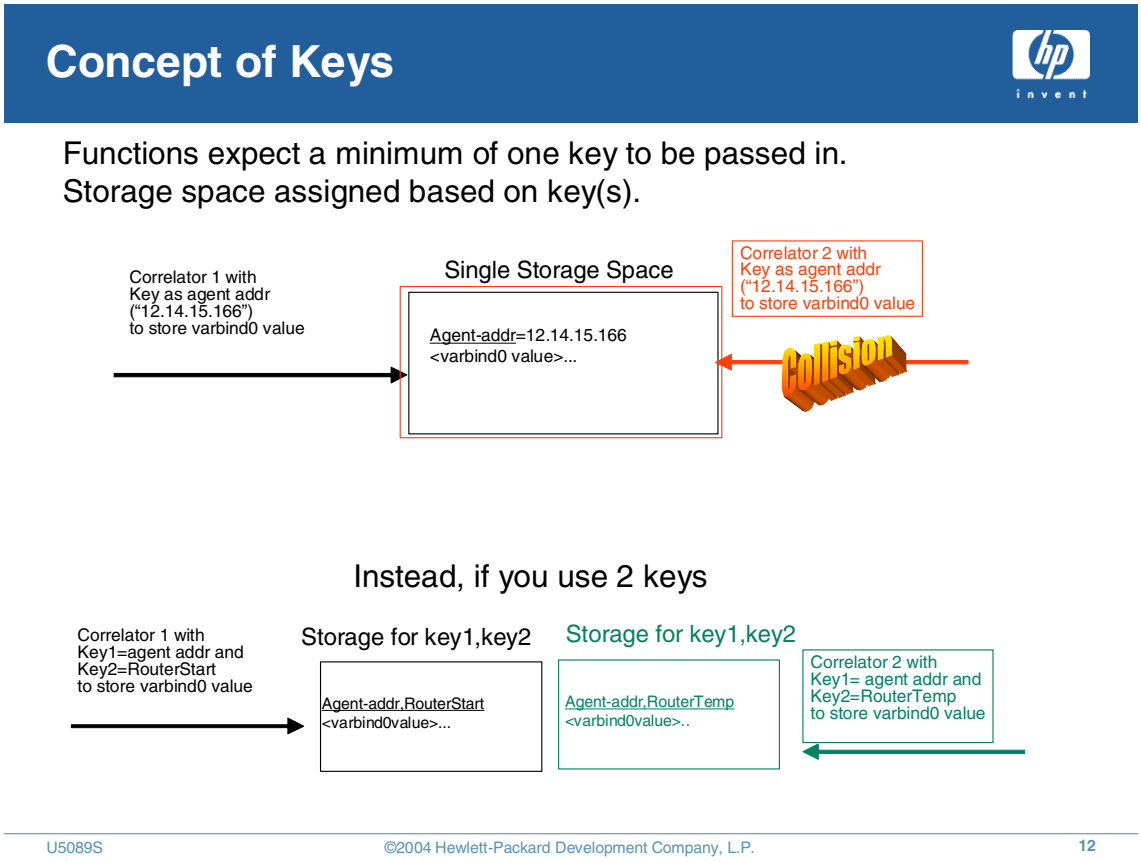
- However when these traps enter the correlator, it is unknown whether they contribute to the breach.
- Store the value for a fixed period of time equal to the window period specified, then it is guaranteed that when the breach occurs only attributes of the traps that contributed to the breach are in the store.

•When the window period is 10 seconds and trap count is 3 (breach value), the store mechanism stores the traps as shown:



©2004 Hewlett-Packard Development Company, L.P.

Concept of Keys



For all the store and retrieve functions (that is, `store`, `retrieve`, `storeStr`, `retrieveStr`) the value stored or retrieved is against the keys passed into the function as parameters. The functions expect a minimum of one key to be passed in. However multiple keys can also be used. When multiple keys are used, the function internally concatenates the values referred to by these keys and creates a single key.

For example, a key X which holds a value 'abc' is equivalent to the set of keys x, y, z where they hold values 'a', 'b', 'c' respectively. Note that the order of keys is very important. Taking the above example passing in keys z, y, x would result in a final key value of 'cba' and NOT 'abc'.

The `store` and `retrieve` functions use a global hash table. While this is a powerful mechanism of passing data between correlators, an incorrect usage would result in correlators overwriting each other's spaces. For example, if Correlator 1 stores a value against a key whose value is 'abc', and another correlator Correlator2, stores a value against a key(s) whose value also evaluates to 'abc', then the value stored would be the last value stored. To ensure that correlators do not step on each other keys should be chosen such that they are unique.

TIP A good way to ensure this is to use the Correlator Name as part of the key.

Note that `store` and `retrieve` use a different hash table than that of a `storeStr` and `retrieveStr`.

storeStr () Syntax

storeStr ( ) Syntax

•storeStr stores the string value based on the key(s) for a specified time period

`storeStr toAppend separator value window key1, key2`

- **`toAppend`**: parameter that decides how value is stored
- **`separator`**: field separator
- **`value`**: value to be stored
- **`window`**: time period for which value will be stored
- **`key1, key2`**...: keys based on which value is stored.

•In this case, the requirement is to store the concatenated string of all the discarded events' varBind[0] values. For this, create a variable to hold values and pass it as a parameter to the function.

©2004 Hewlett-Packard Development Company, L.P.


```
storestr toAppend seperator value window key1, key2,...
```

Where,

- `toAppend` parameter decides how the value will be stored
- `separator` is the field separator
- `value` is the value to be stored
- `window` is the time period for which the value will be stored
- `key1, key2...` are the keys based on which the value will be stored.

Description The `storeStr` function stores the value as a string based on the key(s) for a specified time period.

The parameter `toAppend` can take the following values

- `0` The value is appended to the existing value and is stored based on the keys
- `1` The value is stored based on the keys. Any values stored previously are erased and only the new value is stored.

The store and storeStr functions store their 'values' in the heap, so available storage is as large as malloc can go for the system.

retrieveStr () Syntax

retrieveStr ( ) Syntax

•retrieveStr function retrieves the value stored(via the storeStr function) based on the keys

`retrieveStr toInit failvalue key1, key2,…`

- **`toInit`**: is the method in which the value will be retrieved
- **`failvalue`**: is the value to be returned by the function if the **`retrieveStr`** function fails.
- **`key1, key2`**,…:are the keys based on which the value is retrieved

©2004 Hewlett-Packard Development Company, L.P.

`retrieveStr toInit failvalue key1, key2,...`

Where:

- `toInit` is the method in which the value will be retrieved
- `failvalue` is the value to be returned by the function if the retrieve function fails.
- `key1, key2,...` are the keys based on which the value is retrieved

Description The `retrieveStr` function retrieves the value (as a string) stored previously via the `storeStr` function based on the same set of keys.

Built-in Example - Design

Built-in Example - Design

- Declare a variable X that stores the return value of storeStr and the value stored is the attribute (varBind[0]) of the incoming event
- Select Correlator EventIn from the Function Usage drop down box
- To retrieve the stored value, use retrieveStr( ) function.

©2004 Hewlett-Packard Development Company, L.P.

The trap PDU looks like:

```
Trap-PDU{
enterprise {1 2 3 4 5},
agent-addr internet : "\x0A\x00\x01\x7F",
generic-trap 6,
specific-trap 8,
time-stamp 414746291,
variable-bindings{
}
}
```

What you need to know?

1. How do you identify a Start trap?

   The specific ID is 8.

2. How do you know which device it came from?

   The Message Key can be the agent-address.

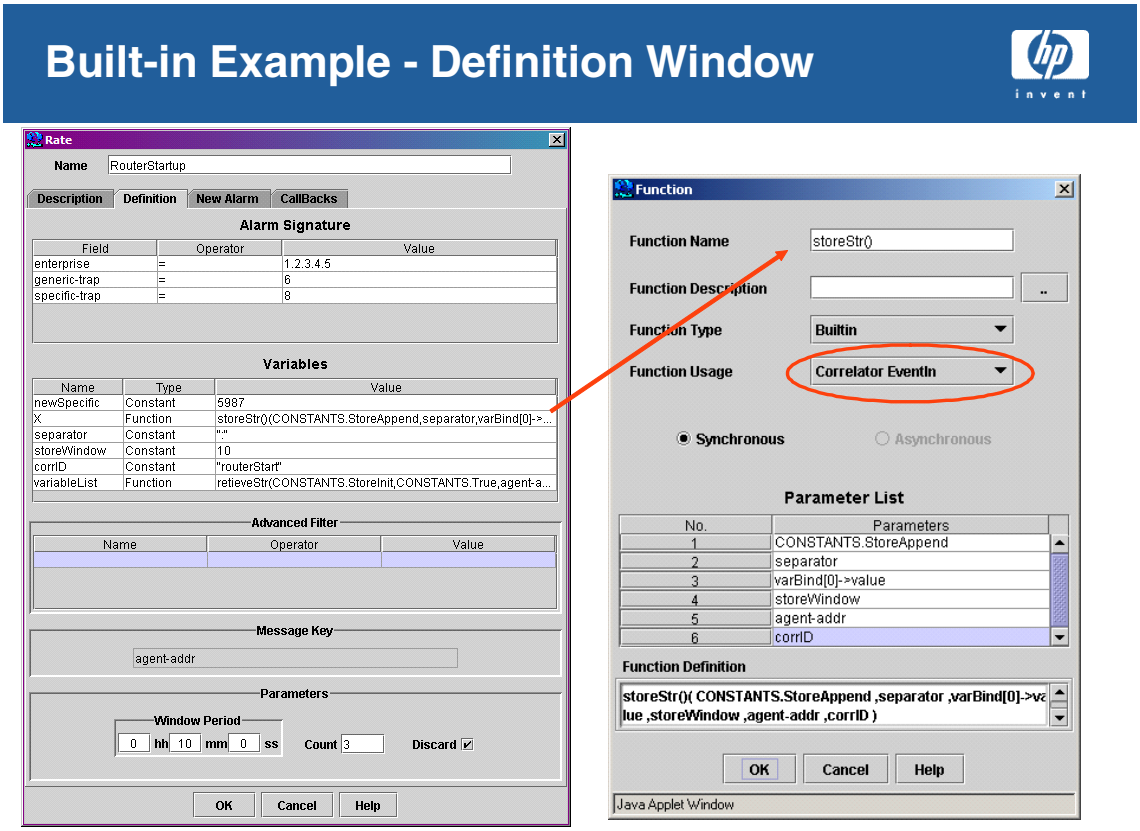
3. How long will you watch for traps?

   The Window period is 10 minutes.

4. How will you collect the `varbind[0].values` from the traps?

   Use the `storeStr` built-in function to collect them into a datastore. Have it evaluated as each event enters the correlator.

Built-in Example - Definition Window



 ©2004 Hewlett-Packard Development Company, L.P.

Create or verify the following Global Constants:

- `CONSTANTS.StoreAppend=0`
- `CONSTANTS.StoreInit=1`

  This causes the storage space to be re-initialized after the list is retrieved.

Create the following variables:

- `Separator = ":"`
- `newSpecific = 5987`
- `storeWindow=10`
- `corrID="RouterStart"`

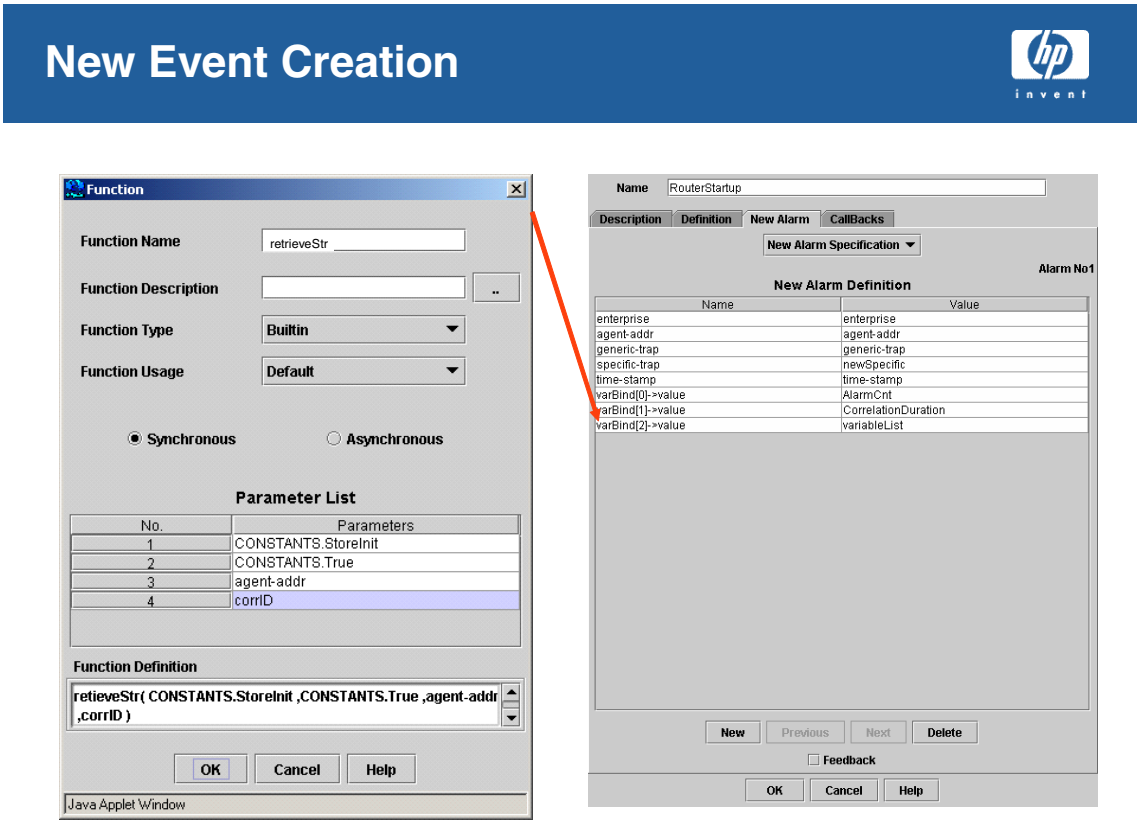
  This differentiates items stored under the same key from different correlators.
- Define the variable `X` that calls the `storeStr` function with these values in the proper parameter order. Have the function evaluated when an event comes into the correlator.
- Define the variable `variableList` to hold the returned list from `retrieveStr` (using the same keys). This will be used in the new event creation.

Use the `agent-address` as the Message Key.

Set the Window to 10 minutes and the count to 3.

Discard the traps as they arrive.

Built-in Example - New Event



Follow the procedure given below to create a new event.

1. Select New Alarm Specification from the drop down menu in the New Alarms panel.
2. The New Alarm Definition table displays all the mandatory attributes that must be filled in. Select values for the fields from the pop-up list displayed.
   a. Set the `enterprise`, `agent-addr`, and `generic-trap` to match the incoming event.
   b. Set the `specific-trap` to use the variable `newSpecific`.
   c. Set the `time-stamp` to match the incoming event.
   d. Add a row for `varbind[0]->value` and set it to `AlarmCnt`.
   e. Add a row for `varbind[1]->value` and set it to `CorrelationDuration`.
   f. Add a row for `varbind[2]->value` and set it to the variable `variableList` that contains the return of all the collected variables from the traps.
3. Do not mark the event for feedback.
4. Click [OK].

GetByIndex() to Access Multiple Return Values

getByIndex Built-in Function Syntax

•The getByIndex function returns the element at *index* position from the list passed in. If *index* number of elements do not exist, then the function returns the *failvalue*.

•getByIndex list index failvalue

•*list* is a list of any data type

•*index* is the position from which the value is to be extracted

•*failvalue* is the value returned if the getByIndex function fails

U5089S ©2004 Hewlett-Packard Development Company, L.P. 18

Syntax

```
getByIndex list index failvalue
```

where:

`list` is a list of any data types

`index` is the position from which the value is to be extracted

`failvalue` is the value returned if the getByIndex function fails

Description

The `getByIndex` function returns the element at index position from the list passed in. If `index` number of elements do not exist, then the function returns the `failvalue`.

Typically the `getByIndex` function is used to retrieve individual elements from the return value of the previous call to an external function.

Example

Let there be a external function called `getInterfaceDetails` which returns the `interfaceName` and interface IP Address and this return value bound to a variable called `details`. To extract the IP address, the `getByIndex` function will be called as

```
getByIndex details 2 0
```

If the `getByIndex` function fails, the value returned is 0.

getByIndex Example

getByIndex Example

- Problem Statement
  - When an event of Type A arrives, forward the original event and simultaneously emit a new event with
    - specific-trap set to 20
    - varbind[0] contains the set of services affected by the failure
    - varbind[1] contains the region affected by the failure
    - varbind[2] contains the customers affected by the failure
- Solution Design
  - Bind variables to two external functions:
    - X=getAffectedRegion - given the IP address of the failed entity, return the Region affected (as a string).
    - Y=getAffectedServiceCustomers - given the IP address of the failed entity and affected region, returns 2 values

      set of services affected

      set of customers affected
  - Extract the first element from Y and bind it to the variable Services using the Built-in function getByIndex. Bind the second element to the variable Customers.

 ©2004 Hewlett-Packard Development Company, L.P.

Type A trap PDU looks like

```
Trap-PDU{
enterprise {1 2 3 4 },
agent-addr internet : "\x0A\x00\x01\x7F",
generic-trap 6,
specific-trap 21,
time-stamp 414746291,
variable-bindings{
{
name {1 2 3 4 0 1},
value simple : number : 40
}
}
}
```

What do you need to know before you begin to configure this correlator?

1. Since the requirement is to create a new alarm for each incoming alarm, use the Enhance

Correlator.

2. Create an Alarm Signature for Type A events.
3. Call the function `getAffectedRegion` and bind it to variable `X`. Then call the function `getAffectedServiceCustomers` and bind to variable `Y`.
4. The function `getAffectedServiceCustomers` returns two values into `Y`, which is a list that has two elements. Extract the first element from `Y` and bind to a variable, `Services`, and do the same for the second element in the list, binding it to `Customers`.

   The mechanism used to extract a value from a list is by using Built-in function `getByIndex`.
5. Specify the new event to be generated.

Example: OV_MultipleReboots

Example: OV_MultipleReboots

•When the trap passes the Alarm Signature (EventIn), storeUUID and storeCount gather information.

•An internal call to Orch_topoAddrToTopoInfo returns all device information given an agent-address.

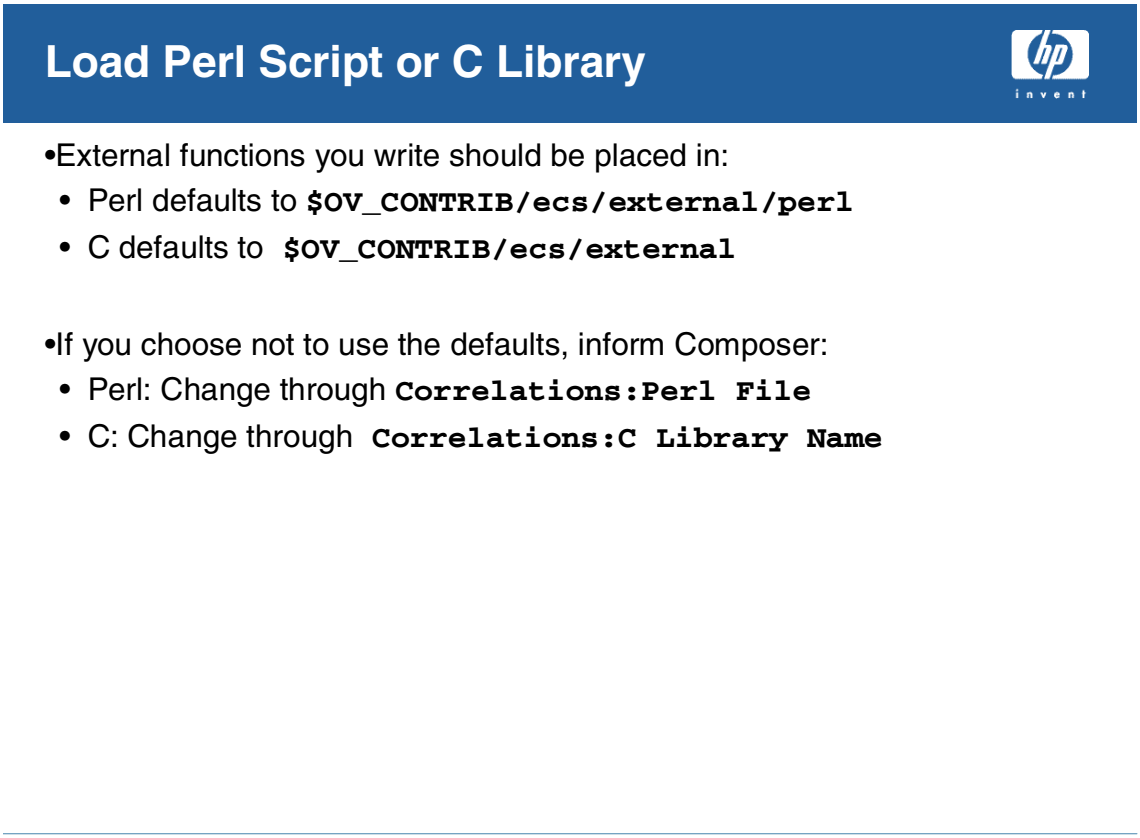
- getByIndex parses this into other variables
- When a new event is created, the parsed information goes into varbinds 1 and 2.

•The callback internal function Orch_log_correlations correlates the duplicate events under the most recent.

- NewAlarm.uuid is the most recent
- clearUUID gives the list of duplicate events (causes retrieveStr to be evaluated and reinitialize the storage space in the process)
- clearCnt tells ECS how many are correlated

©2004 Hewlett-Packard Development Company, L.P.

Load Perl Script or C Library



External functions can be written in Perl or C. The Composer must be supplied with the names of the files where these functions reside.

To provide the name of the Perl script file to the Composer:

1. Select `Correlations:Perl File` from the Correlator Store window. The Perl File window is displayed.
2. Enter the name of the Perl script file. The default path for the Perl file containing the external Perl functions is `$OV_CONTRIB/ecs/external/perl`. To pick up Perl files from a different location, the relative path must be specified in the Perl File window.
3. Click [OK] to close the window.

To set the default C library:

1. Select `Correlations:C Library Name` from the Correlator Store window. The C Library Name window is displayed.
2. Enter the name of the C library that contains the external function.

   The default library for the C function is placed under `$OV_CONTRIB/ecs/external`.

   You can also specify the library name while defining the function. The function name can be prepended with the C library name, where this function resides. For example, if a function named `BSCName()` resides in a library called `SNMPlib`, you could mention function name as

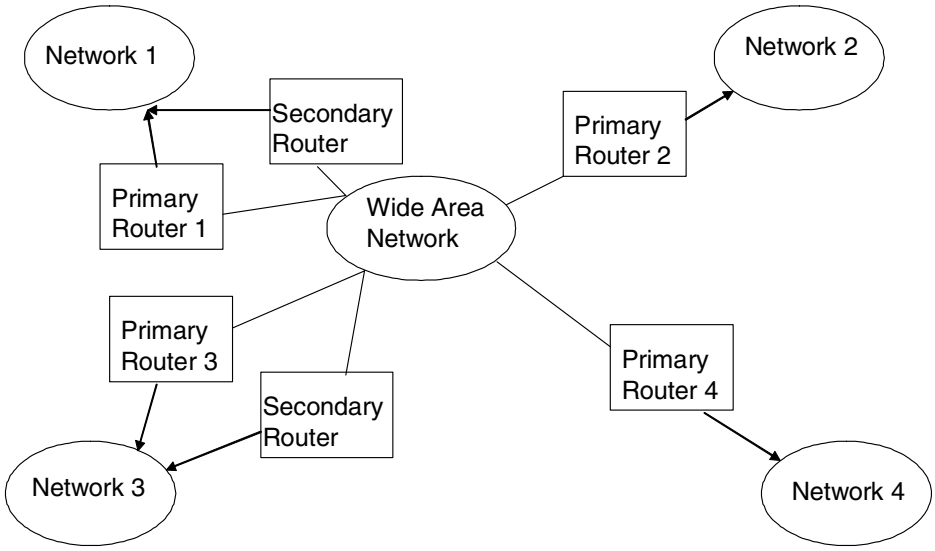
`SNMPlib:BSCName()` in the Function Name text box.

Note that multiple libraries can be loaded in this way. The library name entered in the C Library Name window is the default library, while the library name specified while defining the function can be a completely different library.

3. Click [OK] to close the window.

Lab Exercises

Lab Exercises



 ©2004 Hewlett-Packard Development Company, L.P.

Lab requires:

- Multi Source Correlator
- Discard Callback
- Built-in Functions storeStr, retrieveStr
- Create Callback
- Feedback

Requirements

NOTE This lab builds on the lab for MultiSource. If you have already done that lab, you may use it as the basis for this lab.

Assume a wide area network with many small networks interconnected by routers (primary and

secondary routers to share the load). Some sub-networks are connected by one router (primary) as shown in the slide.

If primary router is down and if a secondary router is connected, then all the traffic is diverted through secondary router. Due to heavy traffic, the secondary router will generate many “Traffic-Threshold-Breached” traps. The requirements are:

- When the Primary Router down trap is emitted, discard all the “Traffic-Threshold-Breached” traps from the secondary router for 30 seconds.
- Output the original Primary Router Down trap.
- Make sure that the trap sent is from the same primary and secondary router connecting the same 2 networks.
- The other requirement is to concatenate and store all the varBind[0] values of the discarded traps and create a new event after the window period. The new event should contain concatenated string of all the varBind[0] values of the discarded traps.

The router is uniquely identified by a combination of enterprise ID + Router ID (varBind[0] value). If the secondary router is connected, then only from those routers, the threshold alarms should be discarded. Whether the secondary router is available or not is stored in the datastore. Passing the unique router as a key to the datastore can identify it.

The data store looks like:

```
ADD DATA ("<enterpriseid + Routerid>", "<Yes/No>" )
```

Trap Definitions

A router down trap is identified by

- enterprise id set to 1.5.6.7.8
- specific-trap set to 8
- var-bind0 contains router id

Example Trap-PDU

```
Trap-PDU {
    enterprise {1 5 6 7 8},
    agent-addr internet : "\x7f\x00\x00\x02",
    specific-trap 8,
    time-stamp 414746291,
    variable-bindings {
{
    name {1 3 6 1 4},
    value simple : number : 10
}
}
}
```

A Threshold breached trap is identified by

- enterprise id set to 1.5.6.7.8
- specific-trap set to 9

- var-bind0 contains router id of the primary router it is backing up

Example Trap-PDU

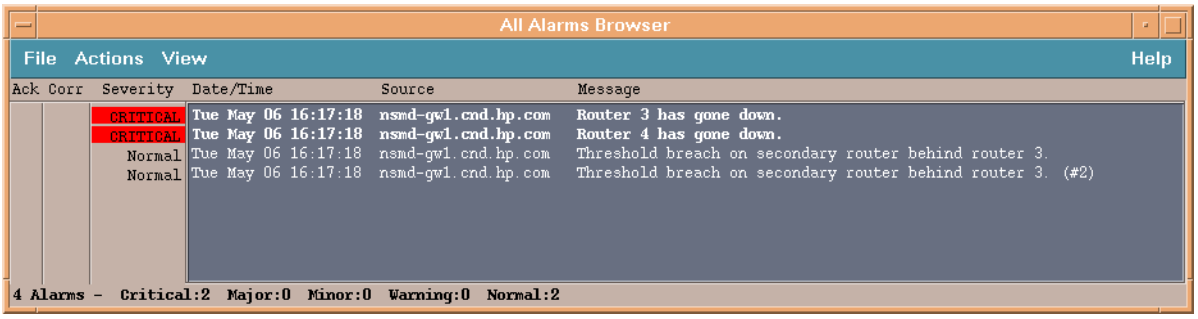
```
Trap-PDU {
    enterprise {1 5 6 7 8},
    agent-addr internet : "\x7f\x00\x00\x03",
    specific-trap 9,
    time-stamp 414746291,
    variable-bindings {
{
    name {1 3 6 1 4},
    value simple : number : 10
}
}
}
```

View Pre-Correlator Results

1. A sample event stream from this environment is available for testing. Copy the following file from `$OV_CONTRIB/OVTraining/NNM3` to `$OV_CONTRIB/ecs`:
   - `secondaryRouter.evt`: sends the following events:
     - — Router down from a router that has a secondary.
     - — Router down from a router that does not have a secondary.
     - — Threshold breach from secondary router (2 times).
2. Open the Composer Labs Alarms Browser.
3. Empty all old alarms by selecting `Actions:Delete->All Alarms in Category`.
4. Type the commands:

   ```
   cd $OV_CONTRIB/ecs
   ecsevgen -n secondaryRouter.evt
   ```

5. Review the alarms in the Alarm Browser.

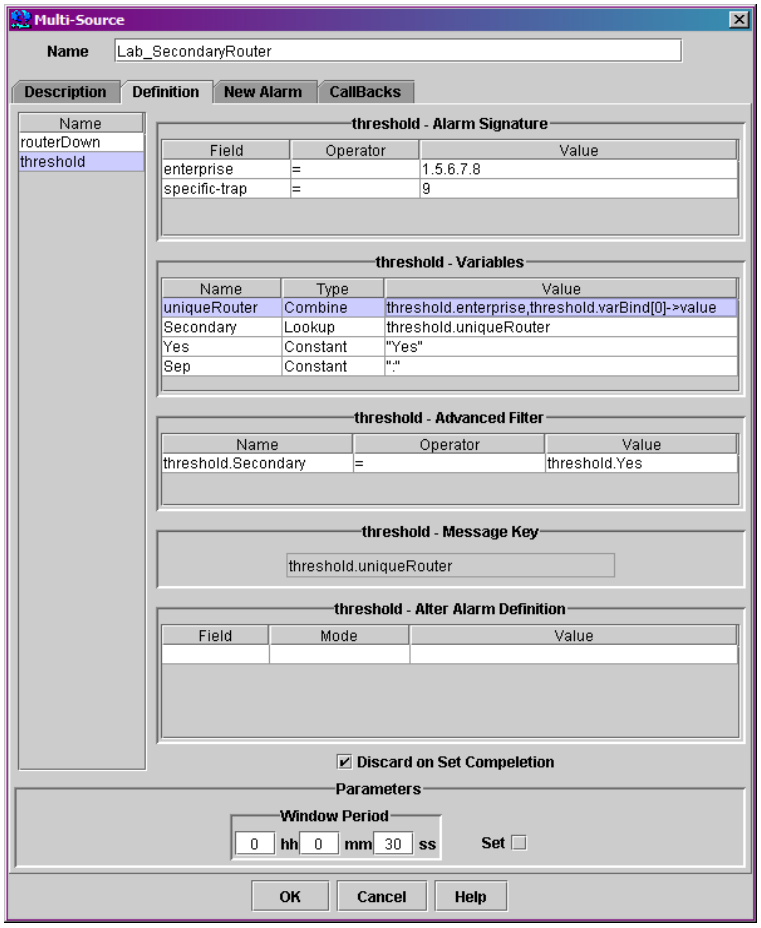


6. To mark which alarms came from this pre-test run, select `Actions:Acknowledge->All Alarms in Category`.

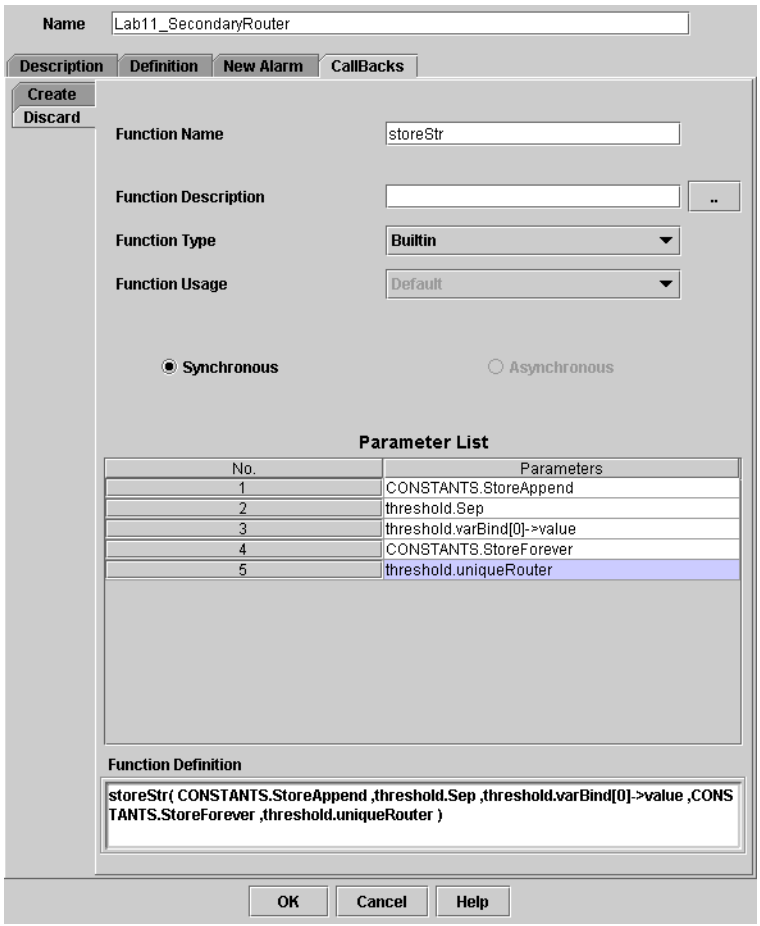
Directions

Execute the following steps to solve the exercise:

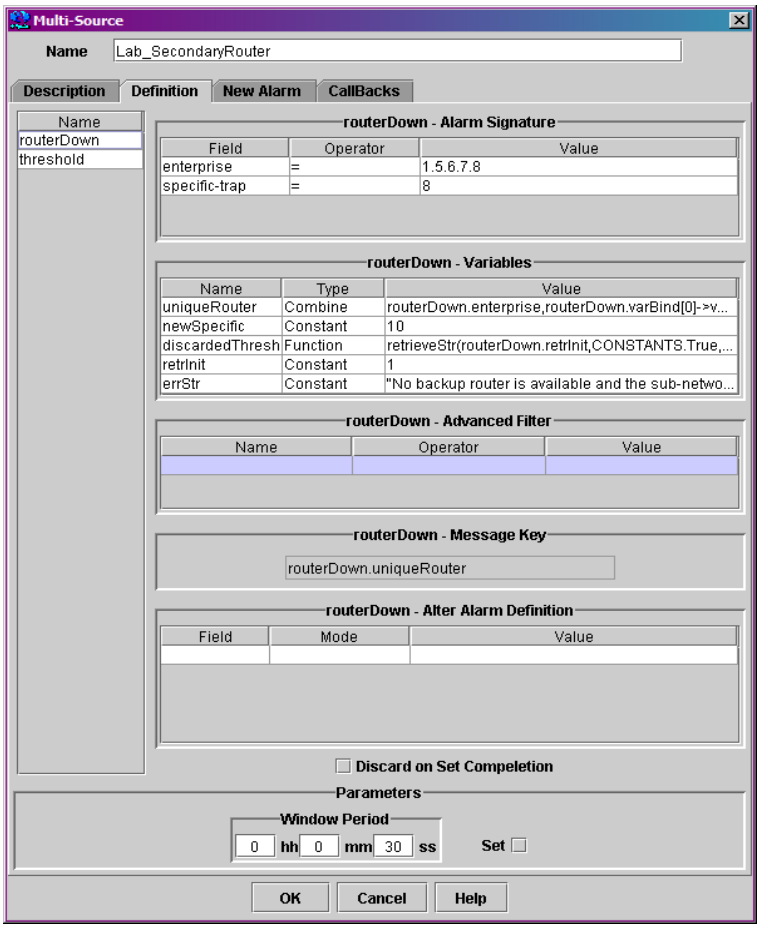
7. Use the Multi-Source Correaltor Template.
8. Create two types of events to filter:
   a. Router Down traps
   b. Threshold Breach traps
9. Specify the Alarm Signature for each type of event.
10. A router is uniquely identified by the combination of enterprise ID + Router ID (varBind[0] value). Use a Combine variable and combine these attributes for each type of event.
11. Create a Lookup variable and pass the unique id (created in last step), to check for secondary router.
12. In the Advanced Filter section, check for secondary router's availability.
13. In the Message Key section, specify the unique router id (which is created in step 10) for each type of event. This will correlate the alarms from the same router together.
14. To discard threshold traps when the set is completed, check `Discard on Set Completion` box for Threshold Breach events .
15. Specify the Window Period to 30 seconds.

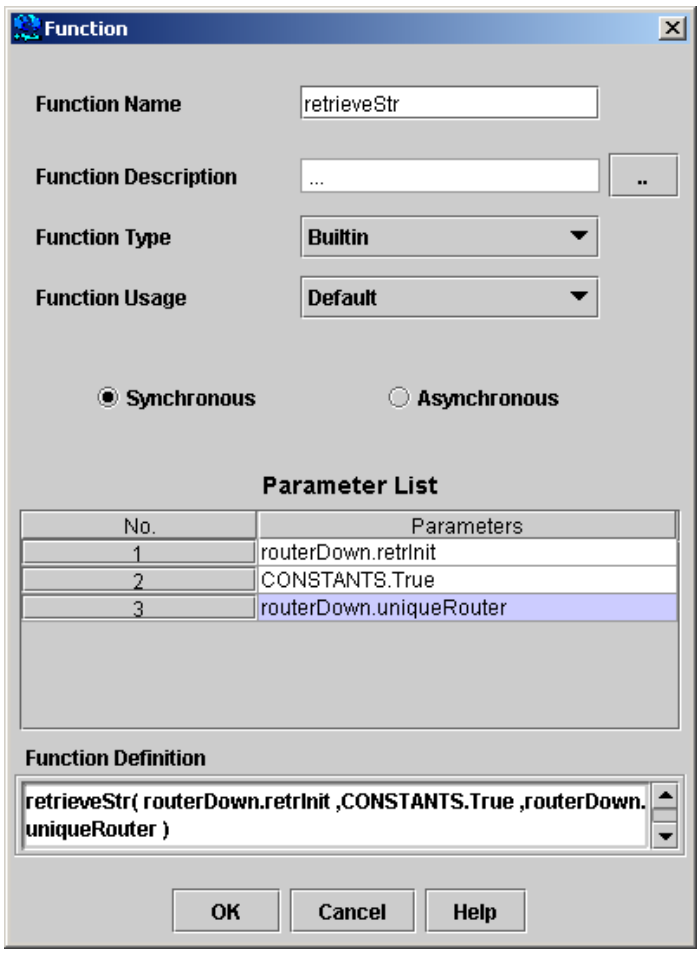


16. To concatenate all the varBind [0] values of the discarded traps, use Discard Callbacks section and call the Built-in function `storeStr` and pass the varBind[0] value. You will need constants defined (either Globally or within the variables table) to instruct the built-in to append your value to the storage space and to store it forever. Review the manual online for the values to use.

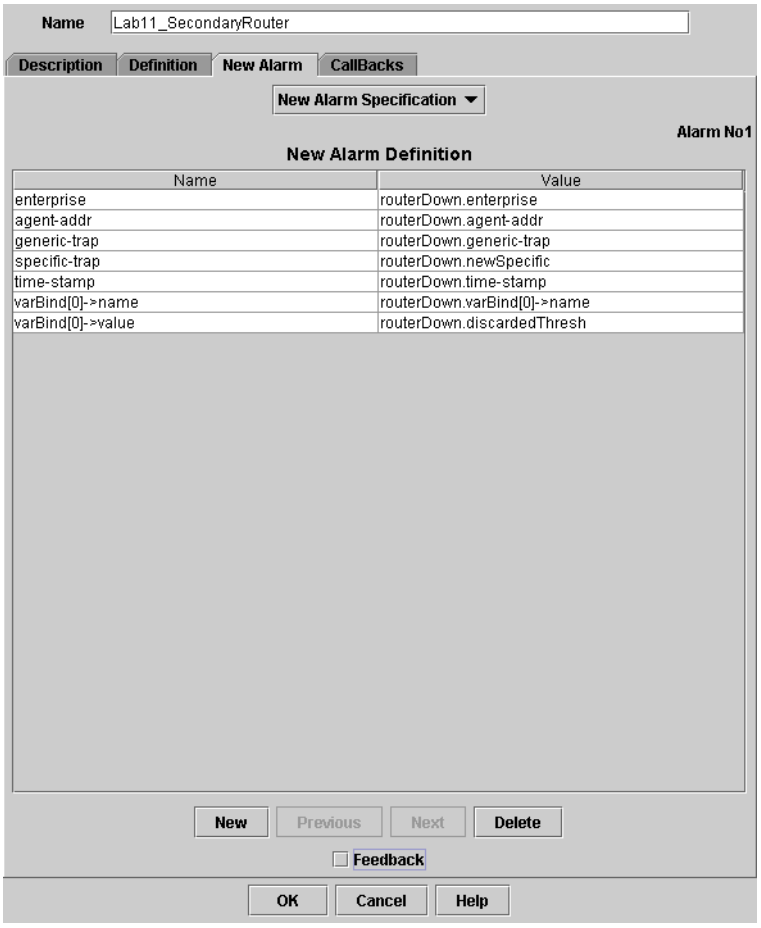


17. To retrieve the concatenated string of all the varBind [0] values of the discarded traps, use function variable type and call the Built-in function `retrieveStr` by passing the same key values as specified for `storeStr` function.You will need a constant value of your choosing for the function to return if the data is not available for your key.




Function
Function Name
retrieveStr
Function Description
...
..
Function Type
Builtin
Function Usage
Default
Synchronous
Asynchronous
Parameter List
No.
Parameters
1
routerDown.retrInit
2
CONSTANTS.True
3
routerDown.uniqueRouter
Function Definition
retrieveStr(routerDown.retrInit ,CONSTANTS.True ,routerDown.
uniqueRouter)
OK
Cancel
Help

18. Create New Alarm and bind the `retrieveStr` variable to any of the attributes to display all strings of all the varBind [0] values.



19. Save your Correlator Store in the `$OV_CONF/ecs/CIB` directory.
20. Exit the Composer developer interface.
21. Add your data to the datastore.
22. Provide Operator access to your correlator.
23. Deploy your correlation.

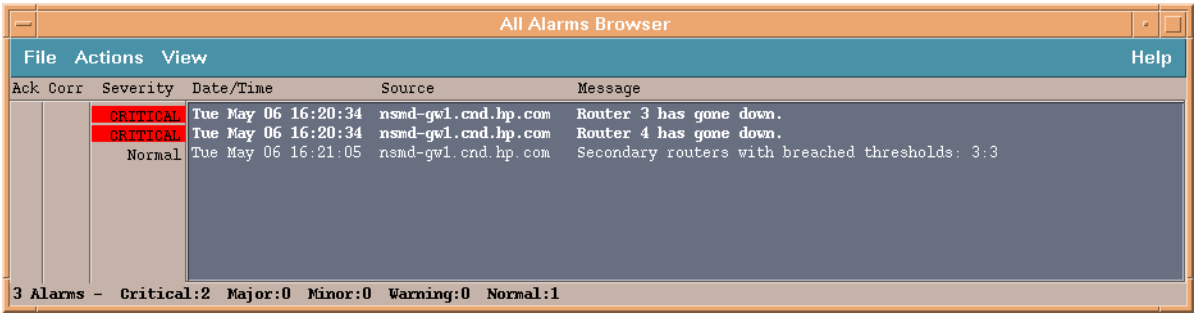
Test Your Correlator

24. Type the commands:

```
cd $OV_CONTRIB/ecs

ecsevgen -n secondaryRouter.evt
```

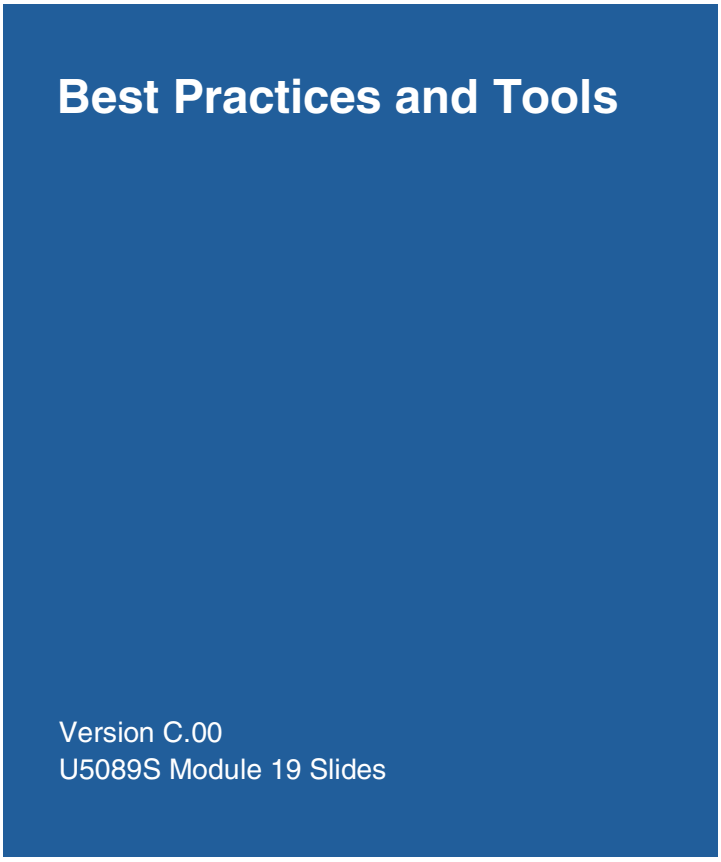
25. Examine the All Alarms Browser. You should see the first two events come through. After the timeout, you will see a new event with the router IDs from the two suppressed threshold

traps.


All Alarms Browser
File
Actions
View
Help
Ack Corr
Severity
Date/Time
Source
Message
CRITICAL
Tue May 06 16:20:34
nsmd-gw1.cnd.hp.com
Router 3 has gone down.
CRITICAL
Tue May 06 16:20:34
nsmd-gw1.cnd.hp.com
Router 4 has gone down.
Normal
Tue May 06 16:21:05
nsmd-gw1.cnd.hp.com
Secondary routers with breached thresholds: 3:3
3 Alarms - Critical:2 Major:0 Minor:0 Warning:0 Normal:1

19 Best Practices and Tools

Module Objectives



At the completion of this module, you will be able to:

- Analyze the Binary Event Store in an environment to determine candidates for event reduction.
- Capture a stream of events for analysis.
- Replay a stream of events for testing.
- Trace operation of your developed correlator.
- Debug correlators.

Best Practices

- Always develop on a test (non-production) station.
- Compare events with existing correlators.
- Test in isolation.
- Test coexistence.
- Test performance.
- Keep track of versions of your Correlator Store.
- Merge working correlators into your production Correlator Store.

 ©2004 Hewlett-Packard Development Company, L.P.

Recommended Procedures for Creating New Composer correlators

The following steps serve as general guidelines for developing any new correlator.

1. Do correlator development and test on a test system.

   To avoid breaking or impairing a deployment, new correlators should be developed and tested on a designated test system; a system that is not in use for active network management. Failure modes such as aborting the `pmd` process or significantly slowing down the event subsystem make this imperative.

2. Verify there are no clashes with existing correlators.

   Review the table in Appendix A of the *Developing_NNM_Event_Reduction.pdf* whitepaper to verify the new correlator will not interfere with any existing correlator, either by having the same input events or releasing any new event that may feed into an existing correlator.

3. Test in isolation first to validate functionality.

   Disable all other rules and correlations and test the functionality of the new correlator by sending the appropriate input events to the new correlator. See the <<ecsevgen.exe>> documentation for doing this. Validate the results of the correlator by using the browser. If the expected results are not being returned then you may need to turn on tracing. See the section on troubleshooting for tracing ECS. A good practice to follow if the new correlator has external functions or perl scripts is to put some tracing capability in the functions and scripts. This

allows the developer to trace the progress of the new correlator without having to get too involved with the ECS tracing.

4. Test coexistence.

   Verify the new correlator will still function properly with the product correlators enabled. If there are coexistence problems then disable the product correlators one at a time to isolate the failure. Once isolated, careful inspection of the rules along with ECS tracing will most likely be required to understand the problem.

5. Test performance.

   Verify the new correlator does not seriously impact the behavior of the systems ability to handle a storm of events while the new correlator is enabled. There are various ways to do this but repeatedly doing the following is a commonly practiced way to simulate a storm:

   ```
   ovtopofix -S down
   sleep 120
   ovtopofix -S up
   ```

   This should be done with all product correlators enabled and only on a test system.

6. Version all working copies of the `Composer.fs` to avoid losing work.

   Once the new correlator is developed and tested then save a copy of the test systems Composer fact store for versioning (`$OV_CONF/ecs/correlations/Composer.fs`). The only backup copy provided by the system is under `$OV_NEW_CONF/OVEVENT-MIN/ecscorrelations/Composer.fs`. This backup copy contains just the product correlators.

7. Merge (`csmerge`) the new correlators with NNM product `Composer.fs`.

   If the new correlator was developed on top of the product `Composer.fs` then merging is not necessary. If new correlators are developed separately then they will need to be merged together to have a single fact store. The merge tool `csmerge` should be used when combing the rules of different fact stores.

Migrating a Correlator Store File

Migrating a Correlator Store

•Files created with previous versions of **ovcomposer** need to be migrated manually.

csmigrate.ovpl *<CorrStorename>* -lang<ENGLISH>
-o *<final Correlator Store name>*

U5089S ©2004 Hewlett-Packard Development Company, L.P. 3

Migrate existing Correlator Stores

Correlator Stores created using Composer prior to Version 3.3 must be migrated to the latest version. The csmigrate script residing in the directory

UNIX: `$OV_CONTRIB/ecs`

Windows: `%OV_CONTRIB%\ecs`

migrates Correlator Stores created prior to version 3.3 to the latest version. To migrate to the latest version, type:

`csmigrate.ovpl <Correlator Store name> -lang<ENGLISH|JAPANESE|CHINESE> -o <final Correlator Store name>`

where,

<Correlator Store name> is the name of the Correlator Store that must be migrated.

<ENGLISH|JAPANESE> is the native language of the Correlator Store that must be migrated.

<final Correlator Store name> is the name of the Correlator Store after migration.

Viewing Previous Correlator Store Revisions

View Previous Correlator Store

- Recover from a disaster
- Recover from developmental errors

- Select **`Options:View Backup`**.

- Composer maintains 3 previous versions of correlator store file.
- Most recent is ***`filename`*`.1`**, then ***`filename`*`.2`**, then ***`filename`*`.3`**.
- Configure number of back versions in **`CO.conf`**.

©2004 Hewlett-Packard Development Company, L.P.

The Composer provides the ability to recover from a disaster by taking regular backups of the Correlator Store files.

The Correlator Store file, when created and saved the first time or opened the first time, creates a default (if it does not exist already) backup file which remains constant throughout the life of the Correlator Store. Changes made to the Correlator Store file and when saved the first time in the current session (involves the time between opening of the file and closure of file) will form the contents of a backup file.

The backed up file is identified by the extension '.1' appended to the filename at the time of Save. Consecutive first saves in future sessions result in the creation of renewed backed up files identified with extensions .2, .3 and so on. Backed up files roll from .1:.2:.3 and so on.

View Backup Files

To open a backed up file, select `Options:View Backup:<`*`Version of file`*`>`. Selecting this option displays the following message:

```
You are now viewing an archive version of the file. To restore this version,
select the Save button.
```

This message warns the user of viewing an older (backed up) version of the file. If you want to make this backed file the latest file, save the file to revert changes to the latest file. When you revert to the latest file, consecutive backed up files roll down to accommodate new changes.

IMPORTANT It is recommended that the decision to revert changes to the latest file be made judiciously, since there is always a threat that data can be overwritten erroneously.

Merging Correlator Store Files

Merging Correlator Store Files

•The **csdeploy.ovpl** script automatically merges files referenced in the NameSpace file prior to deployment.

•If you need to manually merge files:

```
csmerge <infile1> <infile2> <final Correlator Store>
      -config <configuration filename>
```

•Configuration file determines what to do in the case of clashes between the two input files.

©2004 Hewlett-Packard Development Company, L.P.

Correlator stores are merged automatically during an operator's deploy action into the runtime Composer.fs file. You may develop your correlators in separate files and store them that way. You do not need to merge your files.

Merging Correlator Store files

The `csmerge` tool is used to merge two Correlator Stores. Within a Correlator Store, no two Global Constants or no two Correlators can have the same name. Having two Global Constants or Correlators with the same name but with different values or Correlation logic is called a clash. The merge will be automatic if there is no clash in names.

However, if there is a clash, then external input is required to continue with the merge. External input is provided either interactively or by specifying it in the Configuration file.

The tool is available under

- `$OV_CONTRIB/ecs` for HP-UX and Solaris
- `%OV_CONTRIB%\ecs` for Windows

NOTE The tool is implemented as a Perl script and requires a minimum revision of Perl

5.6.

The csmerge tool recognizes the following options:

```
csmerge -namespace NameSpace.conf <final Correlator Store name>
```

```
csmerge -rm_desc < Correlator Store name> <final Correlator Store name>
```

```
csmerge <file1> <file2> <final Correlator Store name> -config <configuration filename>
```

The csmerge -h command summarizes the usage of csmerge. You can give only one command at a time. The csmerge command ignores all commands except the first.

Merge Correlator Stores that are specified in the NameSpace

Correlators Stores listed in the NameSpace can merged by specifying the name of the NameSpace file. All Correlators from the Correlator Stores are prefixed with the Logical name (as mentioned in the NameSpace file) of the Correlator Store as < Logical Name>_< Correlator Name> in the final Correlator Store. In the event there is an overlap of names of Global Constants, the Global Constants are also prefixed with the Logical Name of the Correlator Store. Hence it is important that Logical Names for Correlators be unique.

When csmerge is invoked with the -namespace option, all Correlator Stores are locked to enable merging. If the locking fails even for one of the Correlator Stores, then the merge process fails.

To merge the Correlator Stores that are listed in the NameSpace file, type:

```
csmerge -namespace <NameSpace filename> <final Correlator Store name>
```

where

- *<NameSpace filename>* is the name of the NameSpace file from which the Correlator Store files will be picked.
- *<final Correlator Store name>* is the name of the merged Correlator Store.

Remove User Description from Correlator Store

To remove the user description from a Correlator Store file, type

```
csmerge -rm_desc <Correlator Store name> <destination Correlator Store name>
```

where

- *<Correlator Store name>* is the name of the Correlator Store from which the user description is to be removed.
- *<destination Correlator Store name>* is the name of the Correlator Store without the user description.

Merge Correlator Stores

To merge two Correlator Stores, type

```
csmerge <file1> <file2> <mergedfile> -config <configuration filename>
```

where

- *<file1>* and *<file2>* are the Correlator Stores to be merged.
- *<mergedfile>* is the resultant file after merger.
- *<configuration filename>* is the file, which if present, specifies which values will be considered while merging the Correlator Stores. When this option is specified, the user in NOT prompted for input and all specifications is picked from the Configuration file. Additional information is available in the `csmerge` man page (reference page on Windows).

For a given clash, one of following can happen:

- the definition is picked from File1.
- the definition is picked from File2.
- both definitions are picked, but the name for one of them needs to change.

In the interactive mode (where is there is no Configuration file) you are prompted for input to decide which of the above needs to happen. In the non-interactive mode, the Configuration file is used to resolve any clashes.

Analyzing Events

- If your customer requirements are not complete
  - ovdumpevents
  - processEvents
  - processCorrEvents
- What events are really arriving
- How much reduction could be achieved
- Get an accurate baseline

 ©2004 Hewlett-Packard Development Company, L.P.

Before investing any effort in developing a correlation it is extremely important to get an accurate 'big picture' view of the events being processed by the NNM management system.

Before developing a correlator, obtain a snapshot of event dumps from the management system and analyze the event dumps (awk, grep) for reduction candidates. This sampling and analysis gives a perspective of the events coming into the system as well as some idea of how much reduction may be achieved.

To help in the analysis of events two scripts were developed (`processEvents` and `processCorrEvents`). These scripts are delivered with the product and are in the '`support`' directory. The procedure for analyzing events is as follows.

Dumping the Event Database

The command `$OV_BIN/ovdumpevents` produces an ascii output of the binary event store (BES) and the correlation log. The command options to do this respectively are:

```
ovdumpevents –s "default" > eventStoreDump
ovdumpevents –c "default" > correlationLogDump
```

The following is an example of the ascii format of an event from the BES:

```
1043024030 1 Sun Jan 19 17:53:50 2003 4kfcc5lc5m01.cnd.hp.com   N If J6 status Critical (was
Normal) station netmgt7.atl.hp.com;1 17.1.0.40000073 5499064
```

The ascii format includes a time stamp, agent address (hostname), event formatted string, severity (displayed as an integer), the trap OID and the specific ID.

The following is an example of the ascii format of a correlation entry:

```
Parent eventId = 03af0ca6-d22c-71d6-11f2-0f2c68020000
Child  eventId = 03aeee6a-d22c-71d6-11f2-0f2c68020000
Relationship = ddup
1043028357 5 Sun Jan 19 19:05:57 2003 atlgwb04.americas.hp.net  N Duplicate IP address: node
atlgwb04.americas.hp.net reported having 15.20.17.1, but this address was previously detected
on node atlhgw2.cns.hp.com;4 17.1.0.58982415 264
```

The first line is the parent (suppressor) event ID, the second line is the child event ID, the third line is type of correlation (ddup/ovin) that distinguishes de-duplication from correlation, and the fourth line is the event data of the child event.

By getting snapshot samples of the correlation logs, you can see how much event reduction is currently happening and get a baseline for measuring any new or modified correlation developed.

Analyze the Events

On UNIX, a utility script (`/opt/OV/support/processEvents`) is provided to help with the analysis of the snapshot event dumps. `processEvents` analyzes the ascii event file by sorting the events according to their OID and generates a summary file detailing each event and its frequency. This is a good utility for easily determining de-duplication candidates.

Two additional data files are optional, but extremely valuable, for `processEvents`; they are `logonly` and `ov_events`. These data files are not required for the script to run but having them results in a better analysis. `logonly` is an ascii list of the OpenView log only trap ids. This file is read by `processEvents` and all events that are configured as log only by the management system are excluded from the frequency analysis. Example data from the logonly is as follows:

```
17.1.0.40000024
17.1.0.40000025
17.1.0.40000026
```

Since each management system may have its events configured differently it is desirable to generate the `logonly` file from `trapd.conf`. The following command is an example of how the log only data can be generated:

```
grep 'LOGONLY' trapd.conf | cut -d ' ' -f 3 | grep '17\.1' | sed -e \
's/\.1\.3\.6\.1\.4\.1\.11\.2\.//'
```

The second data file `ov_events` contains all the OpenView events. This file is read by `processEvents` to distinguish OpenView events from others. Example data from the `ov_events` file is as follows:

```
OV_HSRP_Down .1.3.6.1.4.1.11.2.17.1.0.60000395 "Status Alarms"
OV_HSRP_Up .1.3.6.1.4.1.11.2.17.1.0.60000396 "Status Alarms"
OV_HSRP_Unknown .1.3.6.1.4.1.11.2.17.1.0.60000397 "Status Alarms"
```

The following command shows how the `ov_events` data can be generated:

```
grep 'OV_' trapd.conf | cut -d ' ' -f 2-5 | grep '^OV_'
```

The syntax for invoking the analysis command is:

```
processEvents eventStoreDump summaryOutput
```

The file `eventStoreDump` is the ascii event store file from `ovdumpevents`. The file `summaryOutput`

is the analysis output. Example output from the summary file is as follows:

```
Total Number for trapId .1.3.6.1.2.1.10.32.0.1 = 1551
.1.3.6.1.2.1.10.32.0.1 is not an OV_ event
```

The first line gives the trap OID and count; the second line is an indication as to whether this is an OpenView trap. You can review this file to determine which events occur with the most frequency and may be candidates for event reduction.

Analyze the Correlation Log

On UNIX, a utility script (`/opt/OV/support/processCorrEvents`) is provided to help with the analysis of the snapshot correlation log dumps. `processCorrEvents` analyzes the ascii correlation output file by sorting correlation entries according to their parent ID and generates a summary file detailing how many events were correlated by each type of suppressor ID. Two separate tables are generated by this script; one for measuring de-duplication and the other for measuring correlation.

The syntax for invoking the command is:

```
processCorrEvents correlationLogDump summaryCorrResults
```

The `correlationLogDump` is the ascii dump of the correlation log generated by `ovdumpevents -c` and `summaryCorrResults` is the results file. Example output from the summary results is as follows:

```
DE-DUP Events Summary
*********************
Total number for trapId .1.3.6.1.2.1.16.0.1 = 3421
.1.3.6.1.2.1.16.0.1 is not an OV_ event

ECS Events Summary
******************
Total number for trapId 17.1.0.58916865 = 57
OV_Node_Down .1.3.6.1.4.1.11.2.17.1.0.58916865 "Status Alarms" Warning
```

Just as with `processEvents` the additional data files are `logonly` and `ov_events`.

The first section of the summary file shows all events that are currently being de-duplicated (although others may be configured for de-duplication but did not appear in the timeframe of the BES). For each event, the summary gives the event ID and how frequently it occurred in this sample. The second line gives an interpretation of the event ID, if it is available from the `ov_events` file.

The second section of the summary file shows events which are actually being correlated, although not by which correlation. It gives the event ID, its frequency, and its interpretation from the `ov_events` file.

You can review this file to determine the events which are already processed before you design correlators or de-duplication rules that may interfere with your running system.

This level of analysis provided by these scripts is by no means complete but it gives a good sense of event frequency and magnitude. It works quite well for understanding de-duplication or suppression candidates. The correlation analysis is good for establishing a baseline of correlation as well as measuring the effectiveness of any new correlator.

Capturing an Event Stream

How to Capture Events

You can use the `ecsmgr` and `ecsevgen` tools to capture a log of events on your runtime NNM management station, that can be played back in your testing environment when developing new NNM event reduction strategies.

You can capture events from either of two points:

- Logging all incoming events (to have a bunch of events to work with)
- Logging output and correlated events (to see if your new event reduction works)

Note: HP support might ask to see these files in certain troubleshooting situations.

Logging All Incoming Events

To capture all events that are actually entering the ECS engine, log in with root or administrator permissions and at the command line, type:

```
ecsmgr –log_events input on
```

This provides a log file of all events entering the ECS engine. The log file is named `ecsin.evt0`.

When this file reaches maximum size the data is copied to `ecsin.evt1`, and the newly received events are logged into `ecsin.evt0`. These files are located in:

UNIX:

`$OV_LOG/ecs/1/ecsin.evt0 and`

`$OV_LOG/ecs/1/ecsin.evt1`

Windows:

`<install_dir>\log\ecs\1\ecsin.evt0 and`

`<install_dir>\log\ecs\1\ecsin.evt1`

To turn off input event logging, log in with root or administrator permissions and at the command line type:

```
ecsmgr –log_events input off
```

To change the log size (512K default), log in with root or administrator permissions and at the command lint type:

```
ecsmgr –max_log_size event <kbytes>
```

These input log files can be used to recreate an input event scenario.

Logging Output and Correlated Events

To capture events (including newly created events) that are being output or discarded by the currently enabled ECS correlations and Composer correlators and De-Dup configuration, log in with root or administrator permissions and at the command line, type:

```
ecsmgr –log_events stream on
```

NOTE: You are logging all events in the NNM 'default' stream.

The log file is named `default_xxx.evt0`. When this file reaches maximum size, the data is copied to `default_xxx.evt1`, and the newly received events are logged into `default_xxx.evt0`. These files are located in:

UNIX:

`$OV_LOG/ecs/1/default_sout.evt0 and`

`$OV_LOG/ecs/1/default_sout.evt1`

Windows:

`<install_dir>\log\ecs\1\default_sout.evt0 and`

`<install_dir>\log\ecs\1\default_sout.evt1`

Events that are discarded by the stream (or suppressed by a correlation) are written to:

UNIX:

`$OV_LOG/ecs/1/default_sdis.evt0 and`

`$OV_LOG/ecs/1/default_sdis.evt1`

Windows:

`<install_dir>\log\ecs\1\default_sdis.evt0 and`

`<install_dir>\log\ecs\1\default_sdis.evt1`

To turn off stream event logging, log in with root or Administrator permissions and at the command line, type:

```
ecsmgr –log_events stream off
```

To change the log size (512K default), log in with root or administrator permissions and at the command line, type:

```
ecsmgr –max_log_size event <kbytes>
```

Replaying an Event Stream

Feeding or Replaying Events into the ECS Engine

To feed the captured events into the ECS engine for your test environment, log in with root or administrator permissions and at the command line type:

UNIX:

```
$OV_CONTRIB/ecs/ecsevgen -n <LogFileName>.evt0
```

Windows:

```
<install_dir>\contrib\ecs\ecsevgen -n <LogFileName>.evt0
```

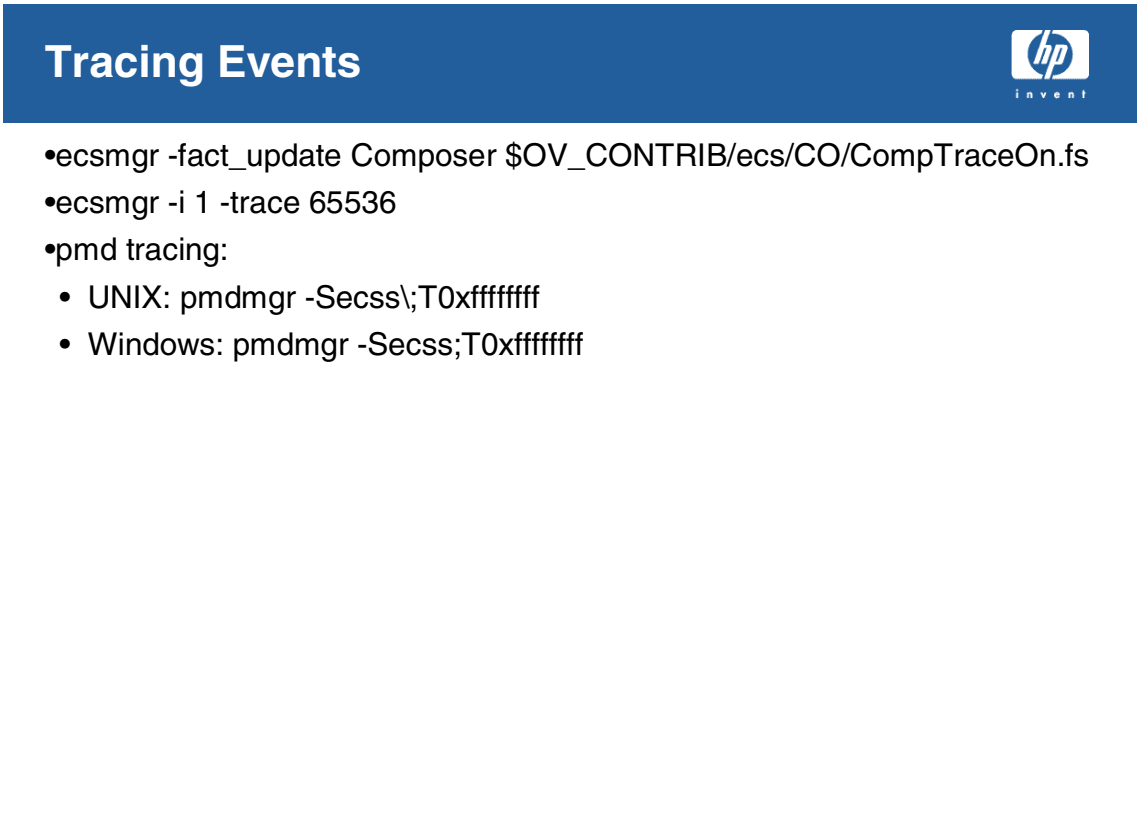
Input Event Log Example

Events that are written to the event log files have the following format. You can also manually create new events using an editor. However, you need to be familiar with SNMP trap formats to create a new event. It is recommended that you capture events using event logging and then modify or replicate the event as needed.

```
# eventid(0:43)                                  - Comment
+0                                               - Time delay in seconds
```

```
!1                                              - Number of times event is repeated
Trap-PDU {
    enterprise {1 3 6 1 4 1 11 2 17 1},
    agent-addr internet : "\x02\x0xq+",         - Network byte address eg, 10.10.10.10
    generic-trap 6,
    specific-trap 58916867,
    time-stamp 0,
    variable-bindings {
        {
            name {1 3 6 1 4 1 11 2 17 2 1 0},
            value simple : number : 76
        },
        {
            name {1 3 6 1 4 1 11 2 17 2 2 0},
            value simple : string : "10.10.10.10"
        },
        {
            name {1 3 6 1 4 1 11 2 17 2 3 0},
            value simple : number : 101
        },
        {
    }
}
% ber:Trap-PDU
```

Tracing Events



The pmd process has many types of trace messages and many of them are intended for experts who have internals knowledge of the NNM product. However, because Composer is a correlation within ECS it is necessary to use pmd tracing to trace the correlators within Composer.

A special debugging fact store was developed for Composer to make it easier to trace flow within correlators. For anyone intending to do Composer tracing it is essential they first read the *HP OpenView Correlation Composer's Guide*, Trouble Shooting the Composer during Runtime.

To do runtime tracing of Composer:

1. First load the debugging fact store.

   UNIX:

   ```
   ecsmgr –fact_update Composer $OV_CONTRIB/ecs/CO/CompTraceOn.fs
   ```

   Windows:

   ```
   ecsmgr –fact_update Composer %OV_CONTRIB%\ecs\CO\CompTraceOn.fs
   ```

2. Secondly tracing needs to be turned on for the ECS stack and then turned on for the pmd process (run as root or Administrator):

UNIX:

```
ecsmgr –i 1 –trace 65536
pmdmgr –Secss\;T0xffffffff -Qt -Ql
```

Windows:

```
ecsmgr –i 1 –trace 65536
pmdmgr –Secss;T0xffffffff -Qt -Ql
```

The -Qt -Ql (ell) truncate and clear the trace file prior to each test.

The trace output is written to:

UNIX:

```
$OV_LOG/pmd.trc0
```

Windows:

```
%OV_LOG%\pmd.trc0
```

(Not `.log0` as in the Composer manual.)

The following is example output from the Composer tracing:

```
TRACE [interpreter]: Composer : 19700101000000.000000Z : "eventid(0:34237)" :
OV_MultipleReboots : Incoming Alarm passed Alarm signature for this correlator

TRACE [interpreter]: Composer : 19700101000000.000000Z : "eventid(0:34237)" :
OV_MultipleReboots : Alarm passed both primary and advanced filter for Correlator

TRACE [interpreter]: Composer : 19700101000000.000000Z : "eventid(0:34237)" :
OV_MultipleReboots : Executing logic for the Correlator - starting

TRACE [interpreter]: Composer : 19700101000000.000000Z : "eventid(0:34237)" :
OV_MultipleReboots : The Correlator has decided the following - :Event will be output.
```

As stated before the output from `pmd` tracing is extremely verbose and quite a lot of it won't make sense in the context of tracing a correlator. To see just those trace messages relevant to Composer, the `pmd.trc0` file should be `grep'd` for the lines that have 'Composer' in them. The above output was obtained by doing:

```
grep 'Composer' pmd.trc0 | grep 'OV_MultipleReboots'
```

This allows you to see all the state transitions occurring within your correlation, such as passing the Alarm Signature, Alarm Filter, state prior to calling a function, and state up return from a function.

To turn the tracing off do the following:

1. UNIX: `ecsmgr –fact_update Composer $OV_CONTRIB/ecs/CO/CompTraceOff.fs`

   Windows: `ecsmgr –fact_update Composer %OV_CONTRIB%\ecs\CO\CompTraceOff.fs`

2. And also turn the tracing off in the ECS stack of `pmd`.

   UNIX:

   ```
   ecsmgr –i 1 –trace 0
   pmdmgr –Secss\;T0x0
   ```

   Windows:

   ```
   ecsmgr –i 1 –trace 0
   pmdmgr –Secss;T0x0
   ```

Function Debugging Tips

Debugging Tips

- Get a list of after-market functions
- Put debug statements in your function rather than use pmd tracing

©2004 Hewlett-Packard Development Company, L.P.

Identifying New Callouts

If when developing a new Correlator the pmd process aborts (dumps core) then it is most likely due to a newly introduced function or perl call out. To determine any new function calls quickly, use the following command:

```
grep 'lib.*:' $OV_CONF/ecs/correlations/Composer.fs | grep '^(1' | cut -f 2  -d ' '
```

The following is the output from the Composer.fs supplied with the NNM product:

```
"libHSRPStatus:Orch_isHSRPInterface",
"libHSRPStatus:Orch_isThisHSRPGroupBeingProcessed",
"libOrchNNM:Orch_log_correlations",
"libOrchNNM:Orch_topoAddrToTopoInfo",
"libOrchNNM:Orch_chassisInput",
"libHSRPStatus:Orch_checkAndComputeHSRPStatus",
"libHSRPStatus:Orch_isHSRPGroupBeingProcessed",
"libHSRPStatus:Orch_getHSRPGroupFromTrap",
```

If they are changes to the functions being called or new ones added then these will be the most likely places to look for the problem. To quickly determine any new perl script callouts use the following command:

```
grep 'perl' $OV_CONF/ecs/correlations/Composer.fs | grep '^(1' | cut –f 2  -d ' '
```

There are no perl scripts used in the Composer.fs provided with NNM so the default results are empty.

Instrumenting the Function

Unless you are already familiar with `pmd` tracing and ECS, the task of tracing at the `pmd` level can be a bit daunting. An alternative technique is to instrument the Correlator from within.

To do this simply add an input variable to that invokes a trace perl script. The perl script can write a message to some file indicating this event passed the input signature. Similarly a variable can be added to advanced filter and to the call back. These would indicate the correlator has proceeded to the advanced filter and to the completion point, respectively.

Logging Composer Events

You can log the events that Composer sends out and the events that Composer discards using the command `ecsmgr -log_events circuit Composer on`. The output events are written to `Composer.cout.evt0` and the discarded events are written to `Composer.cdis.evt0`. To turn the logging off again, type `ecsmgr -log_events circuit Composer off`.

Lab Exercises

- Review current event database
- Turn on tracing
- Turn off tracing

 ©2004 Hewlett-Packard Development Company, L.P.

1. Review the contents of your current Binary Event Store to see if you have any candidates for correlation or de-duplication.
   a. Use the following commands to see the contents of the BES:
      1. `cd /opt/OV/support`.
      2. `ovdumpevents -s "default" >eventStoreDump`
      3. `ovdumpevents -c "default" >correlationLogDump`
      4. Briefly review the files to see their formats.
   b. (UNIX only) Create the helper files for the analysis tools
      1. Create the `logonly` file using the command

```
grep 'LOGONLY' $OV_CONF/C/trapd.conf | cut -d ' ' -f 3 | grep '17\.1' | sed -e \
's/\.1\.3\.6\.1\.4\.1\.11\.2\.//' >logonly
```

      2. Create the `ov_events` file using the command

```
grep 'OV_' $OV_CONF/C/trapd.conf | cut -d ' ' -f 2-5 | \
grep '^OV_' > ov_events
```

c. (UNIX only) Analyze the event log.

1. `cd /opt/OV/support`
2. `./processEvents eventStoreDump summaryOutput`
3. Execute `more summaryOutput`.
4. What are your top 5 events by frequency?

d. (UNIX only) Analyze the correlation log.

1. `cd /opt/OV/support`
2. `./processCorrEvents correlationLogDump summaryCorrResults`.
3. What are the top 5 already being de-duplicated in `summaryCorrResults`?
4. What are the top 5 events already being correlated?
5. How would you use these two files together in designing correlators?

2. The file `$OV_CONTRIB/OVTraining/NNM3/MultipleReboot.evt` simulates the capture of a series of starup events from a connector. Examine `MultipleReboot.evt` to see what the capture looks like. How many events are in the file?

3. Turn on tracing. Run `MultipleReboot.evt` to see how the tracing works. Note when the functions get evaluated.

a. Turn on tracing.

1. Ensure that your Composer and ECS configuration GUIs are closed.
2. Load the debugging fact store for Composer by running `ecsmgr -fact_update Composer $OV_CONTRIB/ecs/CO/CompTraceOn.fs`.
3. `ecsmgr -i 1 -trace 65536`
4. UNIX: `pmdmgr -Secss\;T0xffffffff -Qt -Ql`

   Windows: `pmdmgr -Secss;T0xffffffff -Qt -Ql`

b. Run `MultipleReboot.evt`.

1. Open your All Alarms Browser.
2. `cd $OV_CONTRIB/ecs`
3. `$OV_CONTRIB/ecs/ecsevgen -n MultipleReboot.evt`
4. What did you see in the Alarms Browser?

c. Review the trace file in `$OV_LOG/pmd.trc0`.

d. (UNIX only) Find the trace messages relevant to Composer and this correlator by typing `grep 'Composer' $OV_LOG/pmd.trc0 | grep 'OV_MultipleReboots'`.

4. Modify MultipleReboot correlator to have `count=5` and rerun `MultipleReboot.evt`. Run it again. Return `count=4`, the original value.

5. Run the unit test `NodeIF.evt` and monitor the trace output to see how the correlators work

together.

a. Clear the `pmd` trace file using `pmdmgr -Secss\;T0xffffffff -Qt -Ql` (UNIX) or `pmdmgr -Secss;T0xffffffff -Qt -Ql` (Windows).

b. Open your All Alarms Browser.

c. `$OV_CONTRIB/ecs/ecsevgen -n NodeIf.evt`

d. Explain what you see in your Alarm Browser.

e. (UNIX only) Find the trace messages relevant to Composer and this correlator by typing `grep 'Composer' $OV_LOG/pmd.trc0 | grep 'OV_NodeIf'`. Which actual correlators were activated?

6. Turn off tracing and remove the debugging Composer.fs.

a. `ecsmgr -i 1 -trace 0`

b. UNIX: `pmdmgr -Secss\;T0x0`

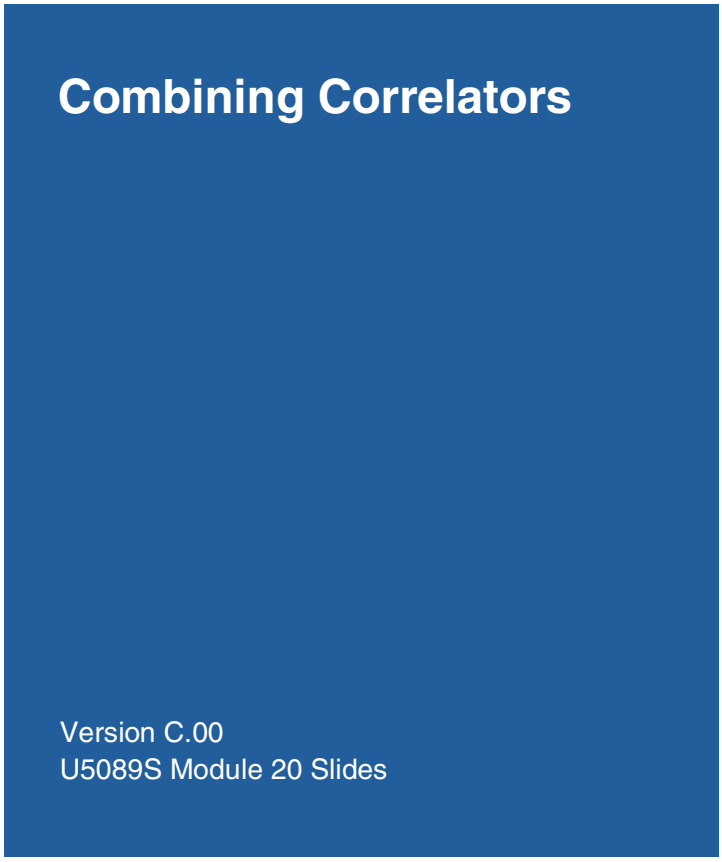
Windows: `pmdmgr -Secss;T0x0`

c. `ecsmgr -fact_update Composer $OV_CONTRIB/ecs/CO/CompTraceOff.fs`

7. Last: Delete all class-created correlators before continuing class.

20 Combining Correlators

Module Objectives

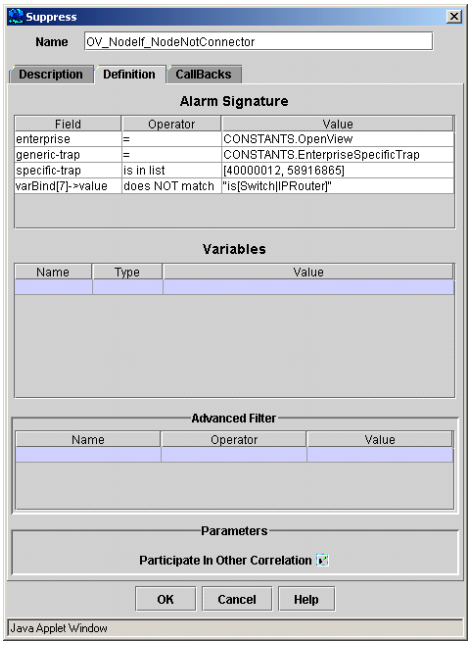


At the completion of this module, you will be able to:

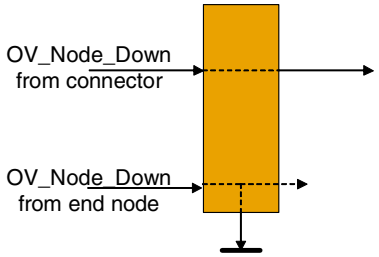
- Describe how correlators work together to accomplish an objective.

Examine the OV_NodeIF Correlators

Examine the OV_NodeIF Correlators



- Suppress all OV_Node_Unknown and OV_Node_Down when the node is not a switch or an IP router.
- Pass all OV_Node_Unknown and OV_Node_Down from switches and IP routers.
- Although the alarm is suppressed, the event may participate in other correlation.



U5089S ©2004 Hewlett-Packard Development Company, L.P. 2

In some cases you may require multiple correlators to solve a single event reduction problem. Each correlator can handle one aspect of the problem. Together they control the level of noise in the Alarm Browser. As an example, look at the collection of OV_NodeIf correlators and how they work together to identify connector node down events.

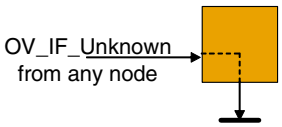
The first correlator in this group is the OV_NodeIF_NodeNotConnector correlator. This correlator examines all Node_Down and Node_Unknown events to see if they are coming from connecting devices. Events from end nodes are suppressed, but not until after they are examined by other correlators that may need their information. The do not show in the Alarm Browser in the end.

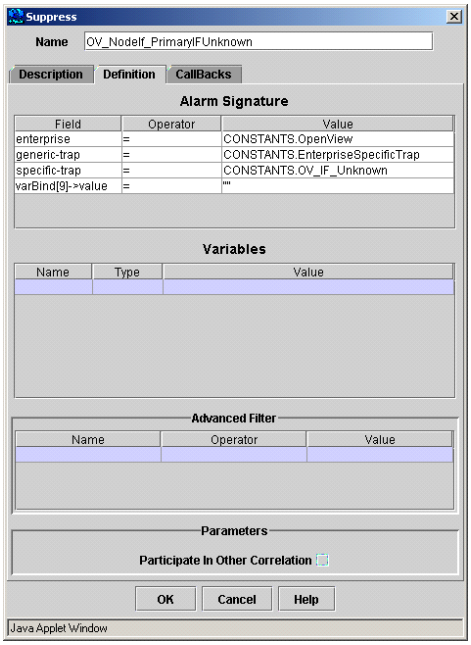
Events from connecting devices (where the varbinds include isSwitch or isIPRouter) are not processed by this correlator (don't pass the Alarm Signature), so they continue in the event stream.

Examine the OV_NodeIF Correlators

Examine the OV_NodeIF Correlators

- Suppress all OV_IF_Unknown events from the local management station.
  - varbind 9 is the ID of the management station.
- This takes care of status alarms from unconnected interfaces in routers and switches.
- Do not participate in any other correlators.





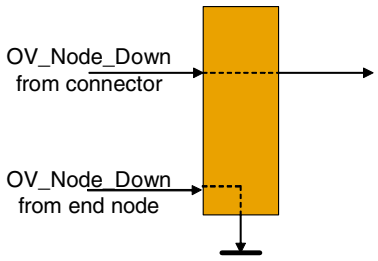
©2004 Hewlett-Packard Development Company, L.P.

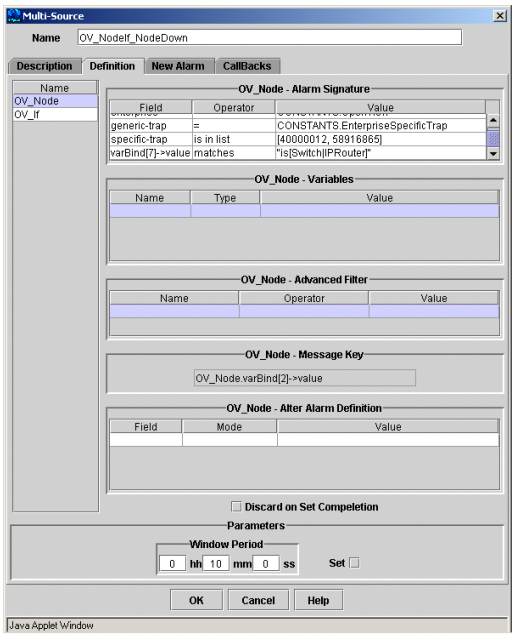
The second in this group of cooperating correlators is OV_NodeIF_PrimaryIFUnknown. This correlator only looks at IF_Unknown events. If they are being reported by the management station itself, they are suppressed. This removes alarms from unconnected interfaces on connecting devices from the Alarm Browser.

Examine the OV_NodeIF Correlators

Examine the OV_NodeIF Correlators

- Correlate OV_Node_Unknown or OV_Node_Down for all switches and IP routers.
- Match on name of the node (varbind 2).





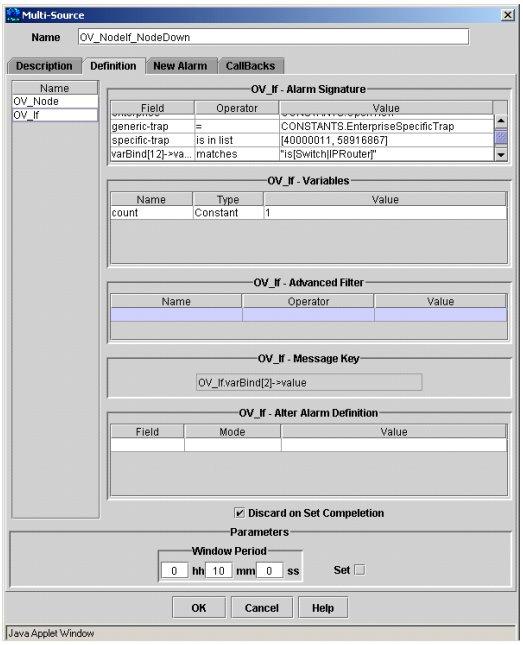
©2004 Hewlett-Packard Development Company, L.P.

The third in this group of cooperating correlators is OV_NodeIF_NodeDown. This is a MultiSource correlator. The events are matched up based on the node name (varbind 2). The set box is not checked, so all related events are being “collected”.

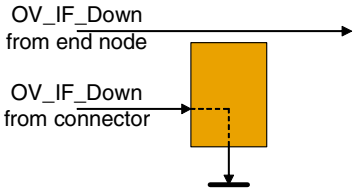
The first event stream looks at all Node_Down and Node_Unknown events from connectors.

Examine the OV_NodeIF Correlators

Examine the OV_NodeIF Correlators



- If an OV_IF_Unknown or OV_IF_Down from the same source arrives in a 10 minute window, discard it.
- Discard callback causes ECS to correlate this under the parent Node event.

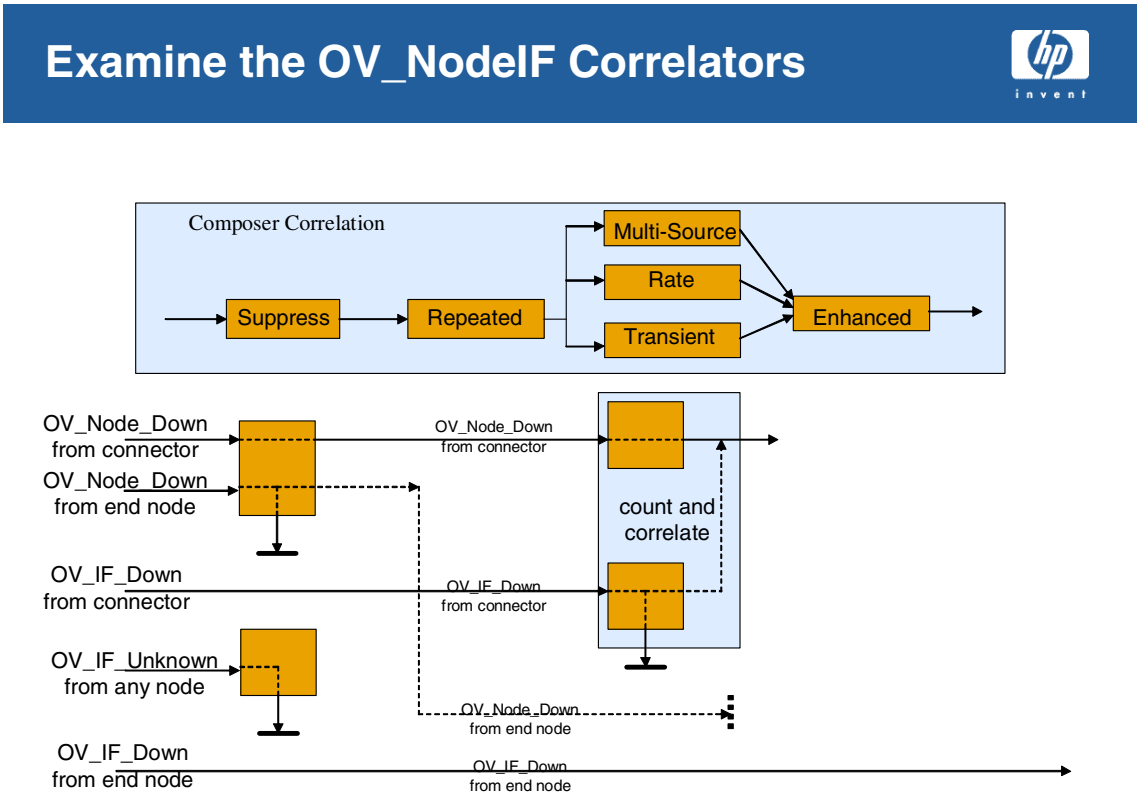


U5089S ©2004 Hewlett-Packard Development Company, L.P. 5

The second part of the correlator looks at corresponding IF_Down or IF_Unknown events from the same connector node. All interface events that arrive from the same node within 10 minutes are set to be discarded.

Since the Alarm Signatures for both parts specify connector devices only, all end node events are ignored by this correlator.

Examine the OV_NodeIF Correlators



When we diagram the three correlators together and compare them against the diagram that shows the order of processing of correlators, we can see that the end result is that a Node_Down event from a connector will have all its IF_Down events correlated under it and only the root cause Node_Down is emitted to the Alarm Browser. (The same holds for Unknown events.)

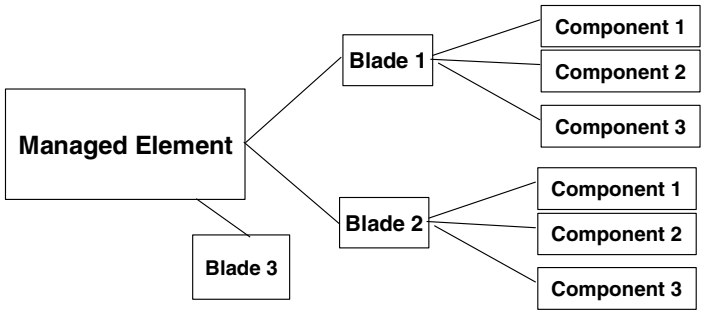
Node_Down events from end nodes are processed by other correlations (dotted line), but are suppressed before they arrive at the Alarm Browser.

IF_Down events are the ones that show in the Alarm Browser from end nodes.

Advanced Function Example - Problem Statement

Advanced Function Example

- A managed element has blades with ‘n’ number of components.
- The requirement is if all the components on a given blade fail within 10 seconds
  - Discard all individual component failure traps
  - Emit a new event indicating that the blade has failed and containing the names of the individual components.



 ©2004 Hewlett-Packard Development Company, L.P.

The semantics of the Multi-Source correlator is such that it can be configured to drop all alarms and create a new alarm when the set is complete. This conceptually fits the requirement. The failure of the last component has nothing special in the trap to indicate that it’s the last component failure. A mechanism to distinguish the last alarm from the rest of the alarms must be devised.

The trap contains the blade number in varBind[0].value and the ID of the component in varBind[1].value.

The number of components in the device can be determined via a call to an external function, getNumComponents.

A Component trap PDU looks like

```
Trap-PDU{
enterprise {1 2 3 4},
agent-addr internet : "\x0A\x00\x01\x7F",
generic-trap 6,
specific-trap 21,
time-stamp 414746291,
```

```
variable-bindings{
{
name {1 3 6 1 4 11 2 17 2 1 0},
value simple : number : 40
}
{
name {1 3 6 1 4 11 2 17 2 1 0},
value simple : number : 4
}
}
}
```

Before you begin to configure this correlator, what do you need to know?

1. Why are we required to use the Multi-Source Correlator?

   There are two kinds of traps which are to be monitored in the this correlator.

   - Last trap - This is the last component_fail trap which is emitted by the blade.
   - NonLast trap- These are all component_fail traps other than the last trap emitted by the blade.

2. How do you determine whether the trap that just arrived is the Last trap?

   Check whether the number of traps that arrived is less than the total number of components connected to the blade.

   - In the nonLast trap Alarm Definition, write the following functions and bind them to variables:
     - — SetCounter - to increment the counter value by 1 if the trap that just arrived is a nonLast trap.
     - — GetNumComponents - to return the number of components attached to the blade.
     - — getCounter - to extract the value of the counter which has been stored.
     - — storeStr - to store the component ID of component that failed.
   - In the Advanced Filter of nonLast trap, compare the return values of getCounter and getNumComponents. If it is lower, then the trap fills both the primary and secondary filter and it enters the correlator as a nonLast trap. So, alarms enter the correlator till the above condition is satisfied. Note that all these are nonLast traps
   - In the Last trap Alarm Definition, write the following functions and bind them to variables:
     - — GetCounter - to extract the number of traps which has been stored.
     - — GetNumComponents - to return the number of components attached to the blade.
     - — RetrieveStr - to retrieve the list of component IDs.
   - In the Advanced Filter of the Last trap, compare the values returned by the functions getNumComponents and getCounter. If they are equal, then the trap enters the correlator as a Last trap.

3. What do you do once the traps have been distinguished?

If the trap that just arrived is a nonLast trap,

- increment counter
- Store the value of component ID using the storeStr built in function

If the trap that just arrived is the Last trap

- Retrieve values using the retrieveStr built in function
- Evaluate the storeStr function before the trap is discarded

4. How do you ensure that the alarms are correlated under the same instance?

   Since the traps are identical, set the Message Key to blade ID.

5. How do you finally meet the requirements?

   In both the Last and nonLast trap, check the [Set] and [Discard on set completion] buttons. This ensures that the trap is discarded once the set is complete

   Create the new alarm.

Advanced Function Example - Design

Advanced Function Example - Design

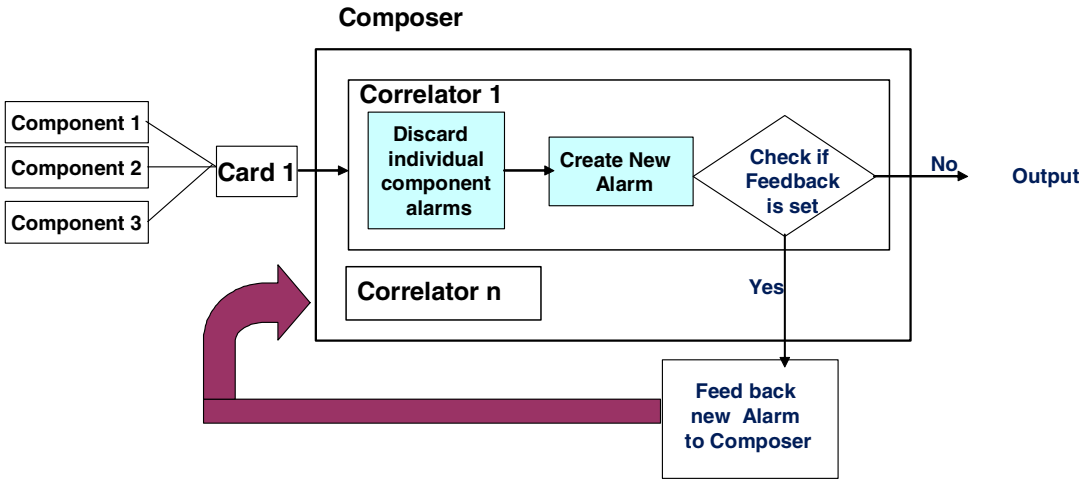
- Create a Multi-Source correlator with two traps
  - Last trap
  - nonLast trap
- In the Advanced Filter, check whether the trap that just arrived is the Last trap.
  - Is the number of traps that have arrived < total number of components from **`getNumComponents`**?
- If the trap is a nonLast trap, increment a counter using the setCounter( ) built-in function.
- Evaluate the storeStr function before the trap gets discarded. It can be done by evaluating the variable in the Advanced Filter.
  - Create a variable X of type Function and in the Advanced Filter, compare X=X to evaluate this function when the trap enters the correlator.
- When the last component fails the set will be complete.
- Store the names of component_fail alarm.

©2004 Hewlett-Packard Development Company, L.P.

Concept of Feedback

Concept of Feedback

•After a new event is created, it can be fed back into the correlator again and it can participate in other correlators.



©2004 Hewlett-Packard Development Company, L.P.

Feedback Example

Feedback Example

- Problem Statement
- Extend Advanced Function Example
- The requirement is
  - discard all Chassis traps emitted from the same managed entity for 10 seconds after the blade failure event is received.
- Solution Design
  - Use Multi-Source Correlator that is configured for two event types, the chassis fail event and the component fail event.
  - Configure the correlator to drop all chassis events if the set is complete (that is, when the all_components_fail event also arrives).
  - Feed the all_components_fail alarm back into the system (where all correlators can see it) .

©2004 Hewlett-Packard Development Company, L.P.

The Feedback Example is designed to be working in continuation of the Advanced MultiSource Example. Once the card_fail (that is, the new event) event has been emitted from Advanced MultiSource Example, feedback the event into ECS.

Select the `Feedback` button in the New Alarm Specification section to feed the event back into ECS and participate in other correlators.

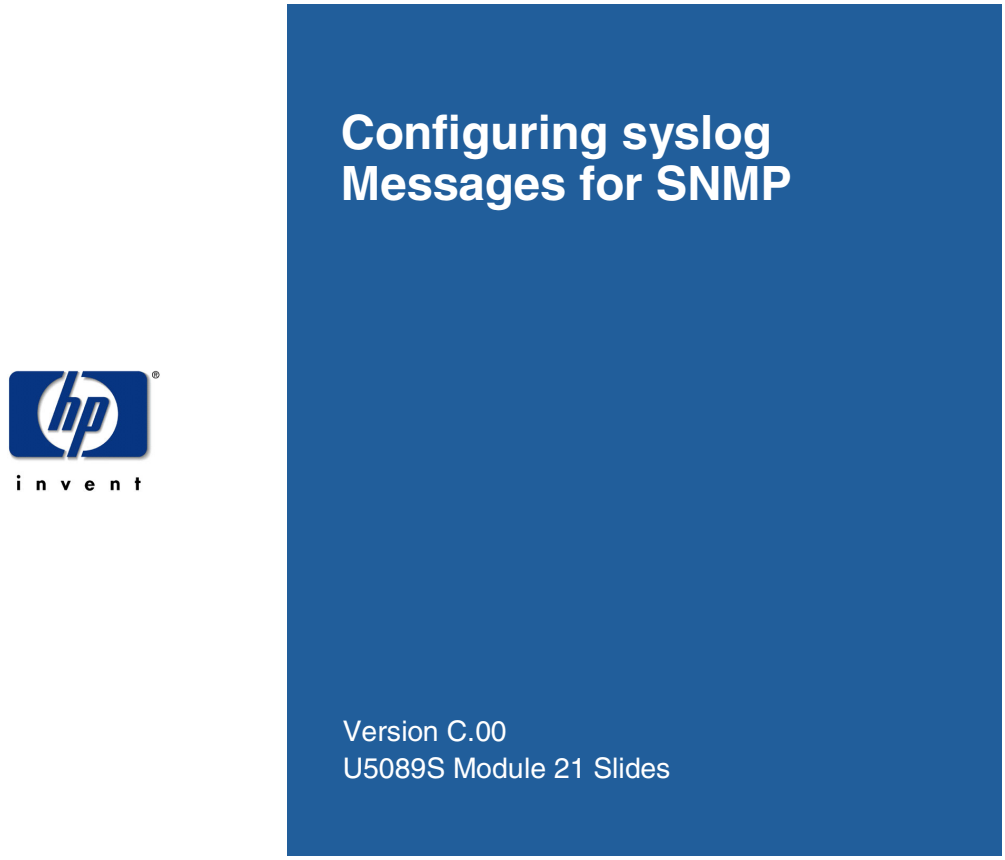
Lab Exercises

Lab Exercises

©2004 Hewlett-Packard Development Company, L.P.

21 Configuring syslog Messages for SNMP

Module Objectives



At the completion of this module the student will be able to:

- Give reasons for converting syslog messages into SNMP traps.
- Describe how NNM's syslog conversion software works with and without OVO installed.
- Describe NNM syslog interaction with OVO 8.0.
- Recognize the default syslog to trap mappings.
- Configure specific syslog message patterns to be mapped to SNMP traps.
- Deploy a syslog configuration.

References

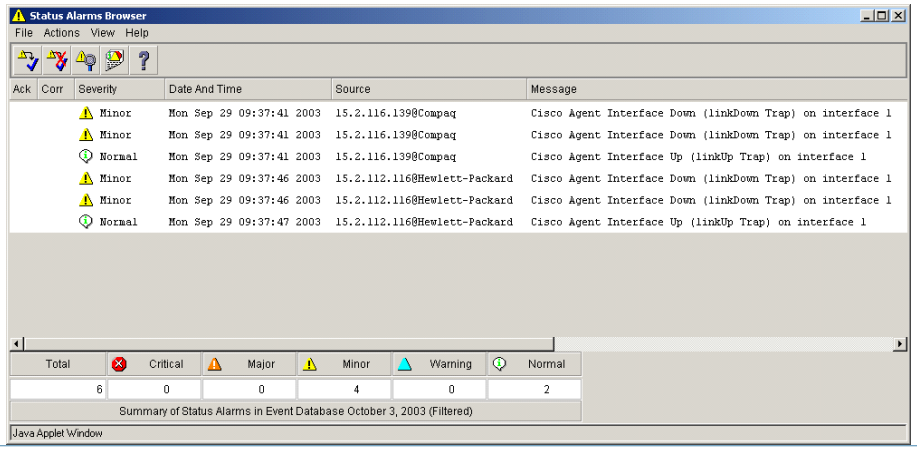
See the *Syslog Integration White Paper* for more information or details on operation in an OVO

environment.

Converting syslog Messages to SNMP Traps

Converting Syslog Messages to SNMP Traps

- Convert syslog messages to OV traps
- Configurable message pattern match
- Not intended for high volume syslog message systems
- Alternative to SNMP trap based management
- Used in root cause analysis
- Enabler for OpenView Solutions



 ©2004 Hewlett-Packard Development Company, L.P.

In many cases the information provided by the syslog message is redundant to what can come in SNMP traps. The primary value of handling syslog messages is to allow for more flexibility in how the network can be managed (that is, syslog instead of traps). Also, operator productivity can be improved, as they are not required to manually monitor the syslog messages along with the SNMP traps.

NNM includes a component that enables certain syslog messages to be parsed and converted into SNMP traps on the NNM management station. NNM customers have for several releases requested the syslog feature. Logec/TUSC have in the past provided this feature, but at significant additional cost to the customer. The specific features of this syslog integration are actually more comprehensive and more extensible than those of the Logec product. Logec's syslog feature supports just the Cisco syslog trap, whereas NNM's mechanism can be extended to support any trap with any type of syslog message format.

The Syslog Integration functionality provided with HP OpenView NNM Advanced Edition enables the management of network equipment from syslog messages. Certain types of network equipment do not have SNMP traps nor supporting MIBs for all error and warning conditions. Often for new equipment, syslog messages are defined before SNMP traps are standardized (for example, PIX firewall routers' Failover and Failback messages). For operators who require managing these conditions, the Syslog Integration functionality provides the ability to map syslog messages into SNMP traps for presentation or root cause analysis.

NNM includes out-of-the-box conditions for which syslog messages are mapped to SNMP traps.

They focus on Frame Relay WAN link management. You can add new conditions through the NNM Syslog Trap Mapping Configuration interface or the HP OpenView Operations message source template configuration windows, depending on your deployment mode.

Each type of syslog message to be mapped is described as a single match condition. Custom message attributes (CMA's) are defined in this template that identify all the SNMP trap attributes. These CMA's must be used correctly by every condition in the template.

syslog Deployment Models

- UNIX management stations only
- NNM
  - **ovspmd** manages Control Agent
  - Default templates install as XML file
  - Generate Template from XML
  - GUI editor for the XML file
- OVO
  - Works in OVO environments
  - OVO manages NNM node
  - Default templates installed in OVO Manager
  - Coexists with Network SPI

 ©2004 Hewlett-Packard Development Company, L.P.

Syslog Integration works with NNM Advanced Edition. Syslog Integration also works with HP OpenView Operations with NNM. Syslog Integration must be configured on an NNM management station running a UNIX® operating system. See the *Release Notes* for supported software versions.

This component includes a standalone OVO agent deployed on the NNM management station to parse the syslog file. Mapping the syslog message to SNMP is done via an additional process that intercepts messages from the OVO agent, translates the message to SNMP and sends the trap to NNM.

The syslog parser component is managed differently depending upon whether OVO is deployed or only NNM is deployed.

NNM Standalone

NNM installs a standalone OVO agent for the syslog parsing capabilities. The agent is a standard OVO 7.1 agent bundle. The install script for NNM installs the OVO agent bundle. It comes with its own configuration GUI. The syslog integration component has no dependencies on the Extended Topology component. The syslog integration component is not activated during installation.

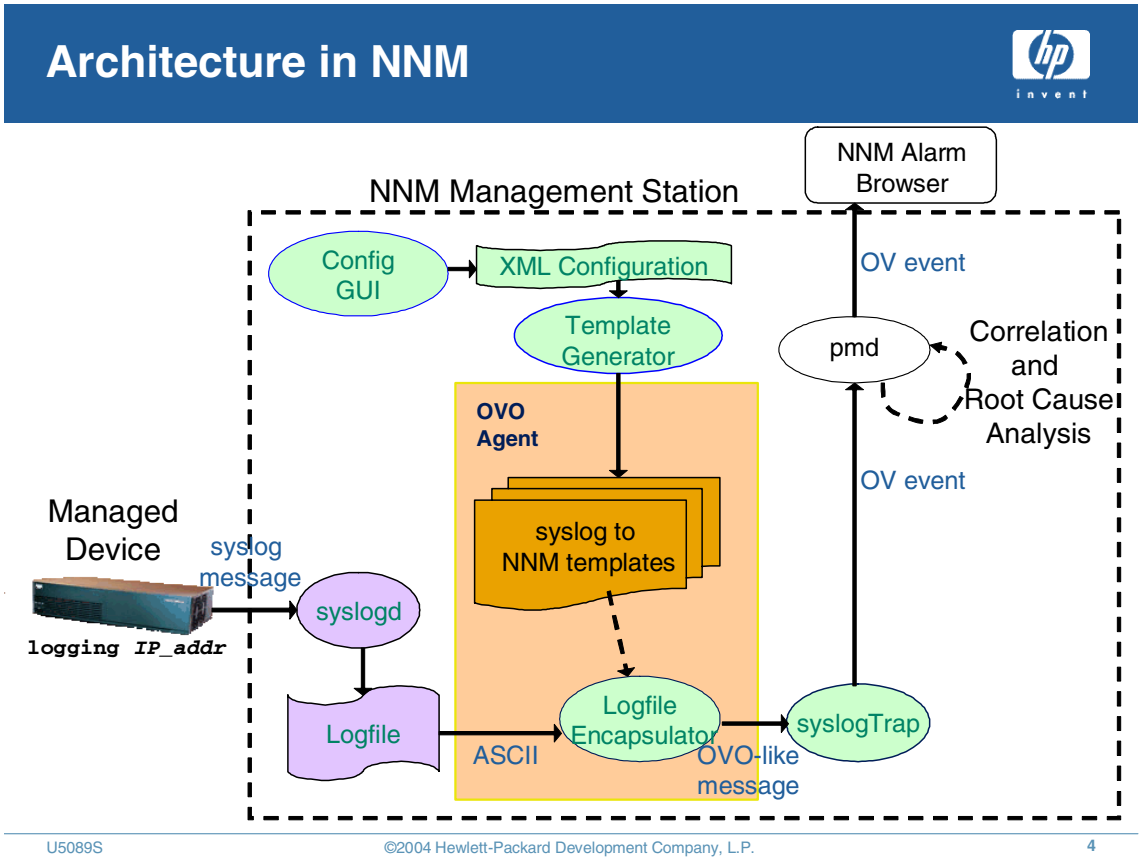
With OVO Installed

When OVO is deployed with NNM, the agent is not embedded with NNM; instead it coexists on the management station with NNM. Use the normal OVO agent deployment processes to deploy the agent to the NNM management station. (Note this is typically already done when OVO is used as the MOM.) Use the OVO template editor to manage the template.

The agent running on the NNM management station needs to be configured with the syslog template. To deliver the out-of-box mappings, `setupSyslog.ovpl -server` generates an uploadable template that can be uploaded into the OVO database. From that point, you can use the OVO management system to configure the syslog agent on the NNM station. All modifications/extensions to the templates are done using the standard OVO template editor. The NNM syslog template configuration process should not be used in this case.

For OVO 8.0 integration, upload the NNM template to the OVO server, convert the template to a policy, and download the OVO 8.0 agent and policy.

Architecture in NNM



When Syslog Integration is configured, an OVO agent is installed on the NNM management station. Part of the embedded OVO agent is the **logfile encapsulator** that is responsible for listening for syslog events from logfiles located in:

HP-UX: `/var/adm/syslog/syslog.log`

Solaris: `/var/adm/messages`

The syslog file must reside on the same machine as the NNM software.

The logfile encapsulator filters and formats syslog events according to information in configured message source templates. The logfile encapsulator then forwards relevant information in the form of OVO messages to an NNM daemon process, **syslogTrap**, which maps the syslog messages into SNMP traps.

The exact set of messages that will flow into `syslogTrap` is determined by the deployed template. The deployed template specifies the exact pattern for a match as well as the values of the CMA's (customized message attributes). Each SNMP trap attribute has a corresponding CMA set in the OVO message.

When Syslog Integration is configured, the NNM management station receives formatted syslog messages from the embedded OVO agent. `syslogTrap`, a daemon on the NNM management station, maps the syslog messages to OpenView events. This daemon process registers with the OVO agent at the message stream interface and intercepts all OVO messages of message type

NNMsyslog_.

All OVO messages that match a configured pattern (such as NNMsyslog_) are mapped into an OV event and sent into NNM via `pmd`.

The SNMP traps are then sent to the NNM postmaster process, `pmd`, where the traps can participate in correlation or analysis. For example, the NNM Smart Plug-in for LAN/WAN Edge provides advanced correlators for frame relay traps and syslog messages. `pmd` forwards the formatted syslog messages to the NNM Alarm Browser Status Alarm category.

NNM control integration of the OVO agent is done by having `ovspmd`-registered scripts perform the start/stop/status management the embedded OVO control agent. The control agent then manages the log file encapsulator and message agent. The embedded agent processes are `opctla` (the control agent), `opcle` (the logfile encapsulator), and `opcmsga` (the message agent where OVO messages are intercepted).

Configuration of the syslog template is done through a graphical interface to edit an XML (`syslogtrap.xml`) file that encapsulates all the template configuration data. The Syslog Config UI configures only the message conditions. All other aspects of the OVO templates are hidden from the user.

A command line utility generates and encrypts the template configuration data from the XML file. The command and XML file are transparent to the user in the configuration process.

Default Trap Mappings

Default Trap Mappings

- Initial configuration is geared toward WAN link management.
- Syslog messages
  - LINEPROTO
  - LINK
  - FR-DLCI
  - OSPF - neighbor DOWN, FULL
  - HSRP - State changes
  - Cisco board messages – down, up, online, reset, configuration mismatch, board removed, board inserted
  - Port aggregation messages – joined bridge port, left bridge port, various error conditions
  - DTP – dynamic trunking protocol (*)
  - STP – spanning tree protocol (*)
  - BGP – border gateway protocol (*)

U5089S ©2004 Hewlett-Packard Development Company, L.P. 5

Because the syslog messages are different from their corresponding SNMP traps, the mapped traps are introduced as OpenView events. Many of the syslog messages will be converted to log-only traps that become factored into root cause analysis and become correlated to root cause events. In instances when the device may send both a trap and a logging message, the syslog-generated OV event and the device's trap are correlated together.

The `Syslog to NNM` template includes out-of-the-box message condition definitions to match the following types of messages:

- Cisco board messages
- HSRP messages
- OSPF messages
- Port Aggregation Protocol (PAGP) messages
- Dynamic Trunking Protocol (DTP) messages
- Border Gateway Protocol (BGP) messages
- Spanning Tree Protocol (STP) message

New OpenView events are defined to support the out-of-box mappings provided with the syslog

mapper. Currently the list of mapped traps is:

Table 21-1 Syslog Trap Mappings

Syslog Message	OpenView Event Generated
%LINK-3-UPDOWN (down)	OV_Syslog_LinkDown
%LINK-3-UPDOWN (up)	OV_Syslog_LinkUp
%LINEPROTO-5-UPDOWN (down)	OV_Syslog_LineProtoDown
%LINEPROTO-5-UPDOWN (up)	OV_Syslog_LineProtoUp
%FR-5-DLCICHANGE (INVALID)	OV_Syslog_FrameDLCI_Inactive
%FR-5-DLCICHANGE (INACTIVE)	OV_Syslog_FrameDLCI_Inactive
%FR-5-DLCICHANGE (ACTIVE)	OV_Syslog_FrameDLCI_Active
%OSPF-5-ADJCHG (DOWN)	OV_Syslog_OSPF_Neighbor_Down
%OSPF-5-ADJCHG (FULL)	OV_Syslog_OSPF_Neighbor_Up
%STANDBY-6-STATECHANGE (Speak)	OV_Syslog_HSRP_State_Speak
%STANDBY-6-STATECHANGE (Standby)	OV_Syslog_HSRP_State_Standby
%STANDBY-6-STATECHANGE (Active)	OV_Syslog_HSRP_State_Active
%STANDBY-6-STATECHANGE (Init)	OV_Syslog_HSRP_State_Init
%STANDBY-3-DUPADDR	OV_Syslog_HSRP_Duplicate_Address
%SNMP-5-MODULETRAP	OV_Syslog_Board_Up OV_Syslog_Board_Down
%SYS-5-MOD_NORESPONSE	OV_Syslog_Board_Failure
%SYS-5-MOD_OK	OV_Syslog_Board_Online
%SYS-5-MOD_REMOVE	OV_Syslog_Board_Removed
%SYS-5-MOD_INSERT	OV_Syslog_Board_Inserted
%SYS-5-MOD_RESET	OV_Syslog_Board_Reset
%SYS-3-MOD_FAIL	OV_Syslog_Board_Failure
%SYS-3-MOD_FAILREASON	OV_Syslog_Board_Failure
%SYS-3-MOD_CFGMISMATCH1	OV_Syslog_Board_Config_Mismatch
%SYS-3-MOD_CFGMISMATCH2	OV_Syslog_Board_Config_Mismatch
%SYS-3-MOD_CFGMISMATCH3	OV_Syslog_Board_Config_Mismatch
%SYS-3-MOD_CFGMISMATCH4	OV_Syslog_Board_Config_Mismatch
%PAGP-5-PORTTOSPT	OV_Syslog_PAGP_JoinedBridgePort
%PAGP-5-PORTFROMSPT	OV_Syslog_PAGP_LeftBridgePort

Table 21-1 Syslog Trap Mappings (Continued)

Syslog Message	OpenView Event Generated
%DTP-3-TRUNKPORTFAIL	OV_Syslog_Trunk_Port_Fail
%DTP-3-NONTRUNKPORTFAIL	OV_Syslog_Trunk_NonTrunkPort_Fail
%DTP-5-TRUNKPORTON	OV_Syslog_Trunk_Port_On
%DTP-5-TRUNKPORTCHG	OV_Syslog_Trunk_Port_Change
%OSPF (all other OSPF messages)	OV_Syslog_OSPF_Default_Message
%STANDBY (all other HSRP messages)	OV_Syslog_HSRP_Default_Message
%PAGP (all other PAGP messages)	OV_Syslog_PAGP_Default_Message
%SYS-*n*-MOD (all other Cisco board messages)	OV_Syslog_Card_Default_Message
%DTP (all other %DTP trunk messages)	OV_Syslog_Trunk_Default_Message
%SPANTREE	OV_Syslog_Spantree_Default_Message
%BGP	OV_Syslog_BGP_Default_Message

This list may grow as new customer requirements are added and to enhance NNM's general network management capabilities.

Configuration Prerequisites

Configuration Prerequisites

- Install DCE Software on HP-UX 11.0
- HP-UX 11.11 includes the DCE prerequisite.

©2004 Hewlett-Packard Development Company, L.P.

DCE Software Requirements for Syslog

NOTE Installation of DCE software prerequisites are required only for HP-UX 11.0 operating systems. On Solaris operating systems, the install process automatically installs HP's lightweight DCE if a supported DCE is not found.

Part of the syslog configuration process includes installing an HP OpenView Operations (OVO) agent on the NNM management station. The OVO agent requires two pieces of software to be installed prior to configuring syslog: DCE RPC and DCE-KT-Tools. See the *Release Notes* for more information about supported software versions.

The required DCE software is available on the HP-UX Application Software CD-ROMs. To install the required DCE software, do the following:

1. Invoke the `SD Install` interface by typing: `swinstall`
2. Change the software view by clicking: `View:Change Software View->Start with Products`.
3. Install the first DCE software package by selecting `DCE-Core.DCE-CORE-RUN` and clicking `Actions:Install`.

4. Install the remaining DCE software package by selecting: `DCE-KT-Tools` and clicking `Actions:Install`.

To check whether you have properly installed the required DCE software, do the following:

1. Type: `swlist | grep DCE`
2. Look for two items in the list:
   - DCE/9000 Programming and Administration Tools
   - DCE/9000 Kernel Threads Support

Configuration Overview

Configuration Overview

- Configure incoming conditions to match.
- Configure trap OIDs and varbinds to emit.
- Deploy the configuration to the running system.

©2004 Hewlett-Packard Development Company, L.P.

There are two primary interactions:

- Configuring specific syslog message patterns to be mapped to SNMP traps

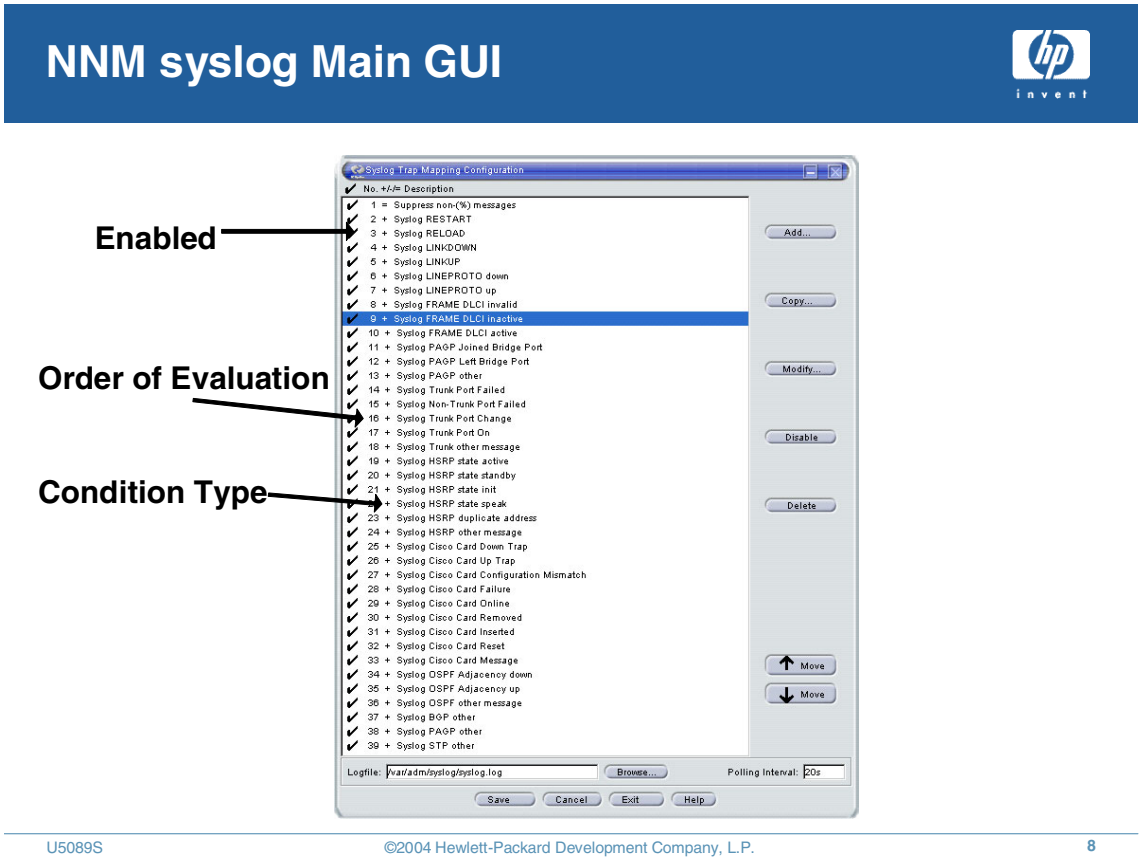
  To configure new syslog message patterns, modify, reorder, or delete existing patterns, run `ovsyslogcfg`. After editing the configuration, back up the previous version and deploy the new configuration by running `setupSyslog.ovpl -deploy`. This option generates and encrypts the new template and restarts the OVO agent to reload the new template.

- Enabling/deploying a syslog configuration

  Enabling and deploying any syslog configuration is done via the command line script `setupSyslog.ovpl`. When NNM is installed, the syslog feature is not enabled by default. `setupSyslog.ovpl` needs to be run in order to start the embedded OVO agent and the SNMP mapping daemon, `syslogTrap`.

Refer to your Cisco documentation on how to use the `logging <IPAddr>` or `set logging <IPAddr>` to direct syslog messages to the management station.

NNM syslog Main GUI



 ©2004 Hewlett-Packard Development Company, L.P.

This window lists message and suppress conditions for Syslog Integration message source template in the order they are compared to incoming messages. Message Conditions are filters that accept messages matching a defined pattern and set of attributes. Suppress Conditions are filters that reject messages matching a defined pattern and set of attributes.

NNM Syslog Trap Mapping Configuration Interface

When the NNM standalone configuration option is deployed, you construct and modify Syslog Integration message source template conditions through the Syslog Trap Mapping Configuration interface.

To launch the Syslog Trap Mapping Configuration interface, execute:

```
$OV_BIN/ovsyslogcfg
```

The main dialog for the Syslog Trap Mapping Configuration interface supports the following actions:

- Reorder conditions by selecting a condition and clicking [Move].
- Add or delete a condition by selecting a condition and clicking either [Add] or [Delete].

- Copy a condition by selecting a condition and clicking [Copy].
- Change the pattern matching rules of a condition by selecting a condition and clicking [Modify].
- Enable or disable a condition by selecting a condition and clicking [Enable] or [Disable].
- Change the location of the Syslog Integration logfile by clicking [Browse].
- Change the polling interval time (how often the logfile encapsulator parses the management station's syslog file to look for new messages). By default, this is 20 seconds.
- Save your changes by clicking [Save].

The fields of the Syslog Trap Mapping Configuration dialog are:

Field	Description
Check mark symbol	Specifies whether the condition is enabled or disabled.
No.	Specifies the order of the message and suppress conditions. Incoming messages are compared with message and suppress conditions in the order that the conditions are listed in this window. In general, if it is more important to filter into messages that match a given condition, order message condition(s) before suppress conditions. Conversely, if it is more important to filter out messages that match a given condition, order suppress condition(s) before message conditions. It is recommended to list more detailed message conditions before more general message conditions, and more general suppress conditions before more detailed suppress conditions. Changing the order of the conditions can affect the performance.
+\-\=	Identifies whether the condition is a message condition or a suppress condition. Message conditions are identified by a "+" character. Suppress conditions are identified by "-" and "=" characters. "-" means suppress matched conditions. "=" means suppress unmatched conditions.
Description	Provides a short text description for the condition.

For instructions on how to use the Syslog Trap Mapping Configuration interface, see the *Syslog Trap Mapping Configuration Online Help*.

Extending the feature to add a new template(s) and set of traps is supported in the OVO environment. This type of development is expected to require in-depth knowledge of OVO.

To avoid any conflicts with OVO templates, the syslog template is delivered into a subproduct location (NNMSyslog).

Overview of Message Source Templates

OVO agents are configured via message source templates. The OVO agent can only format and forward a message that is described in a message source template.

In the NNM standalone configuration, the OVO agent is embedded on the NNM management station. In the OVO with NNM configuration, the OVO agent coexists on the NNM management

station. In either case, the OVO agent is configured to monitor the status of and collect information from syslog messages through the `Syslog to NNM` message source template.

Message source templates work by identifying strings within messages in message streams. When messages match the conditions defined in the message source templates, they are processed according to the rules defined in the template. When the Syslog Integration functionality is enabled, messages matching markers defined in the `Syslog to NNM` template are forwarded to the NNM `syslogTrap` process. This process maps the syslog messages to SNMP traps.

Message source templates consist of the following elements:

- Type of message source from which you want to collect messages, such as a logfile, a trap, an OVO message interface, or an action. In the case of the `Syslog to NNM` template, the message source is a logfile.
- Message conditions and suppress conditions that match a set of attributes and define responses to received messages. These conditions filter incoming messages from the message source. The conditions also determine how the “important” messages are displayed in the operator window.
- Options, such as default message logging.

Extract Patterns

Extract Patterns

- Used in NNM and OVO
- Parse an input string to extract tagged variables
- Similar to regular expressions
- These tags appear as sub-variables to the assigned variable, and can be used like any other variables.

©2004 Hewlett-Packard Development Company, L.P.

Event attribute values can be extracted and used inside a Composer correlator or syslog trap definition.

Pattern Matching

- Special characters
  - ^ anchors to beginning of line
  - $ anchors to end of line
  - | Or operator allows a string to match one of two possibilities
  - \ mask or disable special meaning of special character and treat as itself
  - < > enclose a pattern to match
- Special sequences are
  - <#> for a number
  - <@> for a word
  - <S> or <_> for whitespace (note: Composer only uses <S>)
  - <*> to match anything

 ©2004 Hewlett-Packard Development Company, L.P.

Pattern-Matching

ECS provides a powerful text pattern-matching language that allows logical testing for the existence of substrings and patterns. Parts of a text string can be extracted and assigned to tags, which may be reused within the same scope. This section describes the operators and syntax of the pattern-matching language.

The pattern-matching language used in the match functions is the same as that used in HP OpenView Operations.

Frequently, pattern-matching means simply scanning for a specific substring in the target string. For example, to search for the substring ERROR anywhere in the target string you search for the pattern:

```
"ERROR"
```

Similarly, should you wish to match text not containing a specific substring (for example, WARNING), you type:

```
"<![WARNING]>"
```

This uses the not operator “!”, together with the chevrons “< >” that must enclose all operators,

and the square brackets “[]” that isolate sub-patterns.

You control case-sensitivity with a separate argument to the Match.make function.

Defining Match Expressions

- Ordinary Characters

  Ordinary characters generally represent themselves. However, if any of the following special characters are used they must be prefaced with a backslash escape character (\ ) to mask their usual function.

  [ ] < > | ^ $

- Expression Anchoring Characters (^ and $)

  If the caret (^) is used as the first character of the pattern, only expressions discovered at the beginning of lines are matched. For example, “^ab” matches the string “ab” in the line “abcde”, but not in the line “xabcde”.

  If the dollar sign is used as the last character of a pattern, only expressions at the end of lines are matched. For example, “de$” matches “de” in the line “abcde”, but not in the string “abcdex”.

  If ^ and $ are not used as anchoring characters, that is, not as first or last characters, they are considered as ordinary characters without masking.

- Expressions Matching Multiple Characters

  Patterns used to match strings consisting of an arbitrary number of characters require one or more of the following expressions:

  • <*> matches any string of zero or more characters (including separators)

  — <n*> matches a string of n arbitrary characters (including separators)

  • <#> matches a sequence of one or more digits

  • <n#> matches a number composed of n digits

  • <S> or <_> matches a sequence of one or more separator or whitespace characters. <_> is the preferred syntax.

  • <nS> matches a string of n separators

  • <@> matches any string that contains no separator characters, in other words, a sequence of one or more non-separators; this can be used for matching words.

  Separator characters are configurable for each pattern. By default, separators are the space and the tab characters. The separator string is specified as the second element in the 3-tuple passed to the Match.make function.

- Bracket ([and]) Expressions

  The brackets ([and]) are used as delimiters to group expressions. To increase performance, brackets should be avoided wherever they are superfluous. In the pattern:

  “ab[cd[ef]gh]”

  all brackets are unnecessary—"abcdefgh" is equivalent.

  Bracketed expressions are used frequently with the OR operator “|”, the NOT operator “!” and when using sub-patterns to assign strings to tags.

- The OR (|) Operator

  Two expressions separated by the vertical bar character "|" matches a string that is matched by either expression. For example, the pattern:

  "[ab|c]d"

  matches the string "abc" and the string "cd".

- The NOT (!) Operator

  The not operator "!" must be used with delimiting square brackets, for example:

  "<![WARNING]>"

  The pattern above matches all text which does not contain the string "WARNING".

  The not operator may also be used with complex sub-patterns:

  "LN<*>: R< ![490|[501[a|b]]] >-<*>"

  The above pattern makes it possible to generate a message for any line connection other than from repeaters 490, 501a or 501b.

  Therefore, the following would be matched:

  "LN270: R300-427"

  However, this string is not matched, because it refers to repeater 501a:

  "LN270: R501a-800"

  If the sub-pattern including the not operator does not find a match, the not operator behaves like a <*>: it matches zero or more arbitrary characters. For this reason, there is a difference between the UNIX expression "[!123]", and the corresponding ECS pattern matching expression: "<![1|2|3]>". The ECS expression matches any character or any number of characters, except 1, 2, or 3; the UNIX expression matches any one character, except 1, 2, or 3.

- The Mask (\ ) Operator

  The backslash "\" is used to mask the special meaning of the characters:

  [ ] < > | ^ $

  A special character preceded by \ results in an expression that matches the special character itself.

  Because ^ and $ only have special meaning when placed at the beginning and end of a pattern respectively, you need not mask them when they are used within the pattern (in other words, not at beginning or end). The only exception to this rule is the tab character, which is specified by entering "\t" into the pattern string.

Extract Variable Assignment

Extract Variable Assignment

- Use tags to define sub-variables.
  - Match attempts to move from left to right to achieve success if possible.
  - <@>, <#>, <S> and <_> match as many characters as possible.
  - <*> matches as few characters as possible.
  - <*> at the start/end of the pattern takes the start/end of the line.

- Example: To extract the card and port numbers from a message such as
  Card = 10 : Port = 1

```
Card<_>=<_><*.Card><_>:<_>Port<_>=<_><*.Port>
```

- Assigns 10 to Card
  1 to Port

U5089S ©2004 Hewlett-Packard Development Company, L.P. 11

Tags

Search patterns may use tags to identify part(s) of the target string to, for example, compose a new string from selected parts of the target string. To define a tag, add ".tagname" before the closing chevron. The pattern:

```
^errno: <#.number> - <*.error_text>
```

matches a string such as:

```
errno: 125 - device not in service
```

and assigns "125" to the tag number and "device not in service" to the tag `error_text`. The tags may be accessed as members of a dictionary. See the *HP OpenView Correlation Composer's Guide*.

Assignment Rules

In matching the pattern "<*.tag1><*.tag2>" against the string "abcdef", it is not immediately clear which substring of the input string is assigned to each tag. For example, it is possible to

assign an empty string to tag1 and the whole input string to tag2, as well as assigning “a” to tag1 and “bcdef” to tag2, and so forth.

The pattern-matching algorithm always scans both the input line and the pattern definition (including alternative expressions) from left to right. <*> expressions are assigned as few characters as possible. <#>, <@>, <S> expressions are assigned as many characters as possible.

Therefore, tag1 will be assigned an empty string in the above example. To match an input string such as:

```
"this is error 100: big problem"
```

use a pattern such as:

```
error <#.errnumber>:<*.errtext>
```

In which:

- “100” is assigned to the tag `errnumber`.
- “big problem” is assigned to the tag `errtext`.

For performance and pattern readability purposes, you can specify a delimiting substring between two expressions. In the above example, “:” is used to delimit <#> and <*>.

Matching <@.word><#.num> against “abc123” assigns “abc12” to `word` and “3” to `num`, as digits are permitted for both <#> and <@>, and the left expression takes as many characters as possible.

Patterns without expression anchoring can match any substring within the input line. Therefore, patterns such as:

```
"this is number<#.num>"
```

are treated in the same way as:

```
"<*>this is number<#.num><*>"
```

Sub-Patterns Assignment

In addition to being able to use a single operator, such as * or #, to assign a string to a tag, you can also build up a complex sub-pattern composed of a number of operators, according to the following pattern:

```
<[sub-pattern].tag>
```

For instance: <[rack<#>.brd<#>].hware>

In the example above, the period (.) between rack<#> and brd<#> matches a similar dot character, while the dot between] and hware is necessary syntax. This pattern would match a string such as “rack123.brd47” and assigns the complete string to hware.

Other examples of sub-patterns are:

```
<[Error|Warning].sev>
```

and

```
<[Error[<#.n><*.msg>]].complete>
```

In the first example above, any line with either the word “Error” or the word “Warning” is assigned to the tag, `sev`. In the second example, any line containing the word “Error” has the error number assigned to the tag, n, and any further text assigned to msg. Finally, both number and text are assigned to complete.

Extract Example

Extract Example

- A print spooler sends a message containing the following string:

  JobID=345;Target=ljet1;Prio=7;Model=Laserjet 5 MX; Status=TonerLow;Error=37

- Extract the status of the printer to see whether it is “Normal.”

- Simplest:

  <*>Status=<*.status>;<*>

- Most Versatile:

  JobID=<*.jobid>;Target=<*.target>;Prio=<*.priority>;<*>; Status=<*.status>;Error=<*.errornum>

©2004 Hewlett-Packard Development Company, L.P.

Testing Extract Patterns

- Test the syntax of extract patterns using **opcpat**

You can verify the syntax of any pattern and test the patterns by executing:

```
opcpat
```

Read the man page for opcpat for instructions on how to use the command.

```
opcpat [ -h ] [ -i ] [ -q ] [ -fp <patternfile> ] [ -fv <valuefile> ] [ -o
<outfile> ]
```

The command opcpat is used for testing NNM and OVO pattern matching routines. Both interactive and automatic tests using input files are supported.

A pattern is matched against each value line. If the pattern matched opcpat displays the substrings replaced by the <*>, <#>, ... subpatterns together with their corresponding parameter name (if present).

opcpat expects a certain format for the input files specified in the command line. Each file starts with an arbitrary number of comment lines introduced by #. The pattern input file consists of a sequence of pattern-separator pairs:

Patternfile:

```
# Commentlines
Pattern1
Separators1
```

```
Pattern2
Separators2 ...
```

Values are expected each in one input line in the value input file.

```
# Commentlines
Valuefile:
Value_a
Value_b
Value_c ...
```

Options

-h Print usage message of opcpat.

-i Use case insensitive pattern matching (default is case sensitive matching).

-q Use quiet mode; only prompts for user input will be sent to stdout.

-fp <patternfile> Name of the input file containing the patterns and separators. By default opcpat prompts for input of the patterns and separators.

-fv <valuefile> If this option is specified opcpat reads the values from file <valuefile>. By default opcpat prompts for the values on stdout and reads the lines from stdin.

Example

Here is a sample value file:

```
# more /tmp/val1
123456
```

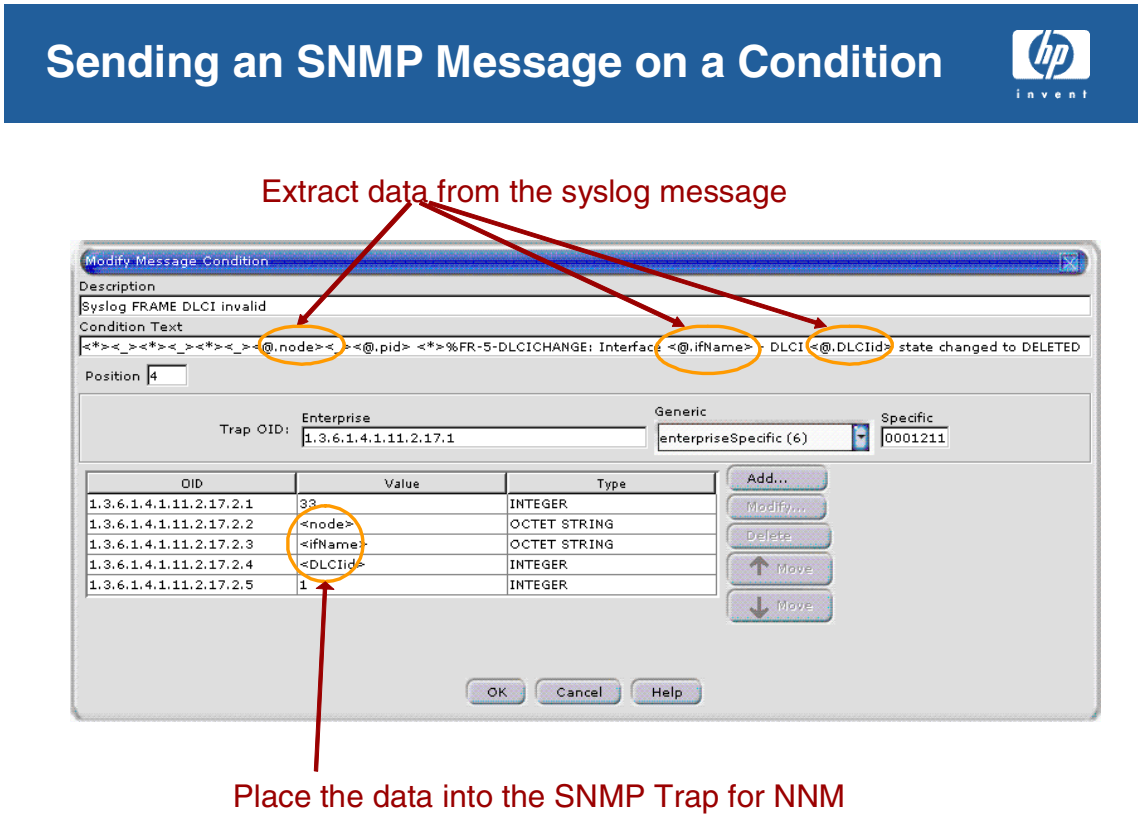
Here is a sample pattern file:

```
# more /tmp/pat1
<@.var1><@.var2>
```

The output from opcpat would be:

```
# opcpat -fp /tmp/pat1 -fv /tmp/val1
using single-byte mode
******** next pattern ********
Pattern: "<@.var1><@.var2>" using seps ""
Value: "123456"
"" var1:"12345" var2:"6" ""
```

Sending an SNMP Message on a Condition

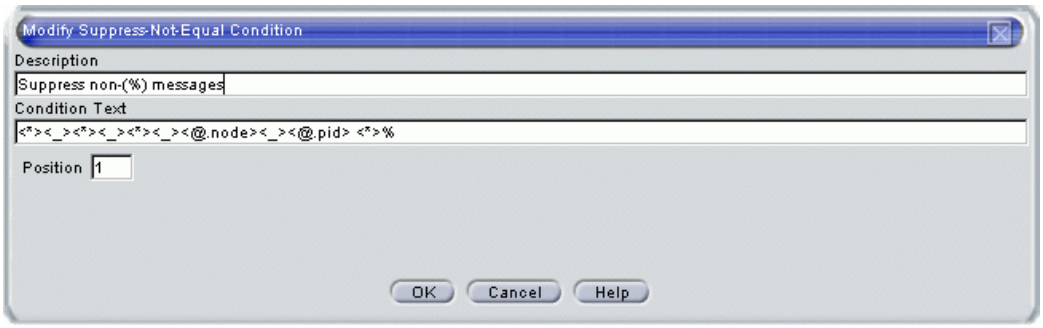


The primary customization of syslog is to add or modify a message condition. A message condition consists of:

- A condition text field that is similar to a regular expression. This field identifies the pattern to match and names any subexpression in the pattern to be used to map into the SNMP trap.
- A set of message attributes that define the SNMP trap to be generated from this type of message. The trap OID is defined by the enterprise, generic and specific fields. The varbinds of the trap are defined in the lower table. You can edit the condition text and the trap OID fields. You can also delete and reorder of any of the varbinds.

See the *NNM Syslog Trap Mapping Configuration Online Help* for more information about constructing pattern matching syntax.

Suppressing a syslog Pattern



 ©2004 Hewlett-Packard Development Company, L.P.

Suppression patterns are the most effective way to optimize the performance of the syslog parser.

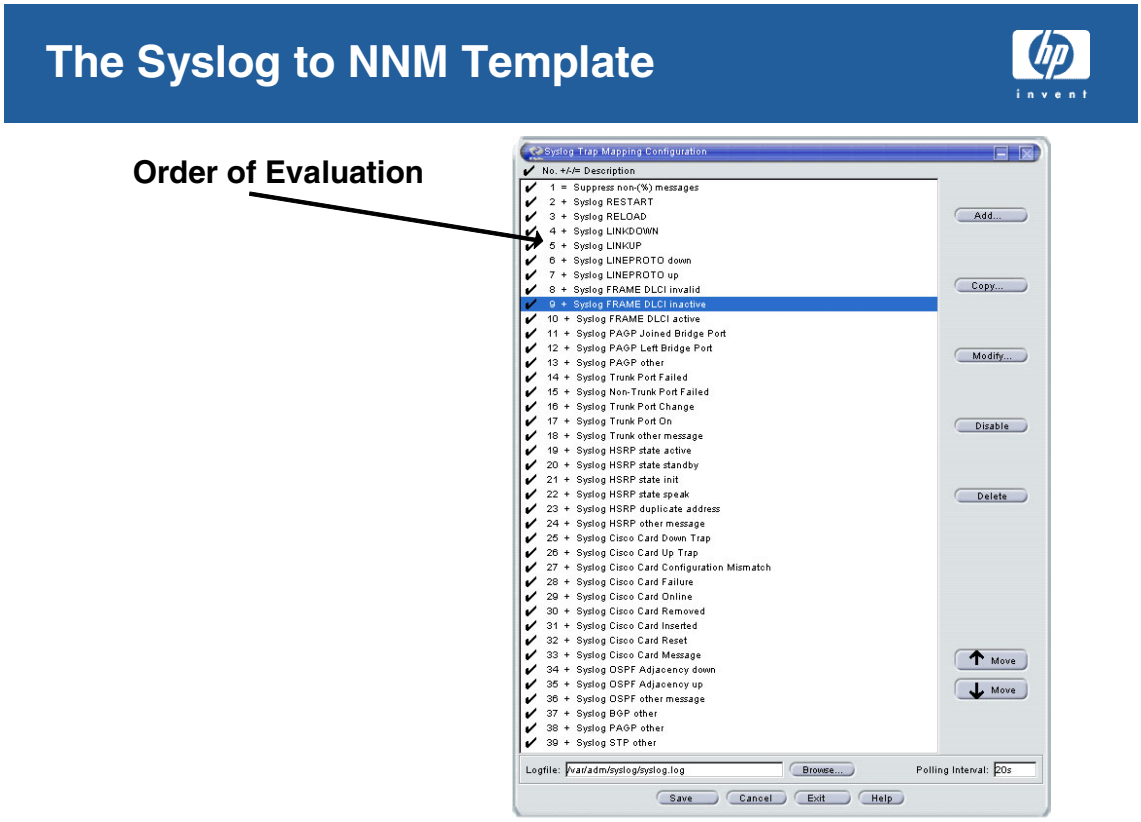
Two types of suppression patterns are supported:

- Suppress if not equal (-)
- Suppress on equal (=)

This screen shot shows an example of a suppress-not-equal condition. The pattern will exclude any message that does not conform to the Cisco syslog format (% as the leading non-whitespace character in the message portion). This pattern does not functionally perform anything but is significant as an optimization because all the other conditions will only be executed on Cisco syslog messages.

See the *NNM Syslog Trap Mapping Configuration Online Help* for more information about constructing pattern matching syntax.

The Syslog to NNM Template



When Syslog Integration functionality is enabled, the Syslog for NNM template is placed in a template configuration directory on the NNM management station. For NNM standalone configurations, the Syslog to NNM template is automatically uploaded to the embedded OVO agent when Syslog Integration is enabled.

NNM includes out-of-the-box template conditions for which syslog messages are mapped to OpenView SNMP traps. Each type of syslog message to be mapped is defined in one template condition. The conditions are contained in the `Syslog to NNM` template.

In both the OVO template editor windows and the NNM Syslog Trap Mapping Configuration interface, the order of the conditions is important for pattern matching. The patterns are tested in the order that they are listed, and the first pattern to match is executed.

NOTE Since ordering of template conditions matters, it is important to place suppression patterns first and more specific patterns at the beginning of the list. More general patterns should go last.

The first condition is a suppress unmatched condition, meaning the pattern will exclude any message that does not conform to its pattern. In this case, the pattern matches only those syslog messages with the % character as the leading non-white space character in the message (specifically, Cisco syslog message types). This pattern does not functionally perform anything, but is significant as an optimization tool, since all other conditions in this template will execute only on Cisco syslog messages.

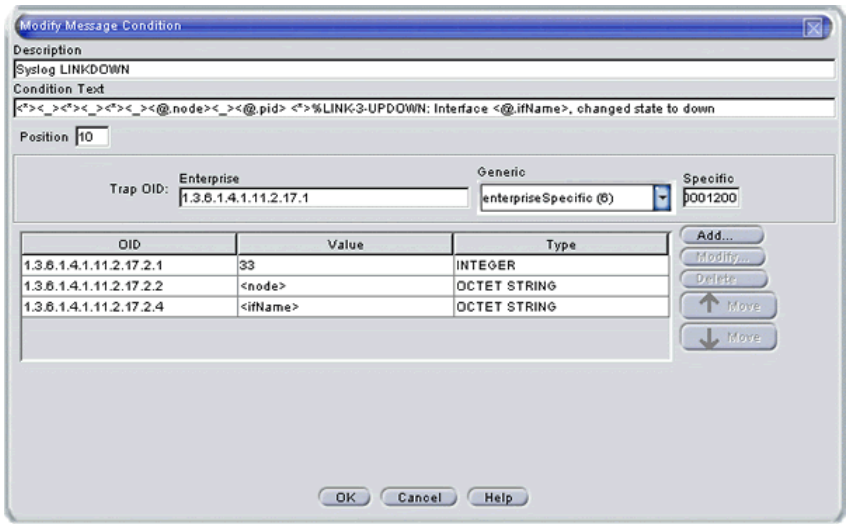
The remaining conditions look for Cisco syslog messages matching defined patterns as identified in Table 21-2.

Table 21-2 Template Conditions and Corresponding Syslog Messages

Template Condition Name	Syslog Message Format
Syslog LINEPROTO down	%LINEPROTO-5-UPDOWN (down)
Syslog LINEPROTO up	%LINEPROTO-5-UPDOWN (up)
Syslog FRAME DLCI Invalid	%FR-5-DLCICHANGE (INACTIVE)
Syslog FRAME DLCI Inactive	%FR-5-DLCICHANGE (INACTIVE)
Syslog FRAME DLCI Active	%FR-5-DLCICHANGE (ACTIVE)
Syslog OSPF Adjacency up	%OSPF-5-ADJCHG (UP)
Syslog OSPF Adjacency down	%OSPF-5-ADJCHG (DOWN)
Syslog LINKUP	%LINK-3-UPDOWN (up)
Syslog LINKDOWN	%LINK-3-UPDOWN (down)

Modifying a Condition

Modifying a Condition



 ©2004 Hewlett-Packard Development Company, L.P.

Messages matching the pattern defined in the `Condition Text` field cause the OpenView event identified by the `Trap OID` to be generated. For example, when a `%LINK-3-UPDOWN` status `DOWN` message is logged to the syslog file, the message is intercepted, since the Syslog LINKDOWN condition looks for this pattern. In that same condition, a trap OID is identified, which corresponds to the OpenView event. This event is then generated.

Select a condition and click [Modify] to modify a `Syslog to NNM` template condition definition. The window is divided into two logical parts: a condition text field to match patterns and a set of attributes that define the SNMP trap to be generated.

The `Condition Text` field uses syntax similar to a regular expression. It identifies the pattern to match and names any subexpressions in the pattern to be used in the mapping to an SNMP trap. See the *NNM Syslog Trap Mapping Configuration Online Help* for more information about pattern matching.

The `Position` field identifies the location of the condition with respect to the other conditions of the template.

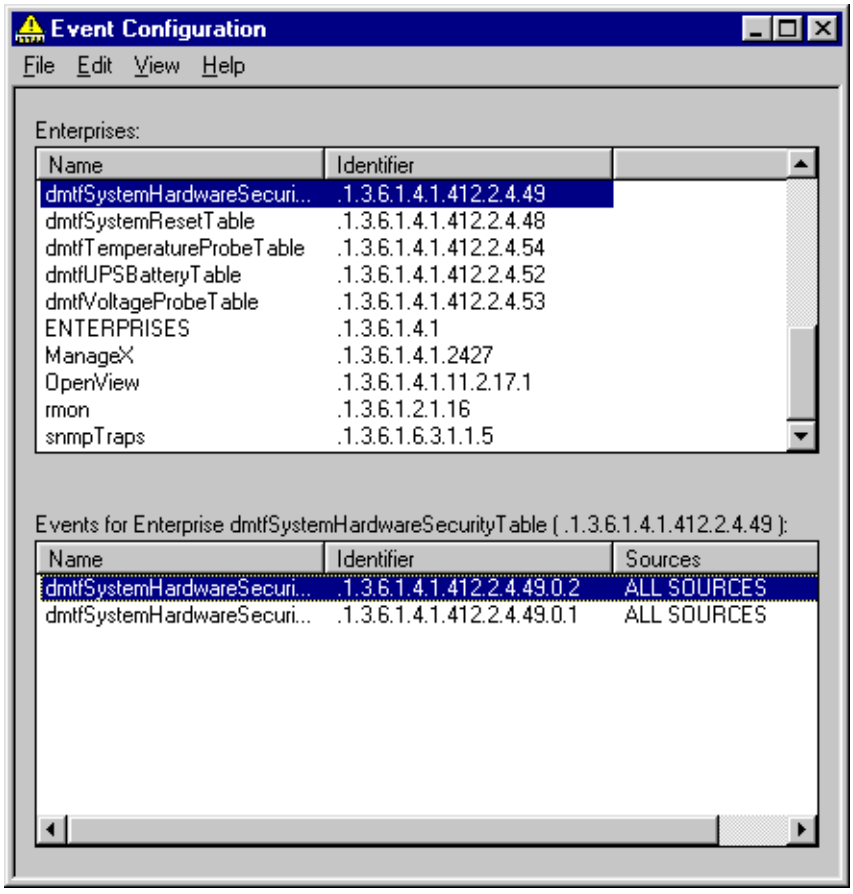
The `Trap OID` is defined by the enterprise, generic, and specific fields. The trap OID is used to determine the type of OpenView event to be generated in response to a message matching the condition pattern.

The varbinds of the trap are defined in the lower table.

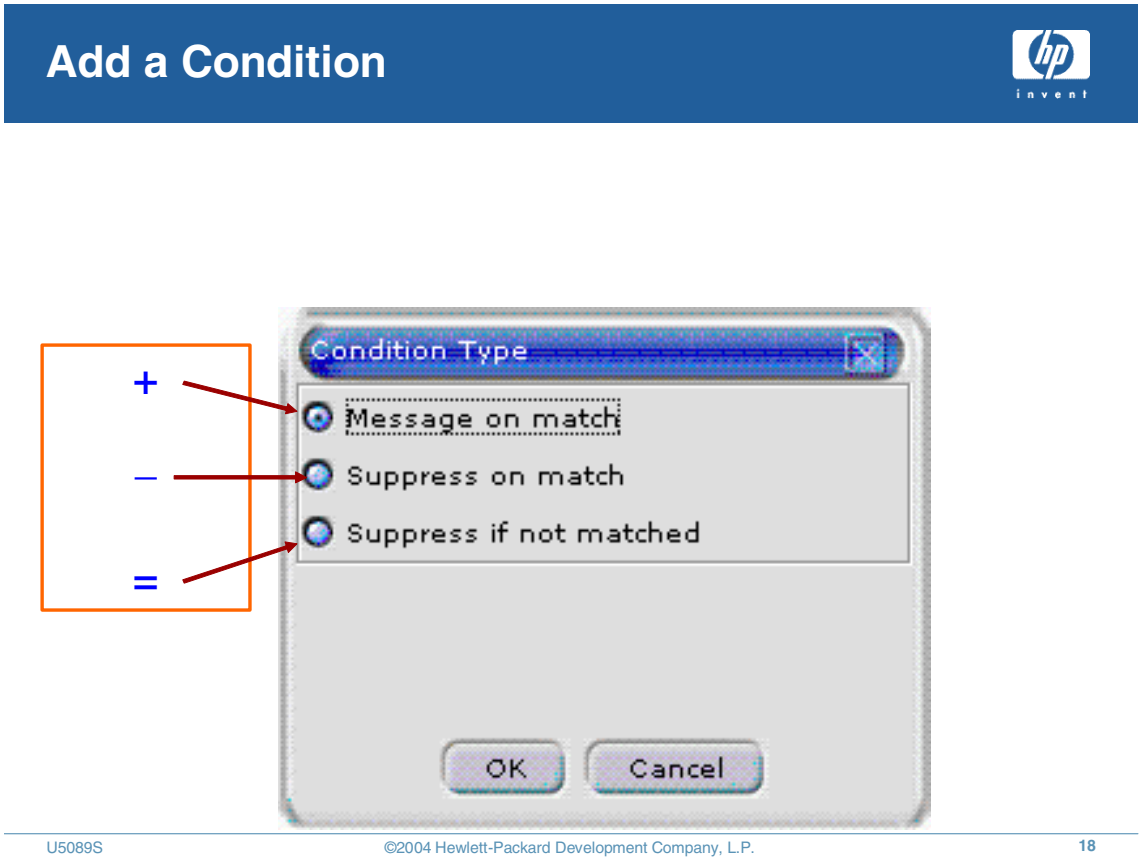
You can edit the `Condition Text` and the `Trap OID` fields. You can also modify and reorder any of the varbinds. See the *NNM Syslog Trap Mapping Configuration Online Help* for more information about modifying these fields.

To view or identify the corresponding OpenView event to be generated, do the following:

1. Start NNM by typing: **ovw**
2. From the `Root` window, click **`Options: Event Configuration`**.
3. Select `OpenView` from the `Enterprise Name` list. A list of OpenView events displays in the bottom pane.
4. Locate the trap OID from the `Event Identifier` list.
5. Double-click the event or click **`Edit:Modify Event`** to display the Event Configurator/Modify Event window.



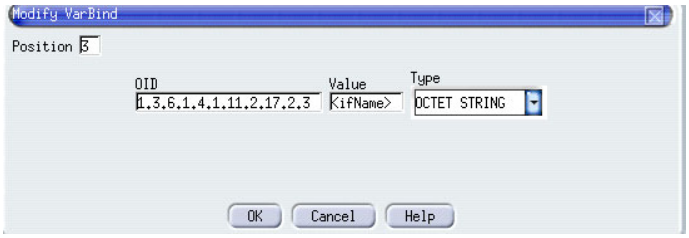
Add a Condition



When you click [OK], the Add Condition detail screen looks like the Modify Message Condition screen.

Adding and Modifying Varbinds

Modifying varbinds



©2004 Hewlett-Packard Development Company, L.P.

In the Modify Message Condition dialog box click [Add] to add a varbind or select a varbind and click [Modify] to change it. You can also double-click a varbind to modify it.

The above screen is for modifying or adding a varbind setting to a message condition. For each varbind the OID, value and type must be specified.

If the value is a named subexpression from the condition text it must be enclosed in <>. Constant values and named condition text subexpressions are the only types allowed for varbind values.

Varbind 1: 33 is the ID for syslogTrap. This tells support where the message came from.

Varbind 2 is usually the source node.

Note that varbind 3 is not used. All other varbinds are 4 (repeat as needed).

Deploying syslog Mappings

- Enable the syslog Integration software

  setupSyslog.ovpl –standalone

- Deploy configured conditions to the running system

  setupSyslog.ovpl –standalone -deploy

©2004 Hewlett-Packard Development Company, L.P.

Enabling Syslog Integration

NOTE When you execute the `ovstatus` command, a background process called `syslogTrap` is listed. Before enabling the Syslog Integration functionality, this process displays as NOT RUNNING. Do not attempt to start this process before enabling the Syslog Integration functionality.

To enable Syslog Integration for NNM standalone configurations, execute the following command on the NNM management server: **`setupSyslog.ovpl -standalone`**

Use the `-help` option to display a help message for the `setupSyslog.ovpl` command options.

This command does the following on your NNM management station:

- Deploy the out-of-the-box syslog template, `Syslog to NNM`.
- Activate the embedded OVO agent.
- Activate and register the SNMP mapping background process, `syslogTrap`.

Deploying `Syslog to NNM` Template

After editing syslog message source template conditions with the Syslog Trap Mapping Configuration interface, execute

```
setupSyslog.ovpl -standalone -deploy
```

to deploy the new configuration.

The `-deploy` command option generates and encrypts the new template and restarts the embedded OVO agent so that the new template is reloaded.

The options support by `setupSyslog.ovpl` are:

- `standalone`: Activate the syslog processes in a standalone (without OVO) environment.
- `server`: Activate the syslog processes in an environment where OVO is installed with NNM.
- `deploy`: Generate and encrypt a new template from the configuration file `syslogtrap.xml` and restart the syslog processes to reload the new template.
- `disable`: Stop the syslog processes.
- `help`: Display a help message for the `setupSyslog.ovpl` command options.

Testing Syslog Monitoring

Testing syslog Mappings

- Test the syntax of template conditions using **opcpat**
- **logger** sends test messages to syslogd on the management station

```
logger %LINEPROTO-5-UPDOWN: Line Protocol on Interface interface2, changed state to down
```

- If you want to control the timestamp, PID, and hostname in the message, **echo** the string into the syslog.log file.
- syslog entries are logged to **/var/adm/syslog/syslog.log**.

U5089S ©2004 Hewlett-Packard Development Company, L.P. 21

Testing Patterns in Template Conditions

Before you redeploy the `Syslog to NNM` template, you can verify the syntax of any template condition and test the patterns by executing:

```
opcpat
```

Read the man page for `opcpat` for instructions on how to use the command.

Sending Sample syslog Messages to the System

Use the UNIX command line tool, `logger`, to write test messages to the system logfile. Read its man page for more information on how to use the command.

For example, to create a *Line Protocol status Down* syslog entry, do the following:

HP-UX: `logger %LINEPROTO-5-UPDOWN: Line protocol on Interface interface2, changed state to down`

Solaris: `logger -p user.err %LINEPROTO-5-UPDOWN: Line protocol on Interface interface2, changed state to down`

System Logfiles

On HP-UX operating systems, syslog entries are logged to `/var/adm/syslog/syslog.log`.

On Solaris operating systems, syslog entries are logged to `/var/adm/messages/syslog.log`.

The Solaris template assumes the correct logfile location.

syslog Traps and Overlapping IP Addresses

Syslog and Overlapping IP Addresses

- The syslogTrap process places the OAD ID in the event.
- Composer supports inclusion of OAD in hostname queries.

©2004 Hewlett-Packard Development Company, L.P.

Troubleshooting Tips

- Use **opcpat** to verify patterns and CMA bindings.
- Not intended for very high volume syslog messages.
- Enable the integration before starting the syslogTrap process.
- Duplicate syslog messages
- Missing syslog messages

 ©2004 Hewlett-Packard Development Company, L.P.

Here are some troubleshooting tips for the Syslog Integration functionality.

Performance

The Syslog Integration functionality is not intended for high volume syslog message systems.

Some performance issues may arise as the syslog messages from the managed network elements can become extremely abundant. Sufficient tuning of the `Syslog to NNM` template conditions may need to be done for exclusion patterns to improve performance. Additionally, you may need to add some filtering mechanism to the NNM background process (`syslogTrap`) that maps the syslog message to an SNMP trap.

Configuration

Error starting syslogTrap process:

You must have the Syslog Integration functionality enabled before starting the `syslogTrap` background process.

To start the `syslogTrap` process, execute:
`ovstart syslogTrap`

Seeing Duplicate Syslog Messages in Message Browser:

This could be caused by a number of reasons, including one of the following:

- In OVO with NNM configurations, you must enable the message stream interface for both the `Syslog to NNM` template and the OVO agent on the NNM management station in order for syslog messages to be processed as documented. However, in OVO, there are a multitude of combinations for diverting messages through the system. For example, you can enable the message stream interface for individual conditions of a template to copy messages as well as enabling the message stream interface for the template to copy messages. This will produce multiple messages in the message browser.
- Templates are not ordered, meaning that if messages match conditions of multiple templates, multiple messages are displayed in the message browser. For example, if a wildcard template is assigned and installed on a system, then every message entering the agent is forwarded to the message browser. Furthermore, if additional templates are assigned and installed on a system, then those messages matching the conditions of the templates are also forwarded to the message browser. Thus, duplicate messages appear in the message browser, formatted according to rules in the templates.

Not Seeing Syslog Messages in Message Browser

This could be caused by many reasons, including one of the following:

- The `Syslog to NNM` template is not installed or enabled on the OVO agent system (on the NNM management station). To verify that the `Syslog to NNM` template is installed and enabled on the OVO agent, execute:
  `$OV_BIN/OpC/opctemplate`

  This command lists all templates with the type, name, and status (enabled or disabled). This command is helpful to check whether a template you have assigned to an agent node has successfully been installed on that agent system. Be aware that this command does not indicate which version of the template has been deployed. If you have made modifications to any assigned templates, you must reinstall the templates on the managed nodes.
- For OVO with NNM configurations, the message stream interface is not enabled for either the `Syslog to NNM` template or the OVO agent on the NNM management station.

  To isolate the problem, you can turn on XPL tracing for the `syslogTrap` process. If you see no activity in incoming messages, it usually means that the message stream interface has not been enabled in all places that must be enabled.

Verifying installation

- Check `$OV_PRIV_LOG/setupSyslog.install` for errors. This file contains the swinstall output. Also check `/var/adm/sw/swagent.log` for errors.
- Check `$OV_PRIV_LOG/setupSyslog.log` for errors. This file logs the `setupSyslog.ovpl` script progress. Look for any steps that had errors (non-zero results).

Verifying healthy operation

- Use `opcagt -status` to verify that `opcctla`, `opcmsga`, and `opcle` are running.
- Verify `syslogTrap` is running with `ovstatus`.
- Verify system is logging syslog messages in syslog file, typically `/var/adm/syslog/syslog.log` for HPUX and `/var/adm/messages` for Solaris.
- `syslogd` daemon may not be running.
- `syslogd` configuration may be excluding that type or severity of message.
- Verify that specific syslog messages are being logged that match the `syslogTrap` template conditions, for example,

```
%FR-5-DLCICHANGE: Interface Serial0 - DLCI 40 state changed to ACTIVE

%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0.2, changed state to up
```

- Verify that `syslogTrap` is receiving the OVO messages.
- `syslogTrap` uses XPL tracing.
- Monitor `syslogTrap` application in `tracemon`.
- Look for syslog messages being received and events being created for output.
- If events are output from `syslogTrap`, you should see them logged in the BES by running `ovdumpevents`.

Removing Syslog Integration

- Disable syslog Integration with
  `setupSyslog.ovpl -standalone -disable`
- **`swremove`** the agent.

©2004 Hewlett-Packard Development Company, L.P.

Disabling Syslog Integration Functionality

To disable the syslog functionality, execute:

```
setupSyslog.ovpl -standalone -disable
```

This command stops the embedded OVO agent processes and the NNM `syslogTrap` process. The OVO agent software remains on the NNM management station.

You can re-enable the Syslog Integration functionality, by executing: `setupSyslog.ovpl -standalone`

Removing syslog Integration

Removing Network Node Manager from the system does not completely remove Syslog Integration. The OVO agent is left enabled and running on the system.

To remove the remaining Syslog Integration components, do the following:

1. Disable the Syslog Integration feature, by executing the following command:

   For NNM standalone configurations, type:
   `setupSyslog.ovpl -standalone - disable`

   For OVO with NNM configurations, type:
   `setupSyslog.ovpl -server -disable`

2. For NNM standalone configurations, remove the OVO agent software from the NNM management station by doing the following:

   a. Type: `swremove` (HP-UX) or `pkgrm` (Solaris)

      This opens the SD Remove window.

   b. Select the `ITOAgent` software package name from the Name list.

   c. Click `Actions:Remove` to remove the OVO agent from the NNM management station.

NOTE Currently the NNM `remove.nnm` script does not remove the OVO agent. If NNM is removed before running '`setupSyslog.ovpl –standalone –disable`', the ITOAgent fileset will have to be removed manually using `swremove`.

Lab Exercises

- Start the syslog configuration interface
- Review existing templates
- Create and test a new condition

©2004 Hewlett-Packard Development Company, L.P.

These exercises are designed to assist you in understanding the operation of the syslogTrap component of NNM.

1. Enable syslog in a stand-alone NNM environment (no OVO).

2. The `$OV_CONTRIB/OVTraining/NNM3` directory contains a file of new patterns (`patterns`). You also have a sample data input line in the file `value`. See how this value is parsed by the various patterns by using `opcpat`.

3. Launch the syslog configuration GUI and browse the default conditions.

4. Test by injecting a `linkdown` syslog message which emits a trap.

5. Using the Cisco web-site for syslog messages: http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/sem/emabout.htm, map SYS-5-CONFIG to a new trap.

6. Deploy your configuration.

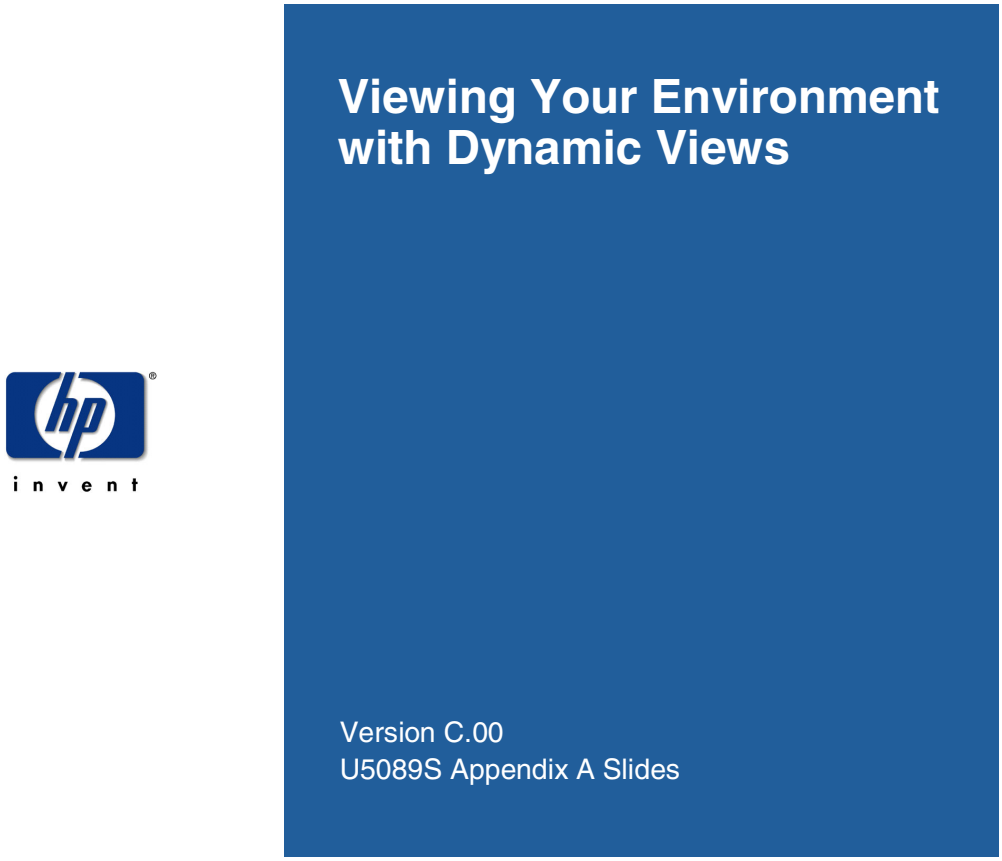
7. Use `logger` to test your pattern and watch for the result in the Alarm Browser.

8. Use `Options:Event Configuration` to recognize event 2001 and display the node name where the failure occurred.

9. Retest your event using `logger`.

A Viewing Your Environment with Dynamic Views

Module Objectives



At the completion of this module the student will be able to:

- Interpret visual cues used in Dynamic Views.
- Access Dynamic Views from Home Base, the Alarm Browser, Tools menu, and the web.
- Change the labels displayed in a Dynamic View.
- Access and sort information in Active Tables.
- Expand only relevant areas of displays.
- Print a diagram.

Accessing Dynamic Views

Accessing Dynamic Views

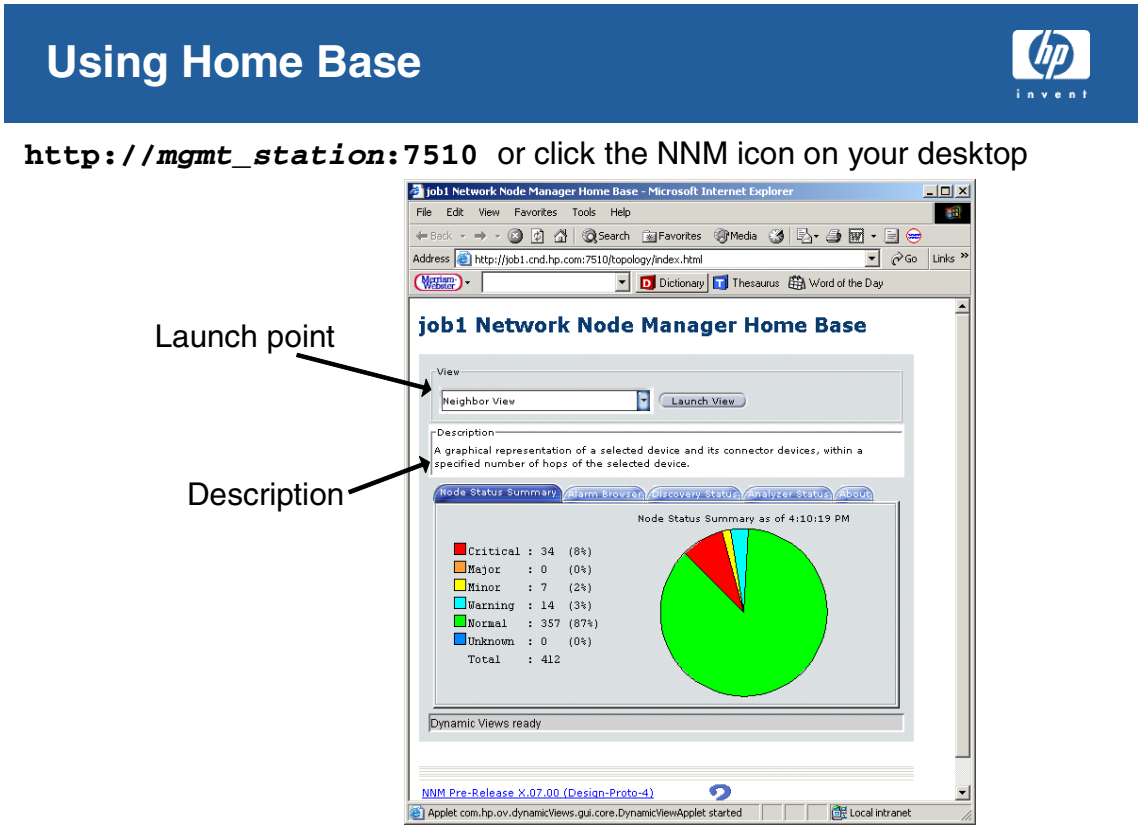
- From your browser window at Home Base
  `http://hostname:7510`
- From the Alarms Browser
- From the **`ovw`** menu: **`Tools:Views->`**
- From the Network Presenter: **`Tools:Views->`**
- From the Launcher Tools tab

U5089S ©2004 Hewlett-Packard Development Company, L.P. 2

There are numerous ways to access the views we just saw. First, you can launch them from Home Base. You can also use the NNM menus, the Network Presenter menus, or the Launcher.

One of the most useful access points is your Alarms Browser.

Using Home Base



Each NNM management station offers a Home Base which launches and describes all the Dynamic Views. This is a convenient page to bookmark for access to views on any NNM station.

If a view requires some context, like a node name or IP address, accessing the view from Home Base opens a blank view where you have a chance to enter the necessary information.

Using Alarms to Launch Dynamic Views

Using Alarms to Launch Dynamic Views

- Select an alarm, then select Actions:Views and the view you want
- Two views available:
  - Neighbor View
  - Path View
- Dynamic view launches with source node

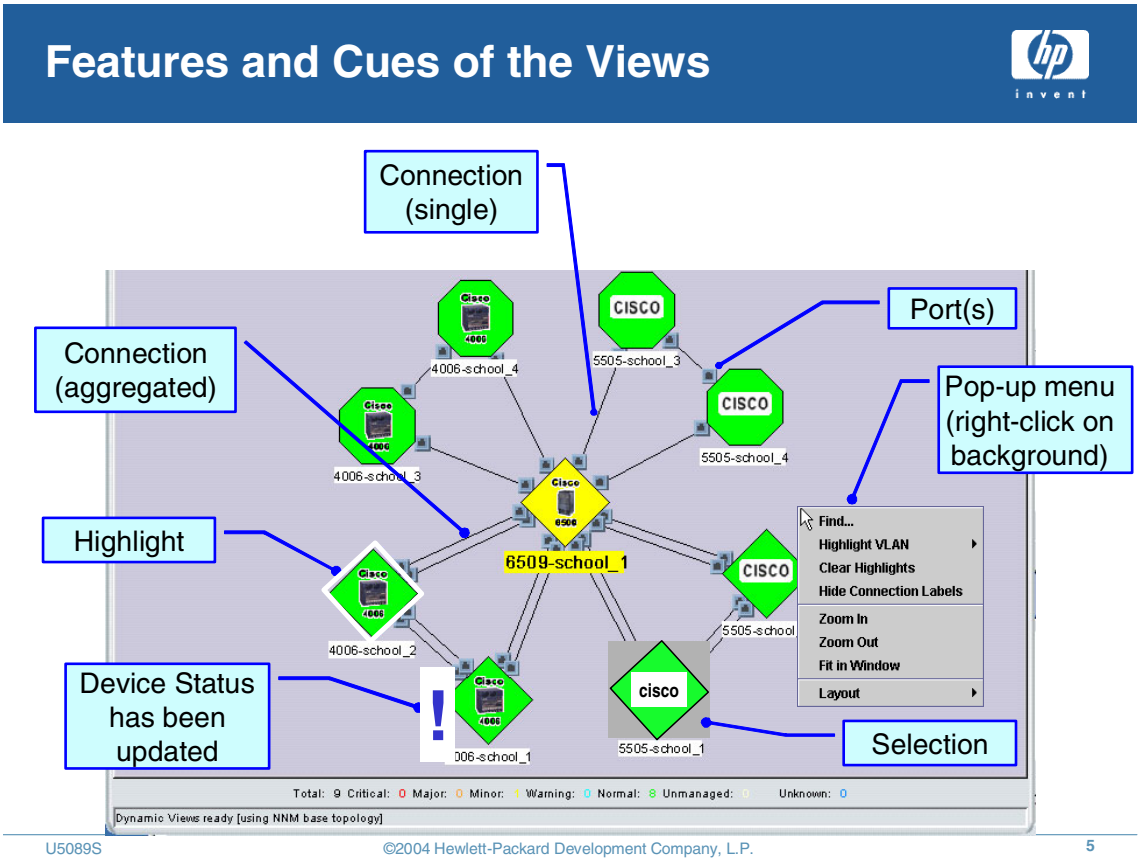
©2004 Hewlett-Packard Development Company, L.P.

You can select an alarm, and launch either the Neighbor View or Path View, with the alarm's source as the context. (Additional views may be available for some alarm types.) If the source of the alarm is "`Switch17.corp.com`", the Neighbor View launches with `Switch17` as the starting node. If you launch the Path View, you get the path from the NNM management station to `Switch17`.

Note that some alarms are not valid launch points for some views. For example, for a "Network Critical" alarm, neither the Neighbor View nor Path View make any sense, and View requests will result in an error.

Other views may be available for some alarms. Administrators can configure this in the file `xnmeventsExt.conf`.

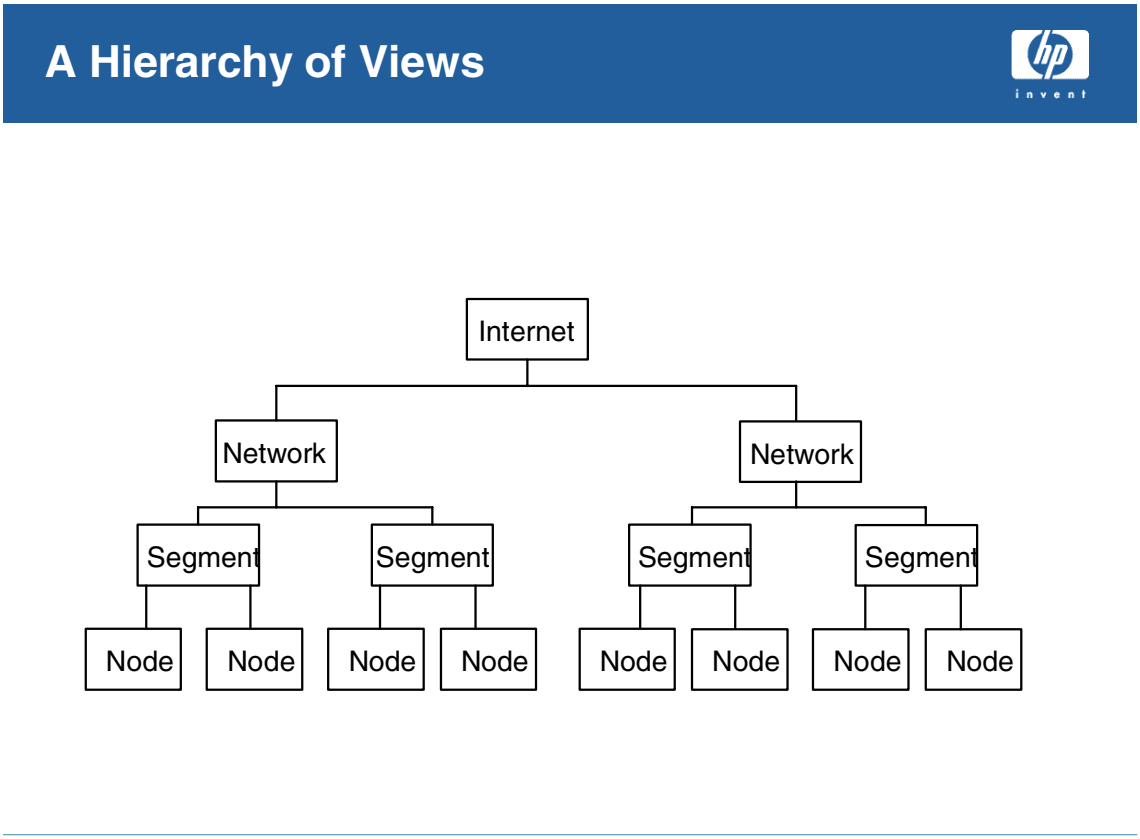
Features and Cues of the Views



First let's quickly review some of the behaviors and visual cues of NNM's dynamic views:

- Thick lines indicate port aggregation on a connection; thin lines indicate a single connection.
- Right-clicking on the background pops up a menu with several tools. Most of them are self-explanatory, but others deserve a few words:
  - The `Find` menu item makes it easy to locate specific objects within the view by name
  - The `Highlight VLAN` menu item puts a white highlighting border, as shown, around nodes that participate in the VLAN you select.
  - The `Layout` menu item lets you choose any of several layouts for the view. Experimentation is your best guide to the layouts.
- You can pause the mouse pointer over a node, connection line, or port icon (the small boxes at connection end-points) for additional information about the object.
- You can double-click on a device to get a highly detailed page of device-specific information.
- The status colors of symbols are dynamically updated as long as the view is open.
- When a node's status changes, it gets tagged with an exclamation mark. The mouseover information includes a status change history.
- A router is represented with a diamond. An octagon indicates a switch.

A Hierarchy of Views



You can view the areas of your network using the hierarchy shown above.

Internet

This view shows your IP and IPX networks at the highest level, showing only network and gateway (e.g. router) symbols. General-purpose computers may be shown in this view only if they are functioning as gateways (i.e. configured with multiple LAN cards).

Network

A network view lists all IP network segments and IP connector equipment within a network address space. Connector equipment displayed at this level includes gateways (including multi-homed hosts), bridges, hubs (multi-port repeaters), and repeaters.

Each segment group represents an actual network topology.

Segment

A segment view shows all IP nodes connected to the network segment. The topology shown corresponds to the segment's actual network topology, if known.

Node

A node details listing displays node components, such as network interface cards.

IPX View Hierarchy

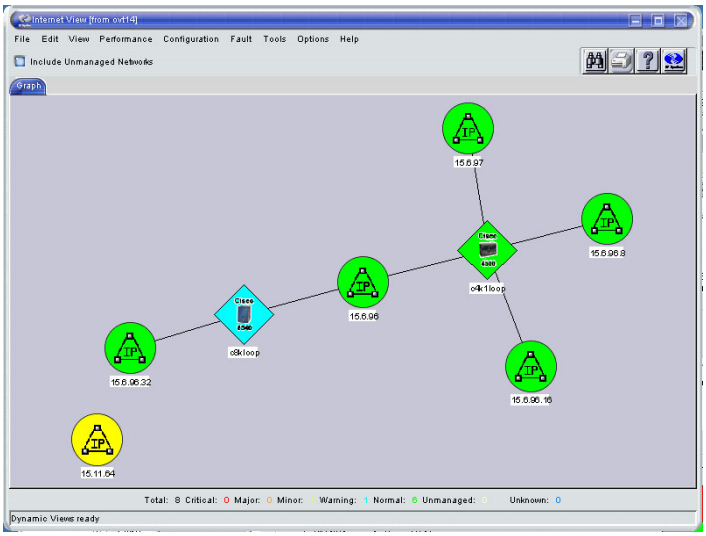
IPX views follow the same hierarchical layout as IP views. The Internet view contains network symbols representing NetWare networks. Opening a NetWare network submap results in the display of IPX routers and one segment. The segment can be opened to display all discovered NetWare nodes, clients, and servers.

All IPX nodes on a given IPX network are shown in a single segment.

Internet View

Internet View

- Shows networks in the environment
- Pop-up menu shows layout options available
- Similar display available in ovw



©2004 Hewlett-Packard Development Company, L.P.

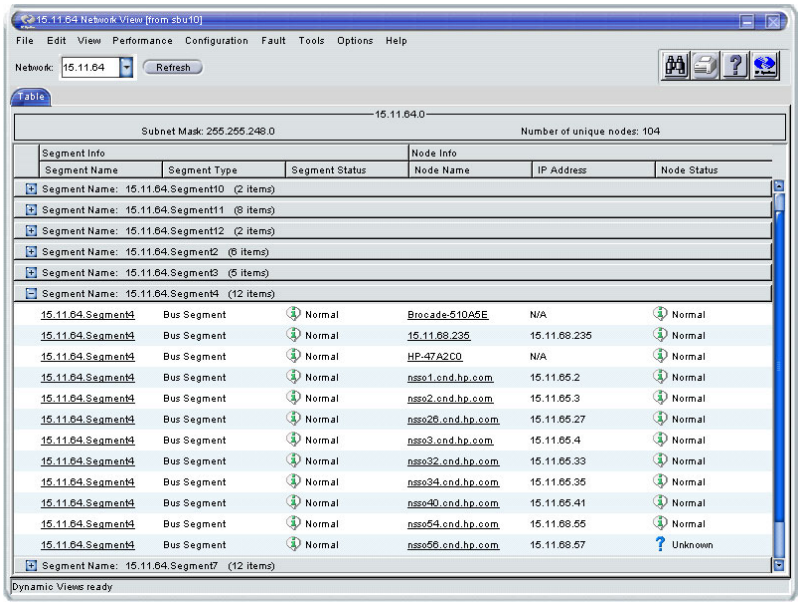
The Internet View is a graphical representation of the networks in your topology.

With the Internet view, you can see the general status of your network and locate problems in your network. Use this view for proactive monitoring of all of your networks.

Network View

Network View

- Unlike ovw, Network View is a table of segments in the selected network
- Shows segments, status, nodes, addresses, and links
- Access by double-clicking a network in the Internet View, or select from drop-down list



 ©2004 Hewlett-Packard Development Company, L.P.

The Network View is a table of the segments in a specific network. The table shows the status of the segment, and lists nodes that are on that segment, with links to detailed information about the nodes. (A segment refers to a collision domain or shared media.)

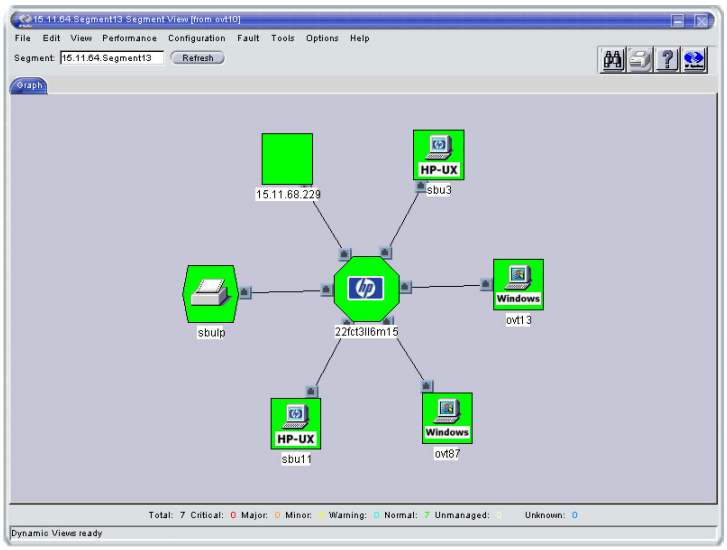
With the Network View, you can see the general status of a part of your network and locate problems there. Use this view for proactive monitoring of all of your networks.

You can access the Network View by double-clicking on a network symbol in the Internet view.

Segment View

Segment View

- Similar to the Segment submap in ovw
- Shows general status of segment and the nodes on it
- Access by double-clicking a segment hyperlink in a Network View.



 ©2004 Hewlett-Packard Development Company, L.P.

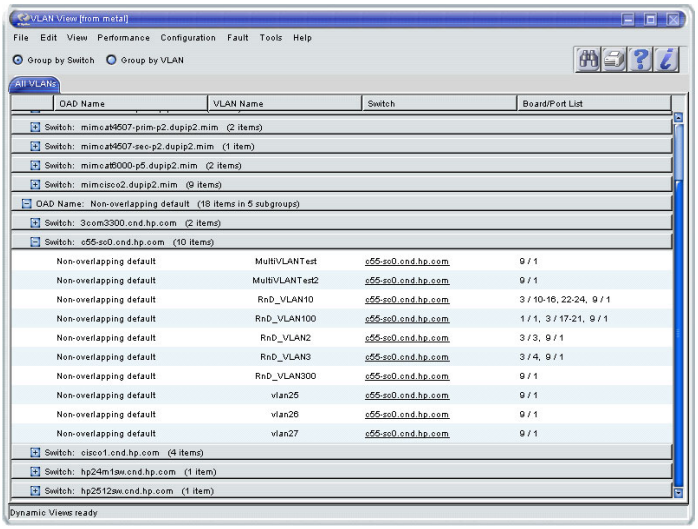
The Segment View is a graphical representation of the nodes on a specific segment in the network. (A segment is a node's collision domain or shared media.)

With the Segment View, you can see the general status of the segment and locate problems there. You can access the Segment View by double-clicking on a segment name in the Network View.

Extended Topology VLANs View

Extended Topology VLANs View

- VLANs View: a table of all your VLANs
  - Lists switches (with board and port) that participate in each VLAN
- Group by VLAN to see summary of all VLANs and which switches participate in each.
- Group by Switch to see all VLANS a switch participates in and which ports are in each one.
- Extended Topology must be enabled.



 ©2004 Hewlett-Packard Development Company, L.P.

A network administrator can utilize bridges to segment a LAN and solve bandwidth problems but this doesn't address broadcast problems. Any broadcast request is sent to all ports within the bridged network. A switch, although acting like a bridge, can be logically partitioned into separate broadcast domains called VLANs. This limits the scope of the broadcast to the individual VLANs. Network administrators find the capability to break up a single broadcast domain into multiple broadcast domains advantageous. Using inexpensive switches over the more expensive traditional routers to break up broadcast domains is also advantageous. It provides significant performance advantages. However, each broadcast domain is considered a separate subnet. In order to go between subnets, a Layer 3 component (such as a router) is still required.

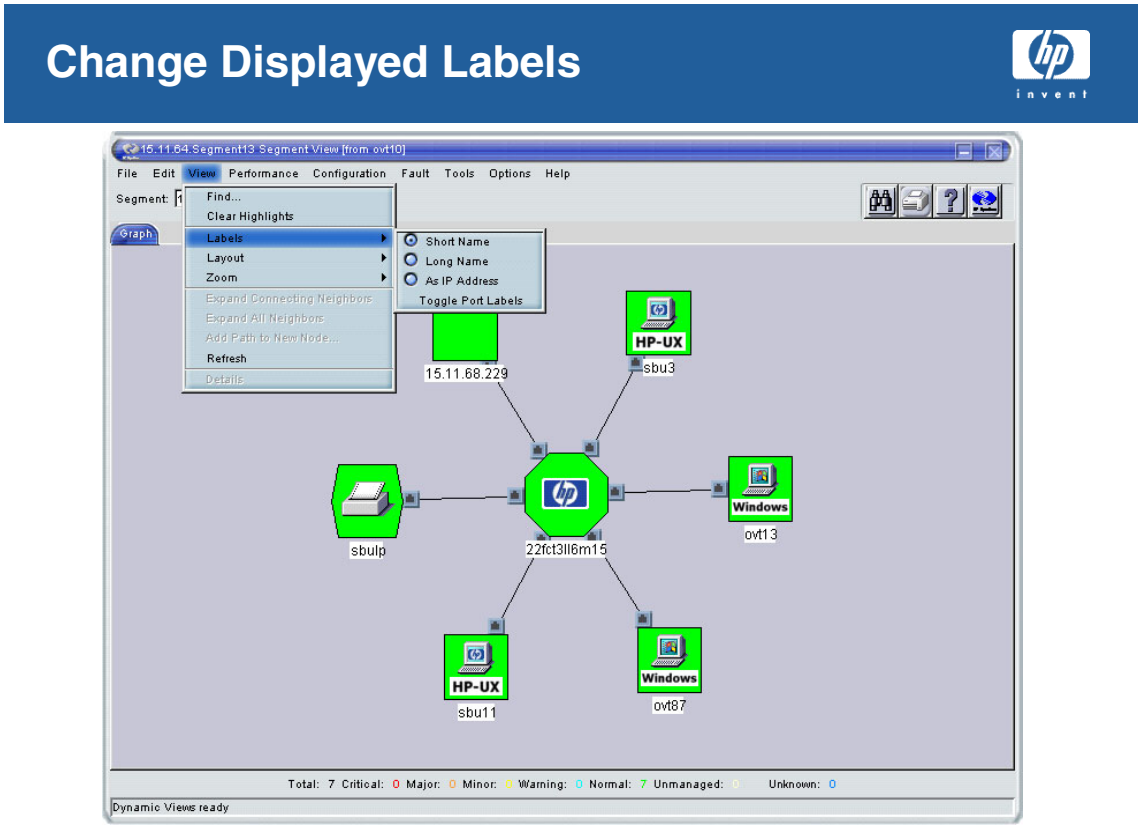
When you enable Extended Topology, the VLANs view presents a table listing the VLANs in your network. Use this table to get a general inventory of the VLANs and to navigate to a specific VLAN or switch. You can access this view from Home Base or by selecting a switch in another view and selecting VLAN View from the pop-up menu.

The table also lists the switches (with their boards or ports) that participate in a specific VLAN. Although the switch has the VLAN ID configured on it somewhere, it may or may not be actively participating in the VLAN.

Click on `()Group by VLAN` above the table to see a VLAN-centric view. The VLAN-centric view shows each VLAN and the switches whose ports participate in that VLAN. Click on `()Group by Switch` above table to see a switch-centric view. The switch-centric view shows the selected switch and the VLANs that it participates in.

You can configure data collection for a VLAN. For example, you can monitor the number of broadcast packets statistically.

Change Displayed Labels



You can view labels in Dynamic Views as short names (default), long names, or IP addresses by selecting `View:Labels`. You can find by short name, long name, or IP address based on the label scheme you are currently using.

You can copy the names to the clipboard for use in other applications.

Toggling Port Labels

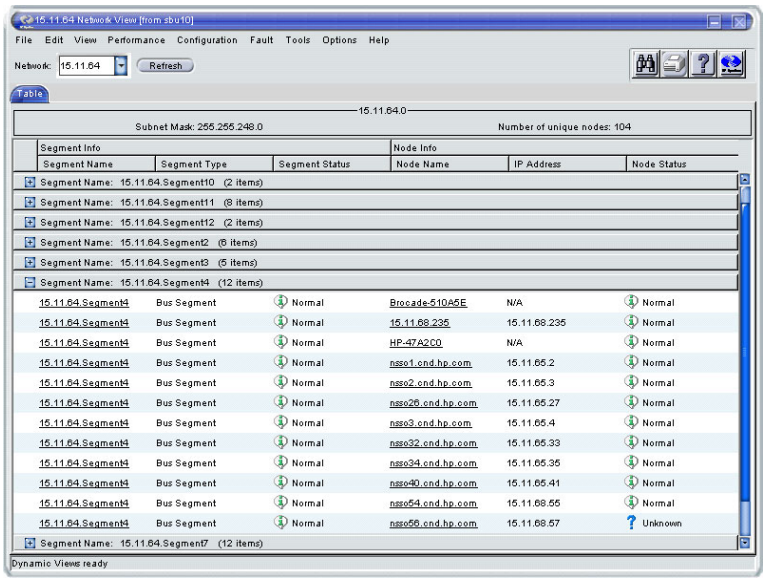
You can turn port labels on and off using `File:Labels->Toggle Port Labels`. This allows you to print out port labels on posters. Although it makes the map busy, the data is sometimes valuable enough to overcome this. Sometimes the nodes obscure the labels, and you need to zoom in more to see them.

This feature only makes sense if you have port icons displayed, which is mostly in Neighbor View.

Active Tables

Active Tables

- Dynamic status
- Sort, group, and filter presentation
- Control fonts and color
- Configuration changes not persistent



 ©2004 Hewlett-Packard Development Company, L.P.

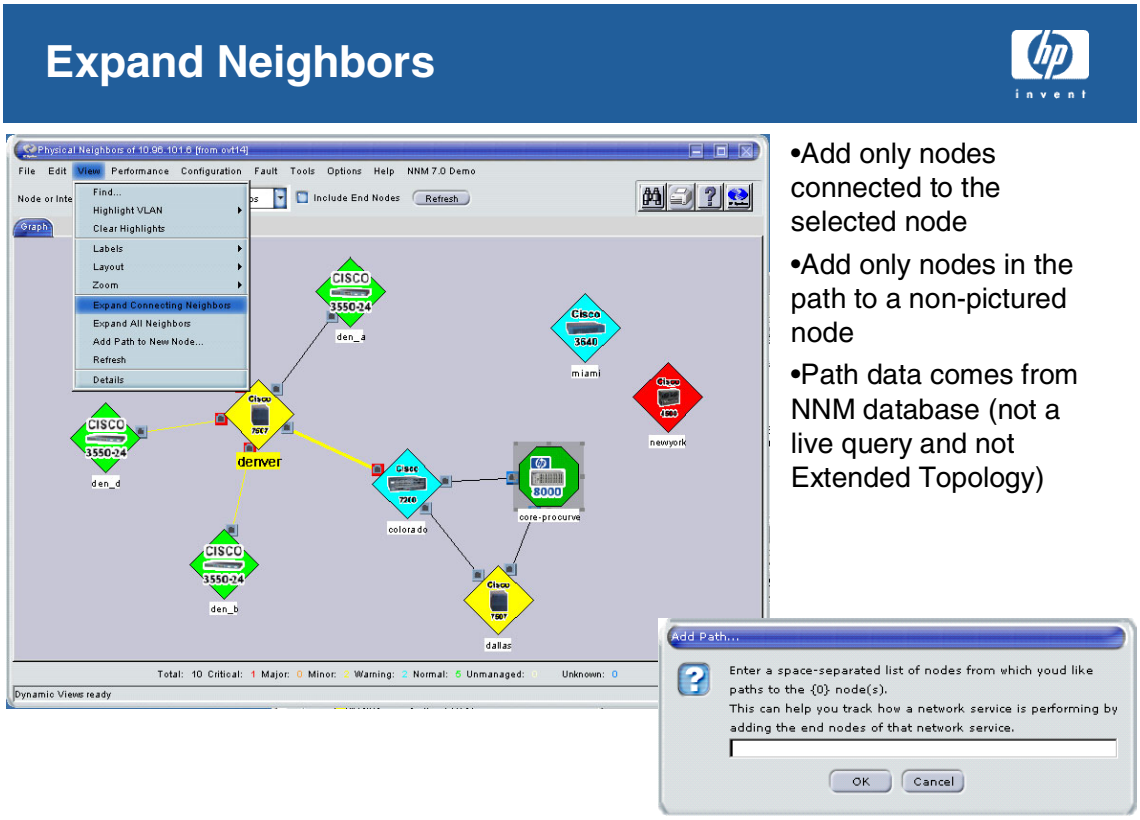
Graphs alone limit scalability of the NNM display. For example, looking at 3500 HSRP groups would be prohibitive. Therefore several views present tabular data exclusively or in addition to the graphical display.

Tables are dynamic. You can customize the sorting order by clicking on the column headers, and customize several other display attributes, but you cannot save your changes.

Where applicable, each dynamic view has an associated table on a Table tab providing you a mechanism to obtain additional information about the objects in the view.

Node, segment or other names provide hyperlinks to other views or details.

Expand Neighbors



 ©2004 Hewlett-Packard Development Company, L.P.

You can select a node and either right click or select `View:Expand Connecting Neighbors`, `Expand All Neighbors`, or `Add Path to New Node`. This allows you to control how to enlarge the map without needing to increase the number of hops.

Expanding neighbors adds all nodes connected to the selected node into your view. This can be much more selective than adding more hops of connectivity from every node in the view.

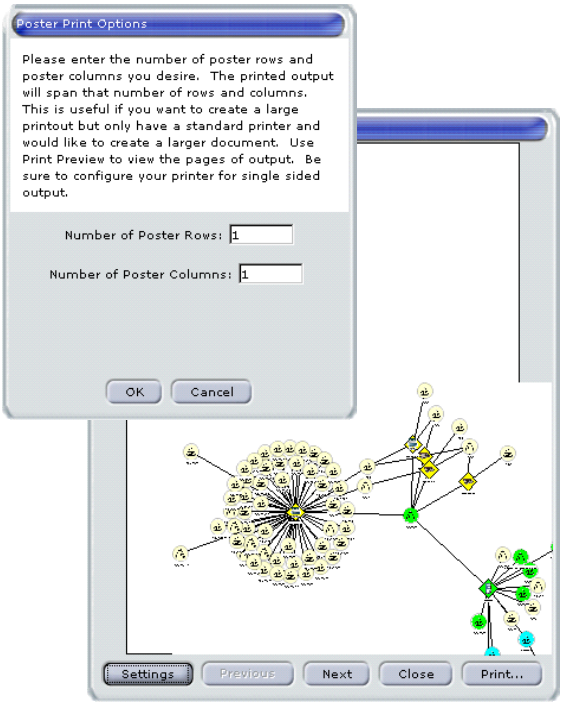
Adding the path to a specific node allows you to extend your diagram in a certain direction. The path information and devices along the way come from the NNM database, not a live query. Extended Topology connectivity information for the nodes added is not available.

The source must be a node or the menu item is greyed out.

Poster Printing

Poster Printing

- What is Poster Printing?
  - Useful if you don't have a big plotter
  - Requires some scissors and tape
  - Makes useful wallpaper, with readable icons for large maps
- Enable through **File:Poster Print Options**.
- **File:Print Preview** adds Next and Previous buttons



U5089S ©2004 Hewlett-Packard Development Company, L.P. 14

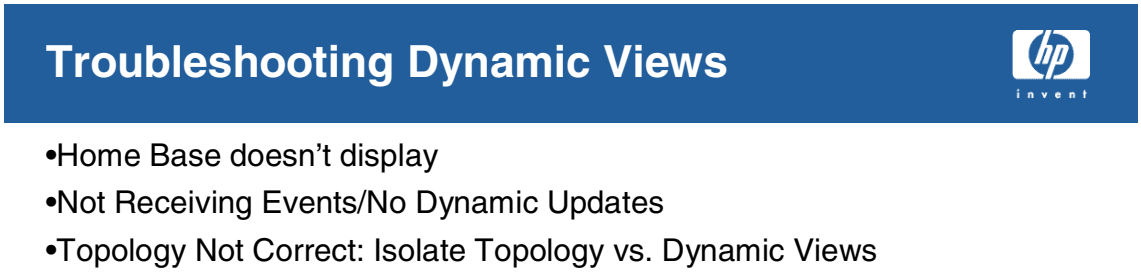
You can print Dynamic Views by selecting `File:Page Setup` or `File:Print Preview`, `File:Print`. This printing is superior to browser printing because the graph is scaled to the paper. With browser printing, the picture from the graph is scaled by the browser, causing the aspect ratio of icons to be incorrect.

You can create a large diagram of your network using Dynamic Views' Poster Printing feature. NNM utilizes as many sheets of paper across and down as you specify in the `File:Poster Print Options` menu.

When poster printing is enabled (something > 1 in either field), `File:Print Preview` provides `Previous` and `Next` buttons to let you page through your printout to see how well it fits.

Unfortunately, poster printing must redraw the view to scale it to fit the size of the output page, so any changes you made are not shown.

Troubleshooting Dynamic Views



- Home Base doesn't display

  Look in the Java Console. You may see stack traces that say why you can't load the .jar files.

  On Unix if port 3443 is unavailable, `ovhttpd` is not running. On Windows, it may be that IIS is not running. Start it via Control Panel->Admin Tools->Internet Information Manager. Right click on web server and Start.

  A great hint is that if the Help Icon on Home Base doesn't display, this indicates that it is the http portion of the interface that is having problems rather than Java.

- Not Receiving Events/No Dynamic Updates

  To see the formatted output from the Binary Event Store (BES) run `ovdumpevents -t -l 1`.

  To test operation in a non-production environment, `ovtopofix -S` is handy for generating NNM base topology events.

- Topology Not Correct: Isolate Topology vs. Dynamic Views

  Use `$OV_MAIN_PATH/support/NM/ovet_ovtopodump.ovpl` to query Extended Topology data.

  Run `ovtopodump -l <node>` to query NNM base topology.

Lab Exercises

Lab Exercises

•In this lab, you will:

- Access Dynamic Views from Home Base and from the Alarm Browser
- Navigate through the views
- Review view menus

©2004 Hewlett-Packard Development Company, L.P.

1. Start Home Base.

   a. What other views can be launched from here?

   b. Launch the “Internet” View. What menus are available from this View?

c. From the Tools menu, select the Views submenu. What Views are available from this menu?

d. From the Internet View, double click the symbol for your network. What is the result?

e. Click the + to open a segment. Double click one of the segment names listed. What are the results of this selection?

f. Take a few minutes to explore this navigation of elements.

2. Visit a view that uses the Table Presenter (e.g. Network View) and do sorting, etc. Note that when you restart the view all customizations are gone.

3. Go to a Path View and use Expand Neighbors.

4. Configure poster printing.

5. Turn port labels on and off.

6. Close all of the Dynamic Views windows except Home Base. Then open the All Alarms browser window from the Home Base Alarms tab.

   a. Select an alarm from your system. Then using `Actions:Views`, open up a Neighbor View.

   b. Experiment with the number of hops and showing end nodes. How does this influence the display?

B Securing Dynamic Views

Module Objectives

Securing Dynamic Views

Version C.00
U5089S Appendix B Slides

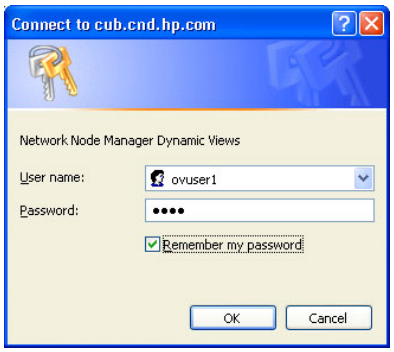
At the completion of this module the student will be able to:

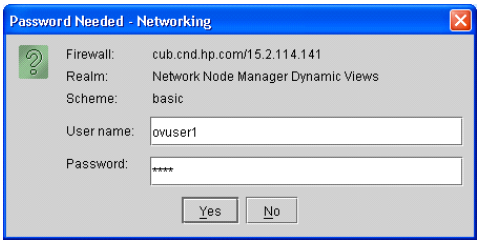
- Describe how Dynamic Views security uses tomcat realms.
- Configure users and passwords for Dynamic Views.
- Configure roles for Dynamic Views.
- Configure password encryption for storage in `dynamicViewsUsers.xml`.

User View of Dynamic View Security

Login Prompt Presented

- Browse to a URL such as:
  `http://host:7510/topology/summary`
- Need to login each time you exit all browsers or restart **`ovas`**, but browser remembers your password for you.
  - Browser caches credentials, so browsing to another page and returning doesn't require a new login





 ©2004 Hewlett-Packard Development Company, L.P.

This module covers how to enable Authorization with Tomcat. It allows Administrators a way to limit access to Dynamic Views from the web to prevent viewing of some servlets, configure logins and control who has access to remote configuration.

The end user is prompted with a login.

During Extended Topology setup (`setupExtTopo.ovpl`), you are prompted to enter an NNM Administrator password. This is used to restrict access to Extended Topology configuration.

Dynamic Views is a mixture of web-based application and pure Java application. Because of this, there exist a number of views that can be accessed from a Dynamic View window even if the web.xml file is configured with a security constraint requiring a user to login. For example, if you require a user login to access the Neighbor View, it will be possible to display the Neighbor View from other Dynamic Views windows without having to log in. In general, any view that can be accessed from the `Tools:Views` menu will be accessible regardless of the security constraints defined in the web.xml file. If a user login is desired for any specific Dynamic View other than the administrator views, require all users to login when accessing Home Base.

What is a Tomcat Realm?

What is a Tomcat Realm?

- Realm – a “database” of usernames, passwords, and roles assigned to those users, similar to groups in UNIX

- You can also use
  - Java Naming and Directory Interface (JNDI)
    - http://www.javaworld.com/javaworld/jw-01-2000/jw-01-howto.html
  - ODBC realms
    - use ODBC to communicate with a relational database which has realm information

- For more information, see
  - http://jakarta.apache.org/tomcat/index.html
  - http://jakarta.apache.org/tomcat/tomcat-4.0-doc/index.html
  - http://jakarta.apache.org/tomcat/tomcat-4.0-doc/realm-howto.html

©2004 Hewlett-Packard Development Company, L.P.

Tomcat is part of the Jakarta Project, part of the Apache Software Foundation. In NNM, `$OV_AS` references Tomcat 4.0.4 (implements J2EE Servlet 2.3 specification).

Tomcat supports declarative (and programmatic) access security. This is done with user to roles mappings called realms. These realms can be used with Tomcat's access security using <security-constraint> and <login-config>. <security-constraint> specifies which resources need to be protected, and which roles have access to the resource. The <login-config> element specifies the style of authentication required by the application.

Using Tomcat Realms for Security

Using Tomcat Realms for Security

- Configuring Tomcat to use Secure Socket Layer (SSL)
  - Possible, but not covered here – consult Tomcat documentation

- 3 files to modify
  - **`server.xml`** (enable Tomcat Memory Realm)
  - **`dynamicViewsUsers.xml`** (specify users/roles)
  - **`web.xml`** (define affected servlets)

- Support clear text and MD5 password encryption in the storage file
- Internet Explorer caches credentials
  - remember to stop and start the browser to activate

- To configure basics, either enable Extended Topology (**`setupExtTopo.ovpl`**) or run **`dvUsersManager.ovpl`**.

U5089S ©2004 Hewlett-Packard Development Company, L.P. 4

You can create an operator role that has access to all of the Dynamic Views, but cannot access any of the configuration tools.

WARNING **These files are overwritten with new installs and you will need to reconfigure your users and roles.**

Additional information on this procedure may be found in the *dynamicViewsUsers.xml* manpage (reference page on Windows).

Basic tomcat realms are configured by `setupExtTopo.ovpl`. If you have NNM Starter Edition or you are not enabling Extended Topology in your Advanced Edition, run the script `dvUsersManager.ovpl` to create the necessary structure. `dvUsersManager.ovpl` only works once (as in when you run `setupExtTopo.ovpl`). After that, edit the `web.xml` file manually if you want to change passwords.

Verify a Realm for Dynamic Views

Step 1: Verify server.xml

- Located in:
  - UNIX: **$OV_AS/conf**
  - Windows: **%OV_AS%\conf**
- Find near the bottom (above the </Host> tag):

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


©2004 Hewlett-Packard Development Company, L.P.

NNM places the line in `server.xml` during installation to point it to the dynamic views file.

Add Roles

Step 2: Add Roles

- Edit **web.xml** found in

  UNIX: **$OV_AS/webapps/topology/WEB-INF**

  Windows: **%OV_AS%\webapps\topology\WEB-INF**

- At the bottom of the file, before </web-app>
- Change from only requiring security for Extended Topology configuration to requiring it for all views.

```
<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>
```

U5089S ©2004 Hewlett-Packard Development Company, L.P. 6

You can modify the `security-constraint` element to enable web page logins. You can add different `auth-constraints` on different resources to provide for multiple different user roles. For more advanced uses, consult Tomcat's documentation at `http://jakarta.apache.org/tomcat/tomcat-4.0-doc/realm-howto.html`.

The `web.xml` file defines the role names available to users. It is located in directory:

UNIX: `$OV_AS/webapps/topology/WEB-INF`

Windows: `%OV_AS%\webapps\topology\WEB-INF`

Make a backup copy of `web.xml` before modifying it.

Security is already turned on for the Extended Topology configuration interface. This block extends that to all Dynamic View access. These lines are present but commented out in the `web.xml` file shipped with NNM. Remove the comment begin line immediately above and the comment end immediately below the web-resource-collection group to enable login protection for all Dynamic Views. (You cannot set security individually for views launched from Home Base. You may do so for other URLs.)

You can also add more roles if you need them in your environment. Once you have setup a role make sure you add a user to the `dynamicViewsUsers.xml` file that has that role.

WARNING **You must complete your configuration changes to both `web.xml` and**

`dynamicViewsUsers.xml` before attempting to have `ovas` read your changes.

You can make applications available to multiple users, as shown in the bottom of the following example:

```
    <security-constraint>
        <!-- Only allow administrators to perform configuration operations -->
        <web-resource-collection>
            <web-resource-name>Dynamic View Administrator Access</web-resource-name>
            <url-pattern>/etconfig/*</url-pattern>
            <url-pattern>/manage/*</url-pattern>
            <url-pattern>/unmanage/*</url-pattern>
            <url-pattern>/add/*</url-pattern>
            <url-pattern>/delete/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>administrator</role-name>
        </auth-constraint>
    </security-constraint>


    <!-- Restrict entire application to role operator (include administrator of course)  -->
    <security-constraint>
     <!-- Require operators and adminstrator roles for general DynamicViews access -->
     <!-- 1. administrator role must be define here to allow administrators to access all areas
-->
     <!-- 2. operators will not be able to access areas defined in previous security constraint
-->
        <web-resource-collection>
            <web-resource-name>Dynamic View Operator 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>
```

Add Users

Step 3: Add Users

- Edit **dynamicViewsUsers.xml**
- Name your own users
- Use role names from **web.xml**

```
<tomcat-users>
    <user name="ovuser1" password="ovpw"
      roles="operator" />
    <user name="admin1" password="ovpw"
      roles="administrator" />
</tomcat-users>
```


©2004 Hewlett-Packard Development Company, L.P.

The `dynamicViewsUsers.xml` file contains the configuration for user security for Dynamic Views and is located in:

UNIX: `$OV_AS/webapps/topology/WEB-INF`
Windows: `%OV_AS%\webapps\topology\WEB-INF`

Make a backup copy of `dynamicViewsUsers.xml` before modifying it. You may find additional information in the *dynamicViewsUsers.xml* manpage.

The `dynamicViewsUsers.xml` file is modified during the first Extended Topology discovery initiated by `$OV_BIN/setupExtTopo.ovpl`.

You can configure additional users and passwords by editing `dynamicViewsUsers.xml`. For each user to be added, insert a line with the syntax:

`<user name="`*user_name*`" password="`*pwd*`" roles="`*role_name*`" />`

user	keyword indicating definition of a user.
name	login name to be used by the user.
password	password to be used by the user.
roles	areas of user interface to which the user should have access. Role names are defined `web.xml`. To allow a user access to multiple roles, use a comma-delimited list of roles.

Restart ovas to Read Configuration

Restart ovas to Read Configuration

- Stop and start **ovas** to make changes take effect

  ovstop ovas

  ovstart ovas

- Check for any XML syntax errors
  - UNIX: **$OV_PRIV_LOG/ovas.log**
  - Windows: **%OV_PRIV_LOG%\ovas.log**

- Exit and restart web browser.

U5089S ©2004 Hewlett-Packard Development Company, L.P. 8

Because this uses Tomcat's in-memory realm, after changing passwords or modifying user roles you must stop and start the application server (ovas) to make this take effect.

Most browsers retain login information, so you need to stop and restart any browsers if you change passwords.

Using MD5 Password Encryption

Using MD5 Password Encryption

- By default, the login and password are stored in clear text.
- The login is sent across the network with passwords in clear text (BASIC).

- To use MD5 encryption for storing the passwords:
  1. Get an encrypted password.
  2. Enter the encrypted password into **`dynamicViewUsers.xml`**.
  3. Inform **`server.xml`** to use encryption.

©2004 Hewlett-Packard Development Company, L.P.

By default, the login is validated using the BASIC method of authentication, which sends passwords weakly encoded across the network.

You can store the passwords with more secure MD5 passwords.

To have your passwords encrypted in the `dynamicViewsUsers.xml` file:

1. Execute (all one line)

   UNIX: `"$OV_JRE"/bin/java -classpath "$OV_AS"/server/lib/catalina.jar org.apache.catalina.realm.RealmBase -a MD5` *`USERPASSWORD`*

   Windows: `"%OV_JRE%\bin\java" -classpath "%OV_AS%"/server/lib/catalina.jar org.apache.catalina.realm.RealmBase -a MD5` *`USERPASSWORD`*

   where *`USERPASSWORD`* is the password you want to enter in the password dialog.

   See the *dynamicViewsUsers.xml* reference page (manpage on UNIX) for information on Linux tools.

2. This returns a string like `:USERPASSWORD:e7dd56ec79c8584f3c25327b0b1c111c`. Use the resulting string as the password field in `dynamicViewUsers.xml`.

3. Edit the `server.xml` file to change the `digest` parameter in the `<Realm>` element of the

"/topology":

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

4. After adding all users, passwords, and roles close all NNM browser windows and restart ovas:

   ```
   ovstop ovas; ovstart ovas
   ```

WARNING **Use DIGEST authentication to encrypt the passwords going across the line at your own risk since Tomcat has known defects.**

Lab Exercises

- Enable existing roles
- Enable existing users
- Configure a role
- Configure a user
- Configure encryption

 ©2004 Hewlett-Packard Development Company, L.P.

Use the demo or classroom topology to complete the following lab exercises.

NOTE If you have not enabled Extended Topology, run `dvUsersManager.ovpl` before starting the lab.

1. Enable existing roles and users to require a password for access to all dynamic views.

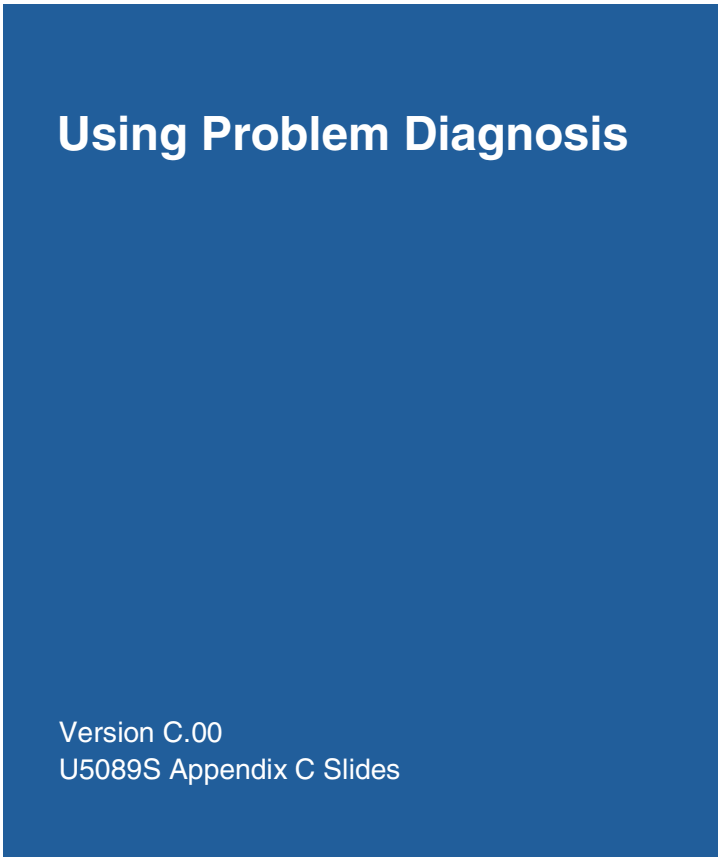
2. Create a role named specialist and add a user for it. You need to do these two steps together because `ovas` requires a user for each defined role. The specialist should be allowed to add and delete nodes in Dynamic Views.

3. Create a user, super, who can do everything an administrator can do and everything a specialist can do.

4. Configure encryption.

C Using Problem Diagnosis

Module Objectives



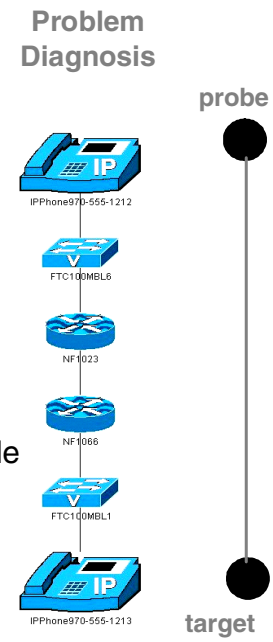
At the completion of this module the student will be able to:

- Track network path performance and detect brownout conditions using Probe-based technology.
- List the components of Problem Diagnosis.
- List the requirements for Problem Diagnosis.
- Start the Problem Diagnosis view.
- Configure endpoints.
- Describe the fields in a Path List.
- Read a Path Map, Path Detail and Trek Detail diagram.
- Describe the meaning of a partial path.
- Describe Problem Diagnosis' brownout alarms.

Overview of Problem Diagnosis

Overview of Problem Diagnosis

- Integrated Fault & Performance
- For your most critical paths, place a probe to monitor the path.
  - What is the current network path from A to B?
  - What are all the possible paths from A to B?
  - Is the path performing normally?
    - Blackout (destination cannot be reached)
    - Brownout (destination can be reached but the packet times are above the expected value)
    - Routing loops, host unreachable, network unreachable, protocol unreachable, port unreachable
  - Has the path changed?
    - Flapping
    - Switched from high speed/low cost to low speed/high cost



U5089S ©2004 Hewlett-Packard Development Company, L.P. 2

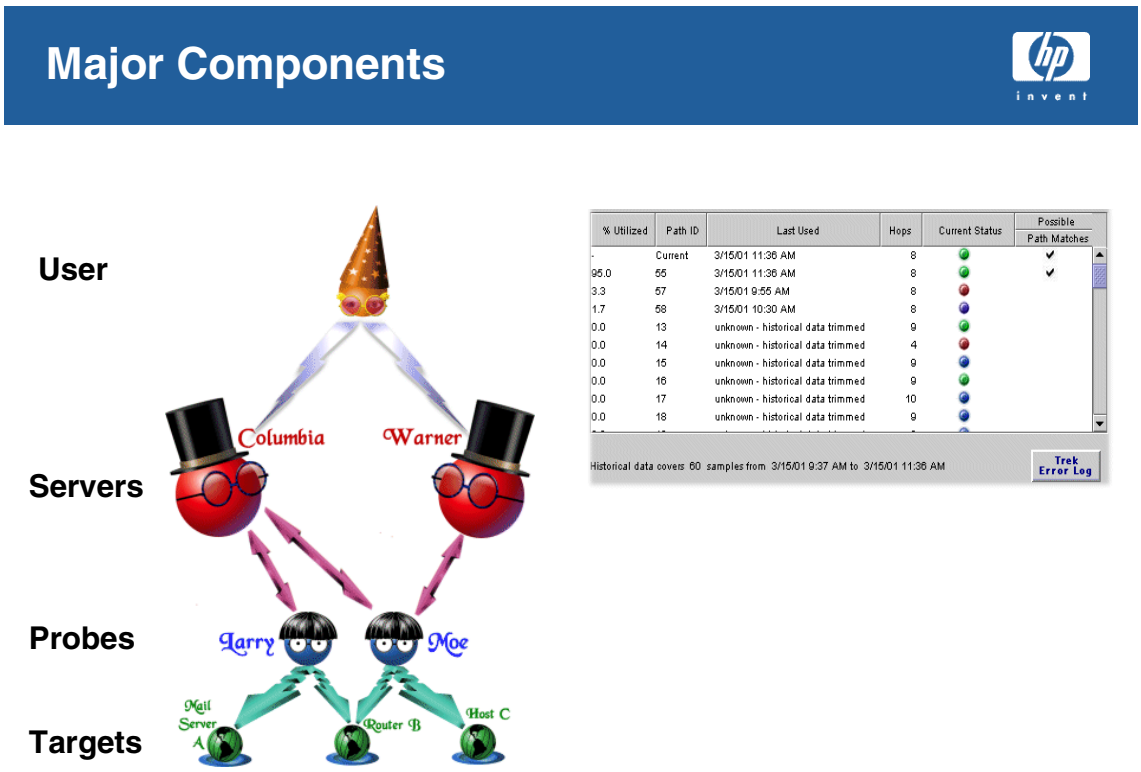
Problem Diagnosis provides powerful, automated IP network path analysis functionality that presents end-to-end path information clearly and concisely. Furthermore, Problem Diagnosis lets you see detailed information from nodes and devices in a particular path.

Problem Diagnosis offers a probe-based path tool that finds and monitors the paths between itself and any reachable node that it is configured to test. A Problem Diagnosis probe collects data over time, and generates statistical and usage data about the paths it monitors.

Brownout detection is unique in network management software. When a path performs statistically poorly, Problem Diagnosis issues a brownout event to your Alarm Browser telling you between which devices the brownout is occurring.

Problem Diagnosis serves a different purpose from Path View. When you select `Tools:Path View`, you can view the path between any two endpoints in the NNM topology database and only shows the expected path based on the contents of the NNM database. Problem Diagnosis shows paths only from a probe to a configured target. Problem Diagnosis is oriented toward those critical devices in the environment. Problem Diagnosis would monitor the top 20 crucial paths on an ongoing basis. When you select Problem Diagnosis view, it performs an immediate, live `traceroute` as well as showing you historical data for that path. Path View is for troubleshooting.

Major Components



Problem Diagnosis has three primary components:

- The Web-based Graphical User Interface
- The Problem Diagnosis Server
- The Problem Diagnosis Probe(s)

Here is a diagram of how paths are discovered. The user (the one in the wizard's hat) uses a web-browser to access one of the two servers (Warner or Columbia). By requesting a path query using a particular probe (say, Moe), it's possible to investigate the current path between Moe and any of its target nodes (Router B or Host C), and to get a detailed history of the paths that have been used between Moe and those target nodes.

Problem Diagnosis Server

Problem Diagnosis Server

- Resides on the management station
  - Process name is **pd** in **ovstatus** listing
- Collects data from one or more probes at strategic positions in the network
- Displays data in views for users
- Topology integrated with NNM Extended Topology

©2004 Hewlett-Packard Development Company, L.P.

The Problem Diagnosis server is the heart of the system, the intelligence behind Problem Diagnosis functionality. It assimilates information and responds to requests from a user running the Problem Diagnosis view.

The Problem Diagnosis server gets its topology data from NNM Extended Topology, from Problem Diagnosis probes, and other HP OpenView applications.

Problem Diagnosis information is based on `traceroute` and uses Extended Topology to fill in level two devices when it is available and L2 is selected. Otherwise it gathers level 3 information from `ovtopmd`.

Based on the topology data it mines from these sources, it can present several alternative ways to examine the path(s) between nodes.

Problem Diagnosis Probe

Problem Diagnosis Probe

- Can report to multiple servers
- Collects information about the path from itself to a target
  - Can monitor an unlimited number of paths
- Tests the route periodically to collect statistics
- Tests the route on user request for real-time information
- Looks for
  - Devices along the route
  - Lag time between devices
  - New routes to the target

U5089S ©2004 Hewlett-Packard Development Company, L.P. 5

The Problem Diagnosis probes are key suppliers of data to the Problem Diagnosis system. Probes are independent Java applications that can reside anywhere. There is no limit to the number of probes you can install or the number of paths a probe can monitor. Likewise, a probe can be used by more than one Problem Diagnosis server.

A probe collects information about paths between itself and any desired target. It uses a technique similar to the `traceroute` utility, and runs periodically to test the route to the target(s) for which it is configured. On each run, it collects data about the route:

- Devices along the route
- Lag time between devices
- New routes to the target

When you request probe data, the Problem Diagnosis server contacts the probe for current data, so that you see the freshest information.

Starting the User Interface

- From Home Base, select Problem Diagnosis View and click [Launch View].
- The server must be running.
- The probes must be running.

 ©2004 Hewlett-Packard Development Company, L.P.

You launch the Problem Diagnosis View from Home Base.

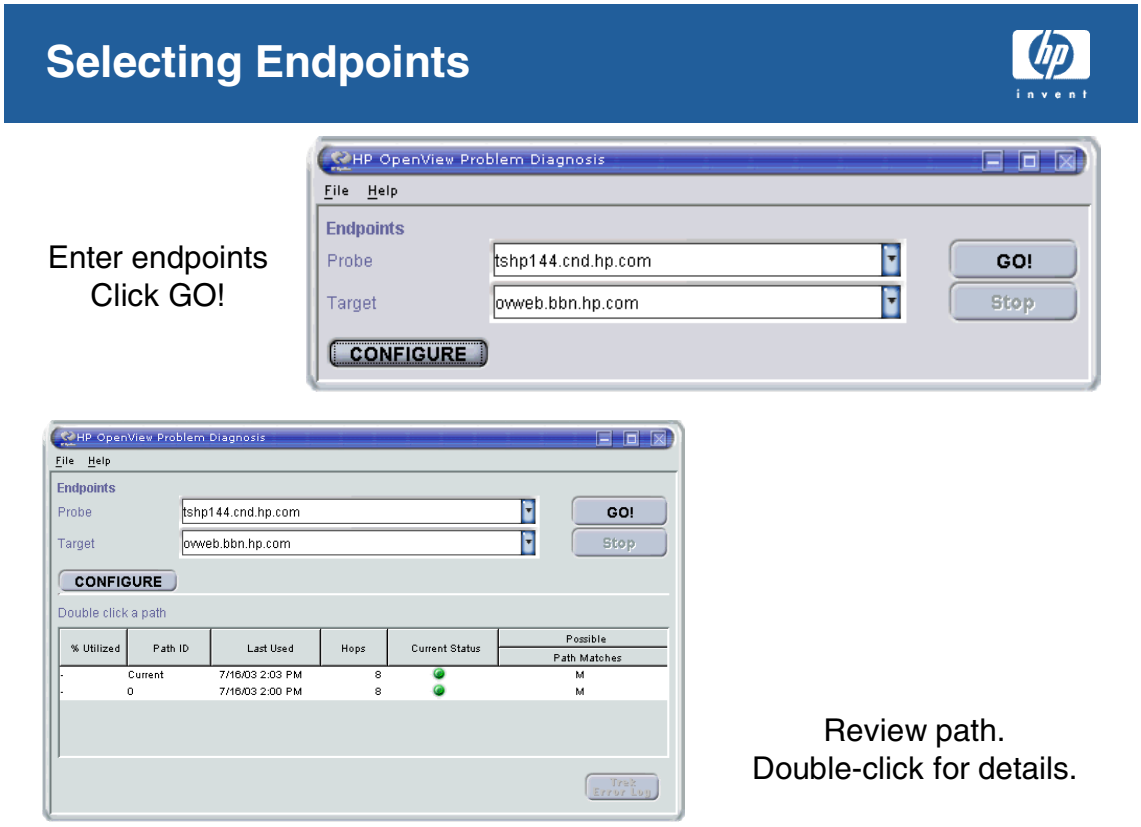
The Problem Diagnosis View is simple to use and presents data in easy-to-understand ways. From the Problem Diagnosis View, Problem Diagnosis opens its own windows for you to work in.

1. Start the Problem Diagnosis server that you will connect with, if it isn't already running.
2. Start all probes for that server, if they aren't already running. It may take a few minutes before Problem Diagnosis has access to all probes.
3. From Home Base, select Problem Diagnosis View from the View drop down list and click `[Launch View]`. The Problem Diagnosis main dialog opens.

Extended Topology must be enabled.

Certain terms differ slightly depending on the platform. For example, the facility for tracing IP routes is named `traceroute` on UNIX and `tracert` on Windows. Generally speaking, your familiarity with the world of computer networking will make these terms obvious.

Selecting Endpoints



When you start the Problem Diagnosis View from Home Base, the Problem Diagnosis main dialog opens. Here, you choose the endpoints you want.

The endpoints are configured into the probe by clicking [CONFIGURE]. Replace the text in the Target column with the node name you want monitored. You may enter any target device, whether it is in NNM's topology or not.

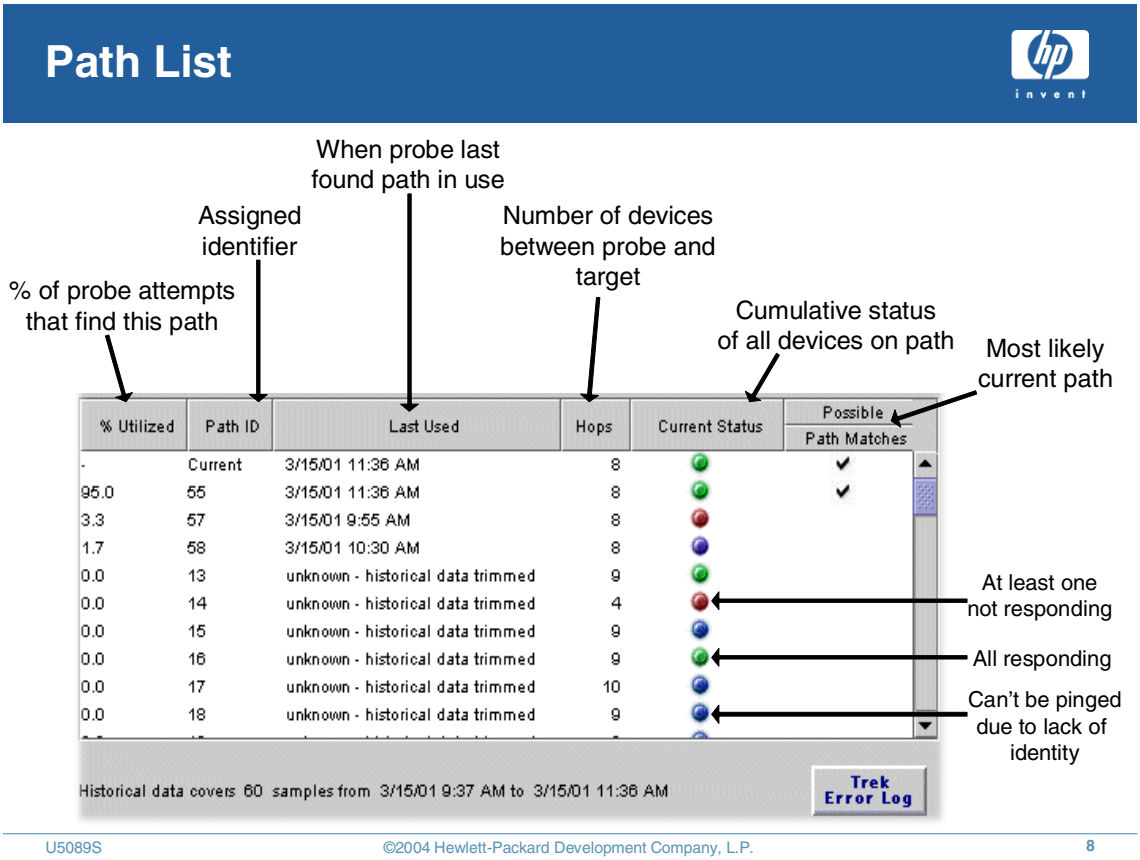
Endpoints are specified as "Probe" and "Target". Any path between them is directional in the sense that it is determined by the probe's activity, and from the probe's point-of-view.

After selecting the Probe and Target, click the [GO] button to proceed. At this point, the [GO] button is disabled while you wait--there is considerable background activity, so be patient for the results of the path query to come in. (You can [Stop] at any time if you change your mind.) The probe does a `traceroute` (`tracert` on Windows) to the target, noting the devices and lag times along the path.

When you request a path query, you get a table listing the known paths between the two endpoints. This table is appended to the bottom of the main window.

You can then double-click on one of the paths to open the Path Details window, with the Path Map of the selected path.

Path List



A path is defined as a series of connecting interfaces and devices between one endpoint device and another. The Current Path is the one returned when you clicked [GO].

The path list is appended to the Problem Diagnosis View window when you click [GO]. You can double-click on a path to launch a specific path window. You can get special information about the Current path through Path Detail (Current Path). The view is static (does not reflect status changes) until you re-launch it.

Entries in the table are as follows:

Item	Description
%Utilized	The percentage of probe attempts that find the given path.
Path ID	A numeric identifier for the path, assigned when the path is first found.
Last Used	Indicates when the probe last found the path in use.
Hops	Number of devices between the probe and target.

`Current Status`

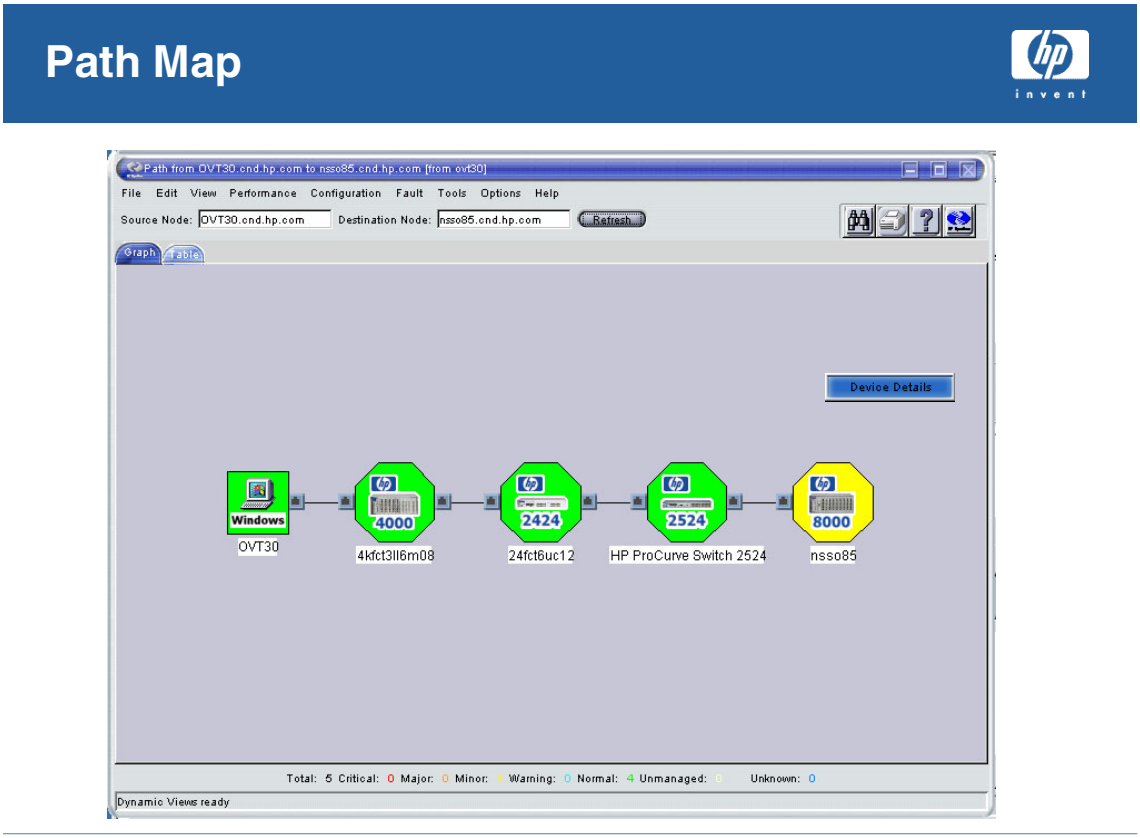
The status of all the elements on the path taken as a whole. Note that this is the present status, determined at the moment the `[GO]` button is pressed. Possible values are:

Normal:	All nodes in the path are known and responding to pings.
Critical/Down:	At least one known node in the path is not responding to ping.
Unknown:	At least one device in this path is "unknown." Unknown has special meaning in the path list. An "unknown" device is known to exist in the path, and is known to be functional. However, the device did not respond to the path query with its identity, and so it cannot be pinged to determine its status.

`Possible Path Matches`

Problem Diagnosis attempts to find, among all the known paths, those paths that are most like the Current path. If there is a perfect match, this column will have a check mark for that path. Sometimes there is more than one possible match (suppose the current path contains an unknown device which could be any of several known devices). In such cases, every path that could potentially match the current path is marked. If no pre-existing path resembles the `Current` path, no other paths are marked.

Path Map

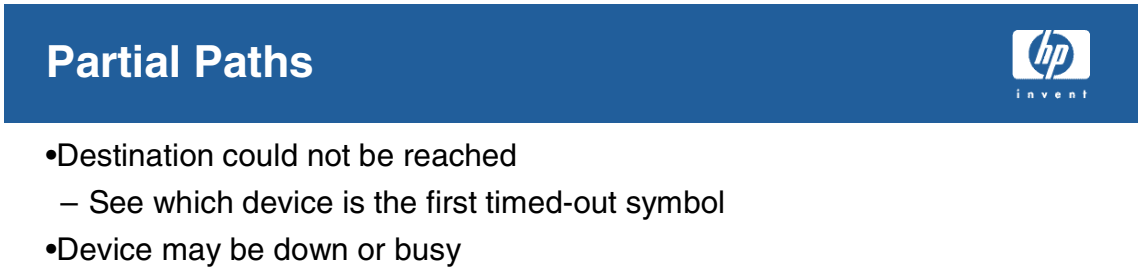


This map shows the path between two nodes as determined by Problem Diagnosis. Notice the following:

- The interfaces in the path are the “inbound” interfaces on each device, that is, the interfaces that the ping packets arrive at on each device. The sole exception is the inclusion of the “outbound” interface on the probe's host.
- Status is shown for each device on the path.
- You may see a `TIMED_OUT` label on one or more devices. A `TIMED_OUT` device is known to exist at this point in the path, and is known to be functional. However, the device did not respond to a path query when it was found, so neither its identity nor its status can be determined. (Seeing many `TIMED_OUT` symbols may indicate Partial Paths.)

The view is static (does not reflect status changes) until you re-launch it.

Partial Paths

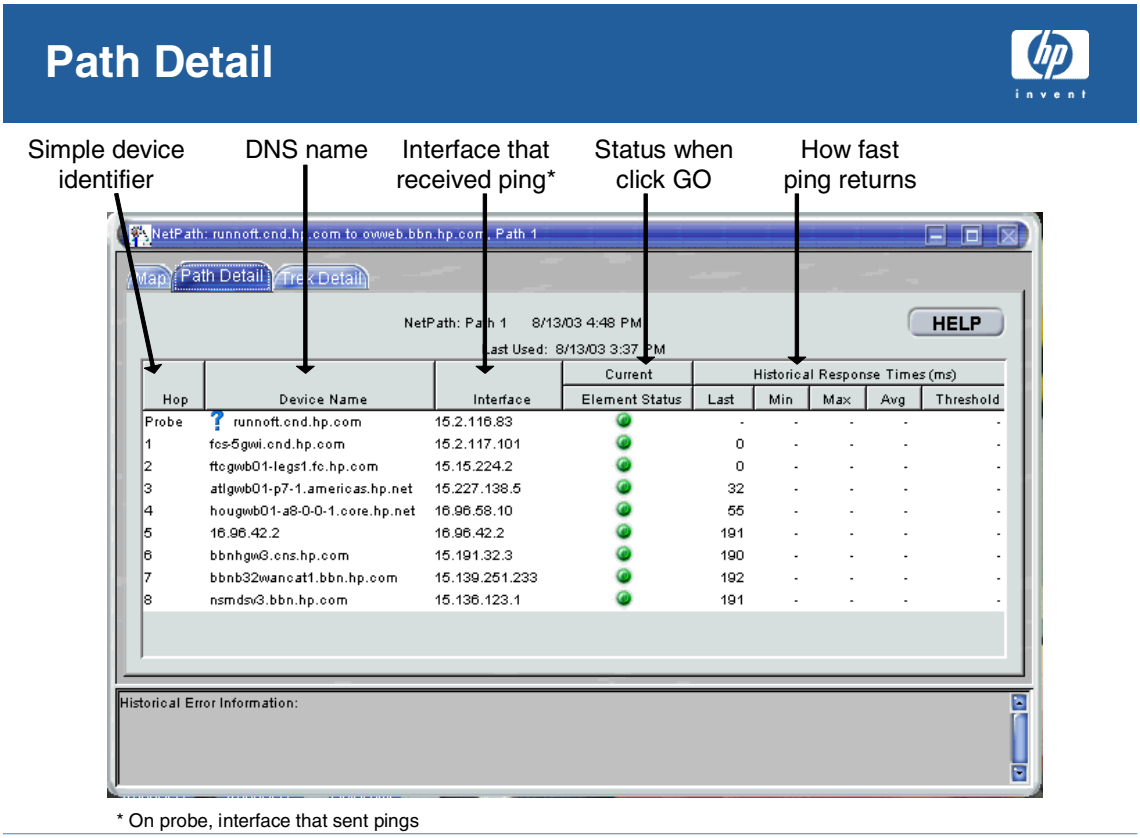


A “Partial Path” from Problem Diagnosis deserves special consideration. When Problem Diagnosis runs a trace against a configured target, several things can happen:

- If the destination is not reachable from the probe, a partial path is all that can be found. The trace inevitably reaches a point where time-outs recur, because a device that is expected to respond never does. Problem Diagnosis translates each time-out into a `TIMED_OUT` symbol on the path map. If the final symbol on a path map is a `TIMED_OUT` symbol (frequently preceded by other `TIMED_OUT` symbols), it means the destination could not be reached. There is a parallel series of `TIMED_OUT` entries in the Path Detail panel, for exactly the same reasons.
- One of the routers on the path may be down, too busy to respond, or configured to silently drop expired packets. In any of these cases, the probe waits for a response, and when one does not arrive in time, it marks that device as `TIMED_OUT`. It is entirely possible that the device is working properly, and that the next hop in the trace succeeds. This is not a partial path.
- If the maximum time-to-live is set too low in the Probe Configuration, the trace may time out before reaching the destination. This is a partial path, in that it does not include the destination. No errors are necessary for this to occur, however, and the destination may still be reachable if the time-to-live is increased.

If you have routing loops in your network, traceroute will notice a routing loop. An event is sent to the Alarm Browser and shown in the Trek Detail error log in Problem Diagnosis. The path detail will stop after first repeat without reaching the destination.

Path Detail



This table lists details about each hop along the path between two nodes as determined by Problem Diagnosis. Note that for all nodes except the first (the probe host) the interface listed is the one on that device that received the ping from Problem Diagnosis. On the first node, the interface listed is the one which sent the pings.

The Current Path is treated specially compared to all other paths.

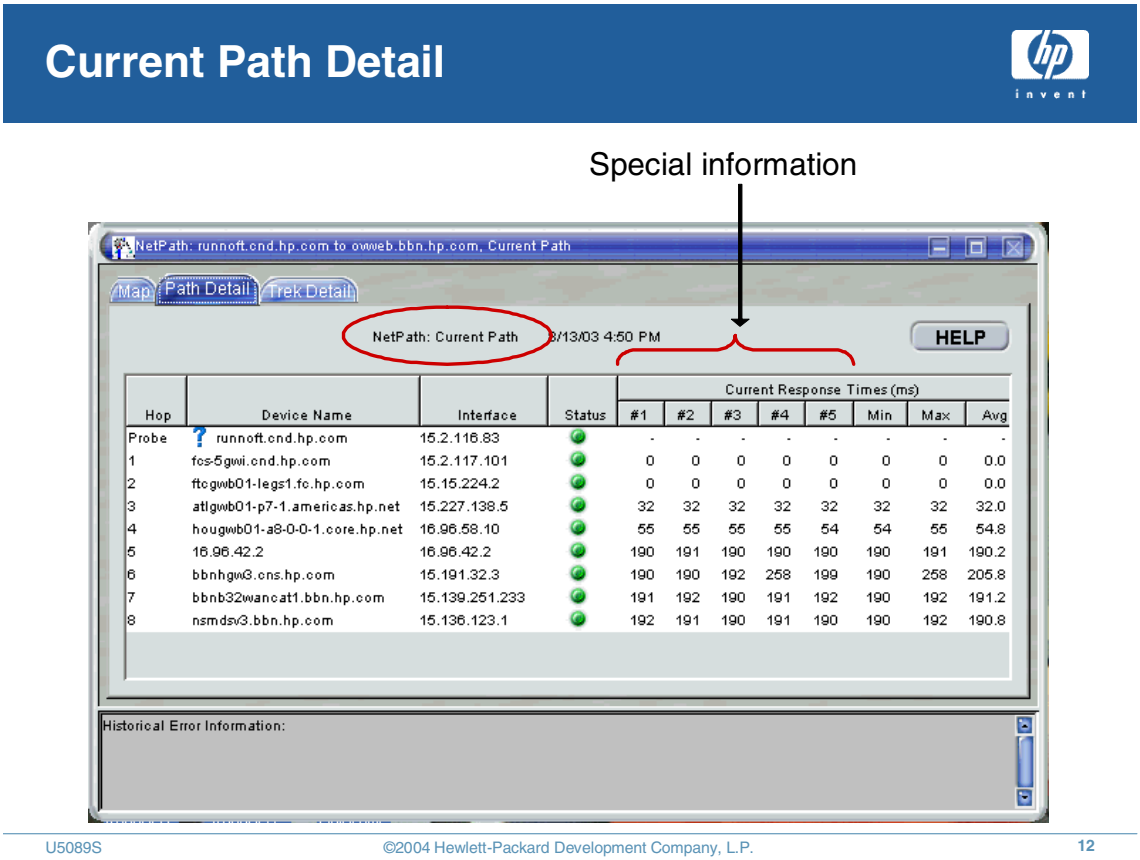
If you click on an entry to select it, you can then right-click on a device or interface to get a pop-up menu with various troubleshooting tools on it.

The table entries are as follows:

Item	Description
`Hop`	A simple identifier for a device on the path.
`Device Name`	The DNS name (or IP address, if a name is not available) of the device. If the type of device can be determined, it is denoted by the symbol. Devices that timed out bear the name `TIMED_OUT`.
`Interface`	The interface on the device which received the ping from the probe. For the first node only (the probe host), it is the interface that sent the pings.

Item	Description
`Status`	The current status of the device. This is the present status, evaluated by Problem Diagnosis when `[GO]` is pressed.
`Historical Response Times (ms)`	Response time measures how fast the ping response returns when this particular device is pinged. Note the following: • Response time includes the cumulative response times of intermediate devices. • The data set includes only the latest "n" samples, where "n" is the maximum number of samples configured for this target. • Devices with Unknown status (which also bear the name `TIMED_OUT`) have null (`*`) response times. Maximum and average response times may tend to drift upward on devices that are near capacity, or suffering other problems. • *Last*: The most recent response time recorded. • *Min*:The fastest response time in the data set. • *Max*: The slowest response time in the data set. • *Avg*: The average response time across all records in the data set. • *Threshold*: The threshold response time is calculated to be three standard deviations (Poisson distribution) above the median response time. If current performance exceeds the Threshold value, there is a high probability that a problem exists on this device.

Current Path Detail



 ©2004 Hewlett-Packard Development Company, L.P.

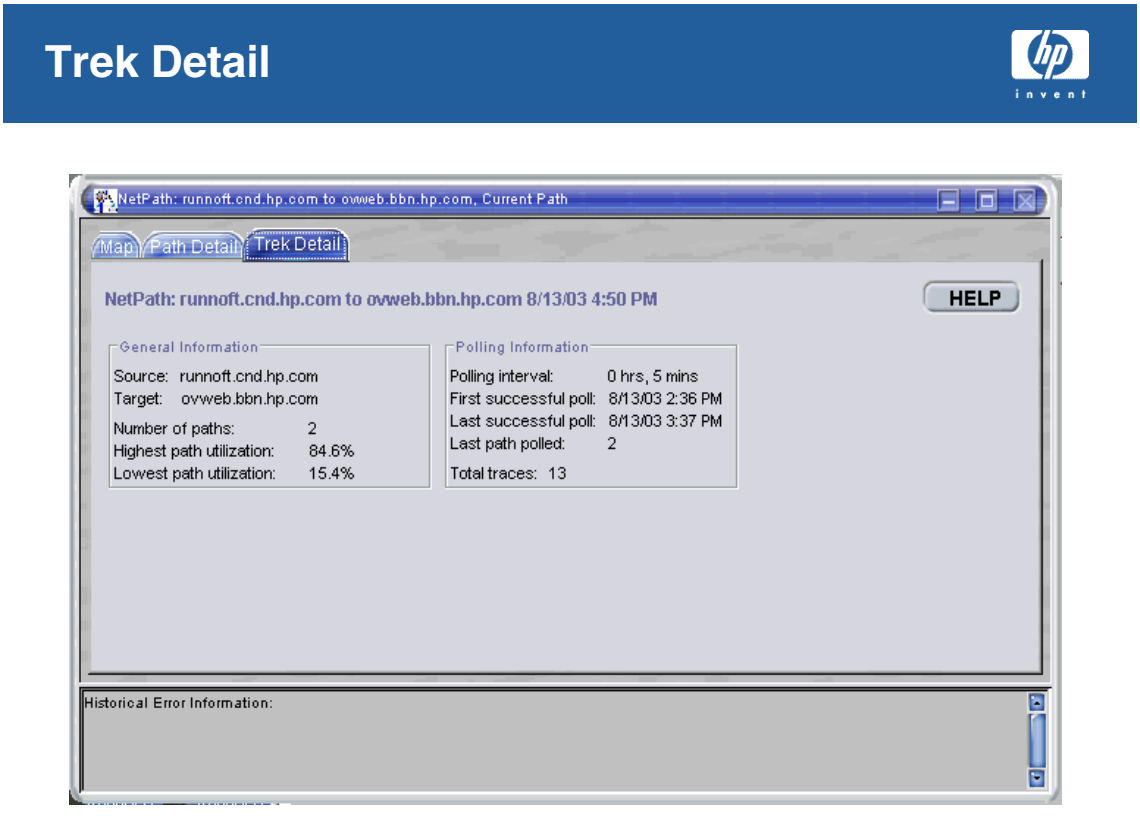
When you double-click the path whose Path ID is Current on the Path List and open the Path Detail tab, you get a special presentation, unique to the Current Path. Paths other than Current get a different presentation.

The table entries are as follows:

Item	Description
Hop	A simple identifier for a device on the path.
Device Name	The DNS name (or IP address, if a name is not available) of the device. If the type of device can be determined, it is denoted by the symbol.
Interface	The interface on the device which received the ping from the probe. For the first (Probe) node only, it is the interface that sent the pings.
Status	The current status of the device. This is the present status, evaluated by Problem Diagnosis when [GO] is pressed.

Item	Description
Current Response Time (ms)	Response time measures how fast a ping response returns when this particular device is pinged. Note that it includes the cumulative response times of intermediate devices. This column is subdivided as follows: • *#1 through #5*: When you click [GO], Problem Diagnosis issues a series of pings against each device in the path. These columns tabulate the results of those pings. UNIX-based probes issue a series of five pings, while Windows-based probes issue a series of three. So if columns #4 and #5 contain a hyphen (-), the probe is on a Windows host. • *Min*:The fastest response time of the series. • *Max*: The slowest response time of the series. • *Avg*: The average response time across the series.

Trek Detail



The Trek Detail panel presents general and statistical information about what the probe has learned about paths between itself and the current target (not all targets). Most of the fields are self-explanatory.

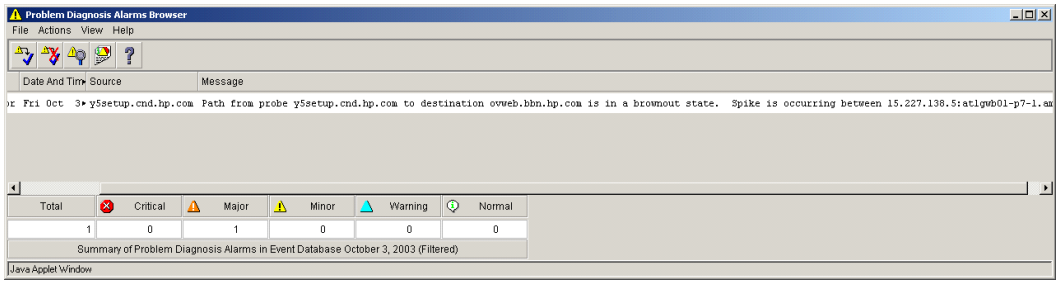
The probe adjusts the Polling interval upward if each probe attempt takes too long to finish, so that it does not get behind in polling.

The Last path polled entry gives the Path ID number of the path that was most recently tested.

Detecting Network Brownouts

Detecting Network Brownouts

- If probing takes statistically longer than normal for this path, send a brownout event to the Problem Diagnosis Alarm Browser.
  - Which probe
  - Which target destination
  - Where in the path the brownout is occurring



U5089S ©2004 Hewlett-Packard Development Company, L.P. 14

When Problem Diagnosis performs a trace from a probe to a destination, the packet round trip time to the destination is noted. If the packet time exceeds a calculated limit (or threshold), then Problem Diagnosis attempts to isolate where performance has spiked by pinging the destination once per minute for fifteen minutes, by default. Each packet round trip time is noted and compared to its threshold. By default, when eight or more of the fifteen times exceed the threshold, then a brownout event is generated.

Problem Diagnosis determines the point where a brownout might be occurring by looking at the nodes (or hops) along a path from the probe to the destination. When a hop time exceeds its calculated limit, then Problem Diagnosis assumes that a spike is occurring between that hop and the previous hop.

Brownout events are sent to the Problem Diagnosis Alarms category of the NNM alarm browser. The brownout alarm description contains the probe, the destination, and the two nodes that are most likely causing the brownout.

Lab Exercises

Lab Exercises

- In this lab you will
  - Select endpoints in Problem Diagnosis
  - View the list of paths between those endpoints
  - Look at statistics for the current path

©2004 Hewlett-Packard Development Company, L.P.

Extended Topology must be enabled for these exercises.

1. Start the Problem Diagnosis View from Home Base. Select your system as the Probe.

2. Add an endpoint to monitor by clicking [CONFIGURE].

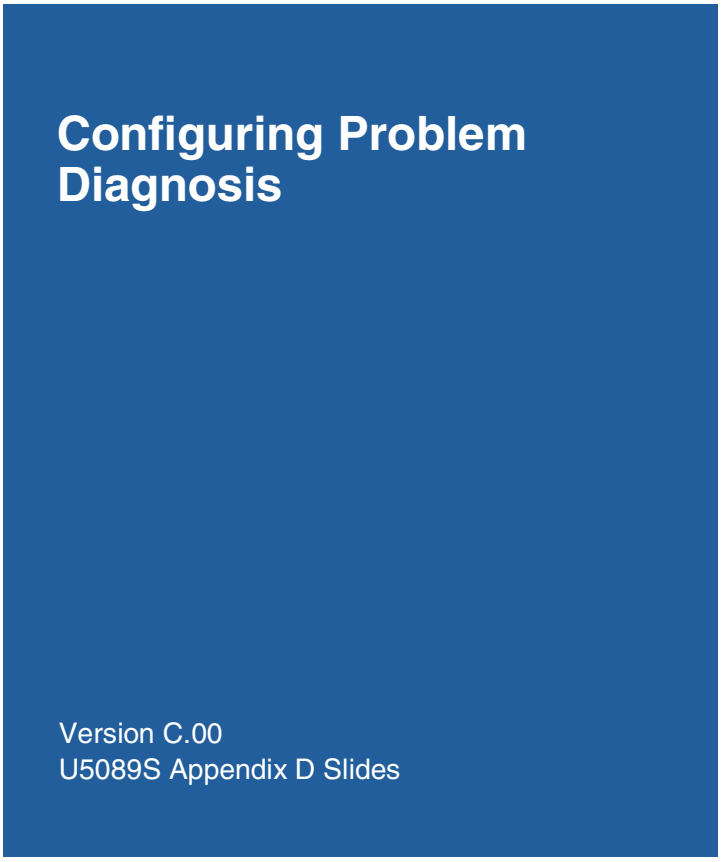
3. View the Path List.

4. View path information for the Current Path.

5. View the Trek detail.

D Configuring Problem Diagnosis

Module Objectives



At the completion of this module the student will be able to:

- Start and stop a Problem Diagnosis Server.
- Link a Server to a Probe.
- Link a Probe to a Server.
- Configure additional actions on the user interface popup menu.
- Change the server port.
- Install a probe on a network node.
- Configure a probe.
- Start and stop a probe.
- Uninstall a probe.

Server Tasks

- Starting and Stopping the Server
- Linking the Server to a Probe
- Configuring the Server Port

©2004 Hewlett-Packard Development Company, L.P.

See the *Release Notes* for a list of supported operating systems.

Installing the Server

- Installed with NNM
- Enabled with Extended Topology

©2004 Hewlett-Packard Development Company, L.P.

The Problem Diagnosis server allows you to view data from many probes. You need only one server in your environment.

When you install Network Node Manager and setup Extended Topology, a Problem Diagnosis server is automatically installed on the same system. Problem Diagnosis views are not available until Extended Topology is enabled.

Starting and Stopping the Server

- ovstart pd
- ovstop pd

 ©2004 Hewlett-Packard Development Company, L.P.

To start and stop the Problem Diagnosis server, do the following:

To start the Problem Diagnosis server, from a command prompt, enter:

- UNIX: `$OV_BIN/ovstart pd`
- Windows: `%OV_BIN%\ovstart pd`

To stop the Problem Diagnosis server, from a command prompt, enter:

- UNIX: `$OV_BIN/ovstop pd`
- Windows: `%OV_BIN%\ovstop pd`

NOTE This automatically starts and stops the probe on the Problem Diagnosis server. You do not need to start and stop the server probe separately.

WARNING **Do not force termination of the Java process (kill -9 on UNIX machines or [End Task] from a Windows Task Manager); doing so may cause irrevocable corruption of the server data. Do a full `ovstop` and `ovstart` if you experience difficulties.**

Linking the Server to a Probe

As you install a Problem Diagnosis probe, you assign it to a Problem Diagnosis server. The two transparently establish the configuration necessary to communicate, and the probe automatically shows up in the probe list on the server.

However, a Problem Diagnosis server can get path data from any probe, providing the server is configured to know about the probe.

There are two ways to establish communications between a server and a probe that was not initially assigned to that server.

- You can add the probe to the list of probes the server knows of. This method is most useful when you want a server to know about several probes that it could draw data from; that is, when you have one server that will use many probes. This is the method described below.
- Alternatively, you can configure the probe to notify the server of its existence, and let the server reconfigure itself. This method is most useful when you want a probe to know about several servers that it provides data to; that is, when you have one probe to be used by many servers. This approach is covered in the Linking the Probe to a Server topic.

Procedure

There are two steps to configure the server to contact a probe which was assigned to a different server when it was installed:

1. Manually edit the Problem Diagnosis configuration file, which defines which probe(s) the server can use. This file is located as follows (assuming a default installation):

UNIX `$OV_MAIN_PATH/pdAE/config/pdconfig.xml`

Windows *install_dir*`\pdAE\config\pdconfig.xml`

You can edit this XML file even if you have no experience with XML, but it requires accuracy; mistakes in the Problem Diagnosis configuration file can cause the Problem Diagnosis server to not work correctly with treks. HP strongly recommends that you make a backup copy of the Problem Diagnosis configuration file before editing, so that you can easily recover if necessary.

The Problem Diagnosis configuration file is created when a probe is installed and assigned to the server. Until that happens, there is no Problem Diagnosis configuration file. If you create the file yourself, be sure to include all the non-bold lines, as well as the desired Probe Definition, as described in this topic.

The Problem Diagnosis configuration file initially looks like this:

```
<?xml version= "1.0"  ?>
<PD_CONFIG>
  <PROBE_LIST>
      <PROBE>        <HOST_NAME> Icarus.naucrates.com </HOST_NAME>
        <IP_ADDRESS> 15.2.114.2 </IP_ADDRESS>
      </PROBE>
</PROBE_LIST>
</PD_CONFIG>
```

The four lines in bold text create a “Probe Definition”, which identifies Icarus as hosting a probe that the server can use. You can use more probes by adding additional Probe Definitions. Probe Definitions must fall between the `<PROBE_LIST>` and `</PROBE_LIST>` tags, and they cannot overlap. (Note that the tags are case sensitive.)

The port must be the port used by the server. The default is 8068; if this does not work, check the probe's `npprobe.conf` file to see what port it is configured to use.

You can use an HTTP proxy for the probe by configuring the `PROXY_SERVER` and `PROXY_PORT` in `pdconfig.xml`.

In the example file above, you can add access to another probe (`Daedalus`) by inserting the following lines in the Problem Diagnosis configuration file on the server:

```
<PROBE>  <HOST_NAME> Daedalus.naucrates.com </HOST_NAME>
  <IP_ADDRESS> 15.2.114.231 </IP_ADDRESS></PROBE>
```

The resulting file looks like this:

```
<?xml version= "1.0"  ?>
<PD_CONFIG>
  <PROBE_LIST>
      <PROBE>        <HOST_NAME> Icarus.naucrates.com </HOST_NAME>
        <IP_ADDRESS> 15.2.114.2 </IP_ADDRESS>
      </PROBE>
      <PROBE>
        <HOST_NAME> Daedalus.naucrates.com </HOST_NAME>
        <IP_ADDRESS> 15.2.114.231 </IP_ADDRESS>
      </PROBE>
</PROBE_LIST>
</PD_CONFIG>
```

2. After saving the Problem Diagnosis configuration file, stop and restart the server to activate the changes. Allow several minutes for the probe and server to synchronize.

   When you access the server, you can now choose to use either Icarus or Daedalus as the probe.

You can remove access to a probe's data (unlink the probe) by editing `pdconfig.xml` to remove the probe block.

Configuring the Server Port

Changing the Server Port

- NOT recommended.
- Default ports (8068 or 8067) may have been used by other software.

1. Stop the server.
2. Stop all probes that link to this server.
3. Edit **`pdconfig.xml`**.
4. Change any probes that link to this server to match the new port number.
   a. Edit **`npprobe.conf`** on the probe system.
   b. Restart the probe.
5. Restart the server.

©2004 Hewlett-Packard Development Company, L.P.

Why Would You Do This?

Ports 8068 and 8067 are used by the Problem Diagnosis server to communicate with the GUI and other clients (8068), and with probes (8067). If either of these ports has been taken by other software, the port(s) used by the server must be changed. In the case of port 8067, probes must be reconfigured to reflect changes to the server.

NOTE HP recommends that you make port changes only if it is absolutely necessary!

To Change from Port 8068 to Another Port

NOTE Any probe accessed by a server that does not use port 8068 must be reconfigured to accommodate this, and its data will not be available to any server that uses another (such as the default) port.

Follow this procedure:

1. Stop the Problem Diagnosis server and stop all probes that are assigned to that server.
2. Edit the Problem Diagnosis configuration file, looking for the port number “8068”. Replace this number with a port number that is not used by any other networking software.
3. For all relevant Problem Diagnosis probes, edit *install_dir*/netpath/config/npprobe.conf and replace “8068" with the same port number you used in the last step.
4. Restart the server, and restart all re-configured probes.

To Change from Port 8067 to Another Port

Follow this procedure:

1. Stop the Problem Diagnosis server.
2. Edit pdconfig.xml. For each <PROBE> entry, edit the entry <HTTP_PORT> 8067 </HTTP_PORT>. Change 8067 to your port number.
3. On the probe system edit:
   - UNIX: /etc/services
   - Windows: %SystemRoot%\system32\drivers\etc\services
4. Restart the server.

Configuring Brownout Detection

Configuring Brownout Detection

- Edit **pdconfig.xml** on the server.
- Default:
  - Threshold = mean + 3*SQRT(mean)
  - If probe time > threshold
    - Begin polling once per minute for 15 minutes
  - If more than 8 of those polls > threshold
    - Send event to **pmd**
- Configurable:
  - Number of deviations to determine the threshold (default is 3)
  - How many times to poll once per minute (default is 15)
  - How many of the re-polls must be over the threshold (default is 8)

©2004 Hewlett-Packard Development Company, L.P.

When Problem Diagnosis performs a trace from a probe to a destination, the packet round trip time to the destination is noted. If the packet time exceeds a calculated limit (or threshold, the `mean+ 3*(SQRT(mean)` by default), then Problem Diagnosis pings the destination once per minute for fifteen minutes, by default. Each packet round trip time is noted and compared to its threshold. By default, when eight or more of the fifteen times exceed the threshold, then a brownout event is generated. Problem Diagnosis does the statistical calculations based on "buckets" indicating various times of day. It does not use `snmpCollect` or any of `snmpCollect's` configuration.

To change the way brownout events are generated, you can modify the Problem Diagnosis configuration file on the server, `pdconfig.xml`, located at:

UNIX: `$OV_MAIN_PATH/pdAE/config/pdconfig.xml`

Windows: `%OV_MAIN_PATH%\pdAE\config\pdconfig.xml`

Then `ovstop pd` and `ovstart pd`.

You need to allow Problem Diagnosis to run long enough to collect enough data samples to be able to do statistical calculations.

The Problem Diagnosis configuration file contains five tunable brownout parameters, described in

the following table.

Tunable Parameter	Description	Default Value
BROWNOUT_INTERVAL	Time in milliseconds that Problem Diagnosis waits before generating another brownout event for a probe and destination combination.	86400000
BUCKET_SIZE	The number of measurements Problem Diagnosis stores for a particular hop from a particular path to the destination. After the value is reached, the oldest measurement is replaced by the newest measurement.	24
BROWNOUT_NUM_SAMPLES	Number of times that Problem Diagnosis attempts to ping the destination device. The frequency that Problem Diagnosis attempts to ping the destination device is once per minute.	15
BROWNOUT_NUM_DEVIATIONS	A number used to calculate the threshold for brownout events. The formula for determining the threshold is the BROWNOUT_NUM_DEVIATIONS times the square root of the mean, plus the mean.	3
BROWNOUT_BAD_SAMPLES	The number of times that the ping round trip time must exceed a threshold before a brownout event is generated.	8

Probe Tasks

Probe Tasks

- Installing a Probe
- Disabling a Probe
- Starting and Stopping the Probe
- Deploying Problem Diagnosis Probes
- Configuring a Probe
- Linking the Probe to a Server
- Troubleshooting a Probe

©2004 Hewlett-Packard Development Company, L.P.

Installing a Probe

Installing a Probe

- A probe is installed on the management station automatically.
- To install on additional systems:

1. FTP **`probeHP.tar`** (HP-UX) or **`probeWIN.zip`** (Windows) to the system.
2. Uncompress the file.
3. Run **`pdpinstall.sh`** (UNIX) or **`pdpinstall.vbs`** (Windows).
   a. Provide server fully-qualified DNS name.
   b. Provide server IP address.
   c. Provide server port number.

U5089S ©2004 Hewlett-Packard Development Company, L.P. 9

When you install Network Node Manager and setup Extended Topology, a Problem Diagnosis probe is automatically installed on the same system. After NNM is installed and Extended Topology is enabled, this probe is available for use in the Problem Diagnosis View main dialog. Additional probes may be installed on remote systems using an automated installation script.

Probe Installation

An automated installation package in the Network Node Manager distribution guides you through the installation process of remote Problem Diagnosis probes on HP-UX, Sun Solaris, or Microsoft Windows systems of your choosing. Consider installing probes in key locations in the network, where they can monitor crucial paths.

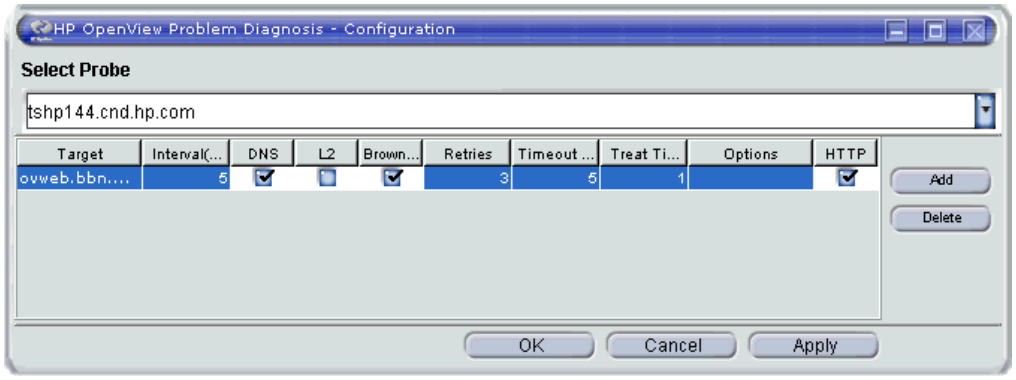
Platform	Instructions
HP-UX	1. From the Problem Diagnosis server system, enter: `cd $OV_MAIN_PATH/pdAE/bin` 2. FTP `probeHP.tar` to the system where the remote probe is to be installed. 3. From the remote probe system as superuser, `untar probeHP.tar` in the root directory. `tar -xvf probeHP.tar` 4. `cd /opt/OV/bin/pdAE/bin` 5. `./pdpinstall.sh`. This installs the probe software in `/opt/OV/bin/pdAE`.
Windows	1. From the Problem Diagnosis server system, `cd %OV_MAIN_PATH%\pdAE\bin` 2. FTP `probeWIN.zip` to the system where the remote probe is to be installed. 3. From the remote probe system, unzip `probeWIN.zip` in the `C:\` (root) directory. This unpackages the zip file to `C:\Program Files\HP OpenView`. 4. Go to: `C:\Program Files\HP OpenView\pdAE\bin`. 5. Double-click: `pdpinstall.vbs`. This installs the probe software in `C:\Program Files\HP OpenView\pdAE`.

During the installation of remote probes, you assign the probe to a Problem Diagnosis server by providing the following information:

- The fully-qualified DNS name of the assigned server.
- The IP address of the assigned server.
- The HTTP port of the assigned server. This is the port that the assigned server uses for Problem Diagnosis communications. The default is 8068, and HP recommends that you accept this default. If you must configure the Problem Diagnosis server to use a different port, follow the special instructions in Configuring the Server Port.

The Problem Diagnosis probe starts immediately, and is configured to run automatically at system startup.

Configuring a Probe



 ©2004 Hewlett-Packard Development Company, L.P.

Starting the Probe Configuration Utility

Problem Diagnosis includes a web-based probe configuration utility. To configure a probe, first start the probe configuration utility as follows:

1. Start the probe to be configured, and also a Problem Diagnosis server that uses this probe. These components must both be running before you can proceed.
2. Launch Home Base from your browser using the following URL: `http://hostname:7510/`.
3. From the Home Base user interface, select Problem Diagnosis View from the View drop down list and click `[Launch View]`. The Problem Diagnosis main dialog opens.
4. Click `[Configure]` to open the Problem Diagnosis Configuration window.

Using the Probe Configuration Utility

To set probe targets and collection parameters, use the probe configuration utility as follows:

1. In the “Select Probe” list, choose the location of the probe you want to configure. If the list is empty, no probes have been installed.
2. Use the [Add] button to create an entry in the targets list. Double-click in the Target field and edit it to contain the fully-qualified name of a destination node. Problem Diagnosis will trace the paths to this node. Edit other fields in the entry as necessary; see the Configuration Options table.
3. After adding all desired targets, click [Apply] or [OK]. Both buttons record your changes; [OK] closes the window, [Apply] keeps it open.

To delete probe targets, select one or more rows, and click [Delete].

This table gives the meaning of each field:

Table D-1 Probe Configuration Options Table

Name	Description	Default
Select Probe	A drop-down list of probes that have been installed. Select the one you want to configure.	First in list
Target	Specifies the DNS name or IP address of a target to which you want the probe to trace paths.	none
Interval (min.)	Specifies how often (in minutes) an attempt to trace the path should be initiated. The probe automatically increases this time if it determines that the interval is too small.	1
DNS Lookup	Turns reverse DNS lookup for intermediate devices on or off. Having DNS Lookup on can result in slower performance in certain environments, but the path results will include the names of intermediate devices, as well as their IP addresses. Turning DNS Lookup off may speed up performance, but the path results will show only the IP addresses of intermediate devices, and not their names.	on
L2	Turns on or off the use of Extended Topology data to determine the layer 2 devices between layer 3 nodes.	off
Brownout	Turns event generation and data collection for brownouts on or off.	on
Retries	Specifies how many times Problem Diagnosis attempts to retrieve a successful trace.	3
Timeout Limit	Specifies the number of consecutive timeouts that Problem Diagnosis will accpet before ending the trace attempt.	5
Treat Timeouts	Sets how timeouts are treated. HP recommends retaining the default value. • 1 = error • 2 = ignore • 3 = keep	1

Table D-1 Probe Configuration Options Table

Name	Description	Default
`Trace Option`	Trace Options are for probes hosted on UNIX systems only; must be left blank on a Windows system. For additional option details, see the traceroute reference (man) page. Specifies options to trace; individual options can contain no white space, but different options must be be separated by white space. Note that all options are case-sensitive ("`-p`" is not the same as "`-P`")! The potential options are as follows: • *-p[source port], [destination port]* Sets the source and destination ports. For example, `-p1234,20` specifies a source port of 1234, and a destination port of 20. For UDP-based traceroutes the ports are normally randomly selected. For TCP-based traceroutes the default source port is 57777 and the default destination port is 23 (the telnet port). • *-I* Performs an ICMP-based traceroute, instead of the default UDP-based traceroute. • *-m [max ttl]* Sets the maximum number of hops allowed in trying to reach the destination. The default is 30.	See specific option
`HTTP`	Use HTTP (on) or TCP (off) to communicate with the server.	`on`

Clicking the column headings in the table sorts the table on values in that column.

Starting and Stopping the Probe

•UNIX: **`$OV_MAIN_PATH/pdAE/bin/pdcentral.sh -start`**
(or **`-stop`**)

DO NOT use the kill -9 command on the Java process. Irrecoverable data corruption may occur.

•Windows: From the Services applet, select NetPath and click Start (or Stop).

DO NOT use the Windows Task Manager to terminate the Java process! Irrecoverable data corruption may occur.

NOTE If you are running an HP OpenView Operations product (OVO), ignore this topic. The OpenView Operations software controls starting and stopping the Problem Diagnosis probe (which is managed as a sub-agent).

On UNIX Platforms

NOTE The `pdcentral.sh` command must run inside the probe's `bin` directory.

To start the probe, execute the following commands:

- `cd $OV_MAIN_PATH/pdAE/bin`
- `./pdcentral.sh -start`Probe

To stop the probe, execute the following commands:

- `cd $OV_MAIN_PATH/pdAE/bin`
- `./pdcentral.sh -stop`Probe

WARNING **Do not force termination of the Java process (`kill -9`); doing so may cause irrevocable corruption of the probe data.**

On Windows Platforms

The probe is installed as a service, which runs automatically at system startup. (See Windows Services Note below.)

To manually stop or start the probe service:

1. Right-click on the `My Computer` desktop icon, and choose the `Manage` menu item.
2. In the navigator pane, double-click `Services and Applications`.
3. Double-click `Services`.
4. In the details pane, select the `NetPath` service, and click the `Start` button or `Stop` button, as desired, in the applet tool bar.

WARNING **Do not use the Windows Task Manager to terminate the Java process! Doing so may cause irrevocable corruption of probe data.**

Windows Services Note

On some systems, running the probe as a service has been seen to cause the screen-saver to stop working. This can be a security hazard if the computer will not lock when unattended. If this happens, you may want to run the probe in stand-alone mode rather than as a service.

To remove the probe from the Windows services (or add it back), see Disabling a Probe.

To run the probe in stand-alone mode, issue the following commands from a command window prompt (steps assume the default installation directory, and quotes are required):

1. Disable the probe. See Disabling the Probe.
2. From a command window prompt, enter: `cd %OV_MAIN_PATH%\pdAE\bin`
3. From a command window prompt, enter: `pdcentral.bat -startProbeNoSvc`

To restore the probe as a service, do the following on the probe system:

1. From a command window prompt, enter: `cd %OV_MAIN_PATH%\pdAE\bin`
2. From a command window prompt, enter: `pdcentral.bat -stopProbeNoSvc`
3. Restore the probe. See Restoring the Probe.

Disabling a Probe

•UNIX: move the file from the **`rc3.d`** directory.

•Windows: Use the Services applet to stop the NetPath service, then run **`pdcentral.bat -uninstall`**

©2004 Hewlett-Packard Development Company, L.P.

You can disable an installed probe, so that it does not run automatically at system startup.

UNIX

To disable the probe automatic startup, follow these steps:

After these steps, the probe will not automatically start on system startup.

HP-UX

1. `cd $OV_MAIN_PATH/pdAE/bin`
2. `./pdcentral.sh -stop`Probe
3. `mv /sbin/rc3.d/S750Netpath <savedRCfile>`

You can later restore the probe:

1. `mv <savedRCfile> /sbin/rc3.d/S750Netpath`
2. `cd $OV_MAIN_PATH/pdAE/bin`
3. `./pdcentral.sh -start`Probe

Solaris

1. `cd $OV_MAIN_PATH/pdAE/bin`
2. `./pdcentral.sh -stop`Probe
3. `mv /etc/rc3.d/S750Netpath <savedRCfile>`

You can later restore the probe:

1. `mv <savedRCfile> /etc/rc3.d/S750Netpath`
2. `cd $OV_MAIN_PATH/pdAE/bin`
3. `./pdcentral.sh -start`Probe

Windows

1. Right-click on the `My Computer` desktop icon, and choose the `Manage` menu item.
2. In the navigator pane, double-click `Services and Applications`.
3. Double-click `Services`.
4. In the details pane, select the `NetPath` service, and click on the `Stop` button in the applet tool bar.
5. Remove the probe from the Windows services. From a command prompt:
   - `cd %OV_MAIN_PATH%\pdAE\bin`
   - `pdcentral.bat -uninstall`

After this, the probe will not automatically start on system startup. Note that this does not stop an already running probe.

Removing the probe from the Windows services does not remove the software from the system, and you can restore the probe's automatic startup using this command:

1. Reinstall the probe as a Windows service. From a command prompt:
   - `cd %OV_MAIN_PATH%\pdAE\bin`
   - `pdcentral.bat -install`
2. Reboot, or
   - Right-click on the `My Computer` desktop icon, and choose the `Manage` menu item.
   - In the navigator pane, double-click `Services and Applications`.
   - Double-click `Services`.
   - In the details pane, select `Netpath` service, and click the `Start` button in the applet tool bar.

Linking the Probe to a Server

Linking the Probe to a Server

- Use when a probe may report to multiple servers.
- Edit **`npprobe.conf`** on the probe system.
  - Modify existing lines
- Stop and restart the probe.
- A probe uses the same port to talk to all servers.

©2004 Hewlett-Packard Development Company, L.P.

In a default installation, a Problem Diagnosis probe is assigned to serve a single Problem Diagnosis server.

However, a probe can be used by multiple Problem Diagnosis servers.

Conversely, a Problem Diagnosis server can use multiple probes.

Notice, too, that a host can be the target of more than one probe.

And of course you can use any server by merely entering its URL in your web-browser.

As you install a probe, you assign it to a Problem Diagnosis server. The two transparently establish the configuration necessary to communicate, and the probe automatically shows up in the probe list on the server.

However, a probe can provide any Problem Diagnosis server with path data, if the server knows about the probe.

There are two ways to establish communications between a server and a probe that was not initially assigned to that server.

- You can configure the probe to notify the server of its existence, and let the server reconfigure itself. This method is most useful when you want a probe to know about several servers that it could send data to; that is, when you have one probe to be used by many servers.
- Alternatively, you can manually add the probe to the list of probes that the server keeps. This

method is most useful when you want a server to know about several probes that it could draw data from; that is, when you have one server that needs to use many probes.

Procedure

There are two steps to make a probe (which was assigned to one server when it was installed) notify another server of its existence:

1. Manually edit the probe configuration file (on the probe system), which names the server it should notify when it initializes. The probe configuration file is located as follows:

UNIX: `$OV_MAIN_PATH/pdAE/config/npprobe.conf`

Windows: *`install_dir`*`\pdAE\config\npprobe.conf`

Editing this file requires accuracy, because mistakes in the probe configuration file can cause the probe to not work correctly. HP strongly recommends that you make a backup copy of the probe configuration file before editing, so that you can easily recover if necessary.

The uncommented part of the probe configuration file looks like this (assuming a Windows host):

```
SERVER=Minotaur.naucrates.com

SERVER_IP=12.12.121.212

SERVER_PORT=8068
```

The lines identify the Problem Diagnosis server, which was the server specified when the probe was installed. When the probe initializes, it notifies the server specified here of its existence. If necessary, the server adds the probe to its list of known probes.

You can have the probe notify additional servers of its presence (one at a time) as follows:

Change the lines in the probe configuration file to identify another server. In the example file above, we can notify another Problem Diagnosis server that the probe exists by changing the bolded lines in the probe configuration file to give the hostname and IP address of the new server, as follows:

```
SERVER=Aeolus.naucrates.com

SERVER_IP=12.12.112.121
```

IMPORTANT You modify the existing lines; do not add any additional lines! The resulting file looks like this (again, on a Windows host):

```
SERVER=Aeolus.naucrates.com

SERVER_IP=12.12.112.121

SERVER_PORT=8068
```

Important: A probe can only communicate with servers that use the port defined by that probe (8068 in this example). If you change the port used by a server, you have to reconfigure each probe that the server uses (via the SERVER_PORT definition in npprobe.conf) to use the new port.

2. After saving the changes, stop and restart the probe. This causes the probe to synchronize with its server. Because Aeolus has no previous knowledge of the probe, it adds the probe to its `pdconfig.xml` file (which is described in the Linking the Server to a Probe topic). After the probe is added to the list of known probes on a server (like Aeolus), it remains on that list even

if you repeat this process to add it to yet another server.

Allow a few minutes for the probe and server to synchronize.

At this point, both of the servers (Aeolus and Minotaur) can use the probe to show path data.

Troubleshooting a Probe

- Messages go to **`npprobe.log`**.
- Open a web browser and
  - Test the path between the pd target and the remote probe

  or
  - Try to talk directly to the probe

©2004 Hewlett-Packard Development Company, L.P.

Messages and errors from the probe are logged to `<PD_install_directory>/log/npprobe.log`, which is always available for examination.

To troubleshoot a remote probe, open a web browser and 1) test the path between the pd target and the remote probe or 2) try to talk directly to the probe.

1. Test the path between the probe and the probe target.
   a. Open any browser.
   b. If you have not already configured a probe target for the probe using Dynamic Views, do so now. Go to <NNM server system>:7510. (or Home Base). Select Problem Diagnosis View and click Launch View. Click Configure and add a probe target (such as your PD server system).
   c. Go to :8068/central/central.req?destination=|<probe_target"http://<pd_server_name>:8068/central/central.req?destination=<probe_system>|<probe_target>
   d. On Netscape, you may not see any text in the window. If not, click View:Source from the Netscape main menu.
      The output in the browser is XML output that describes the current path between the probe and target. If you do not see path information, such as <Path ID> and so on, then you know you have a problem.

2. Talk directly to the probe using any web browser -- see if you can connect to the probe.

   a. Go to :8067/netpath/netpath.req?destination=<probe_target"http://<Probe_system>:8067/netpath/netpath.req?destination=<probe_target>

   b. On Netscape, you may not see any text in the window. If not, click View:Source from the Netscape main menu.

      The output is again in XML format that describes the NetPath information, such as the hop list.

Additional information on troubleshooting is available in *Managing Your Network with Network Node Manager*.

Uninstalling Problem Diagnosis Software

- UNIX: **pdpuninstall.sh**
- Windows: **pdpuninstall.vbs**

- These instructions remove ALL components
  - Probe and Server go together
  - Otherwise just disable the probe

©2004 Hewlett-Packard Development Company, L.P.

Uninstalling Problem Diagnosis Software

You can easily remove Problem Diagnosis components from a system (the server and probe at once). The method you use depends on your operating system.

UNIX	1. `$OV_MAIN_PATH/pdAE/bin/pdpuninstall.sh`
Windows	1. `cd %OV_MAIN_PATH%\pdAE\bin` 2. `pdpuninstall.vbs`

IMPORTANT The above removes all Problem Diagnosis components. You cannot remove the probe and keep the server, or remove the server and keep the probe. You can, however, disable probe functionality and keep the server active. There is no need to actually remove the probe or its data (which together occupy under a megabyte of storage).

Lab Exercises

Lab Exercises

- Link the server to an additional probe.
- Link the probe on the management system to a remote server.

©2004 Hewlett-Packard Development Company, L.P.

Extended Topology must be enabled to use Problem Diagnosis. If you have not already done so, run `setupExtTopo.ovpl`. Answer yes to all questions and use “ov” for the requested user and password.

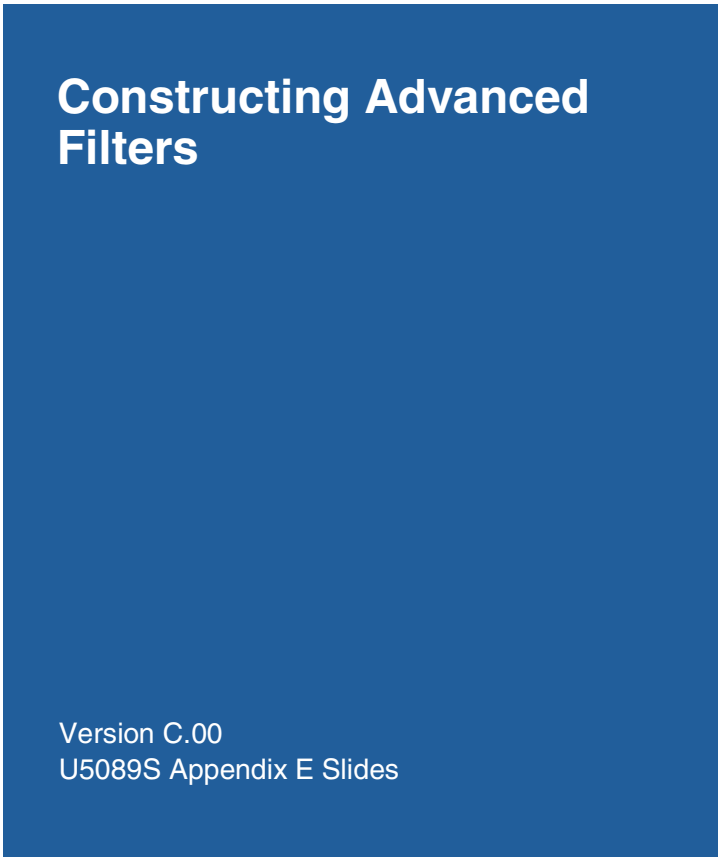
1. Stop and restart the server

2. Link the server to a probe on another system

3. Link this system’s probe to another server

E Constructing Advanced Filters

Module Objectives



At the completion of this module the student will be able to:

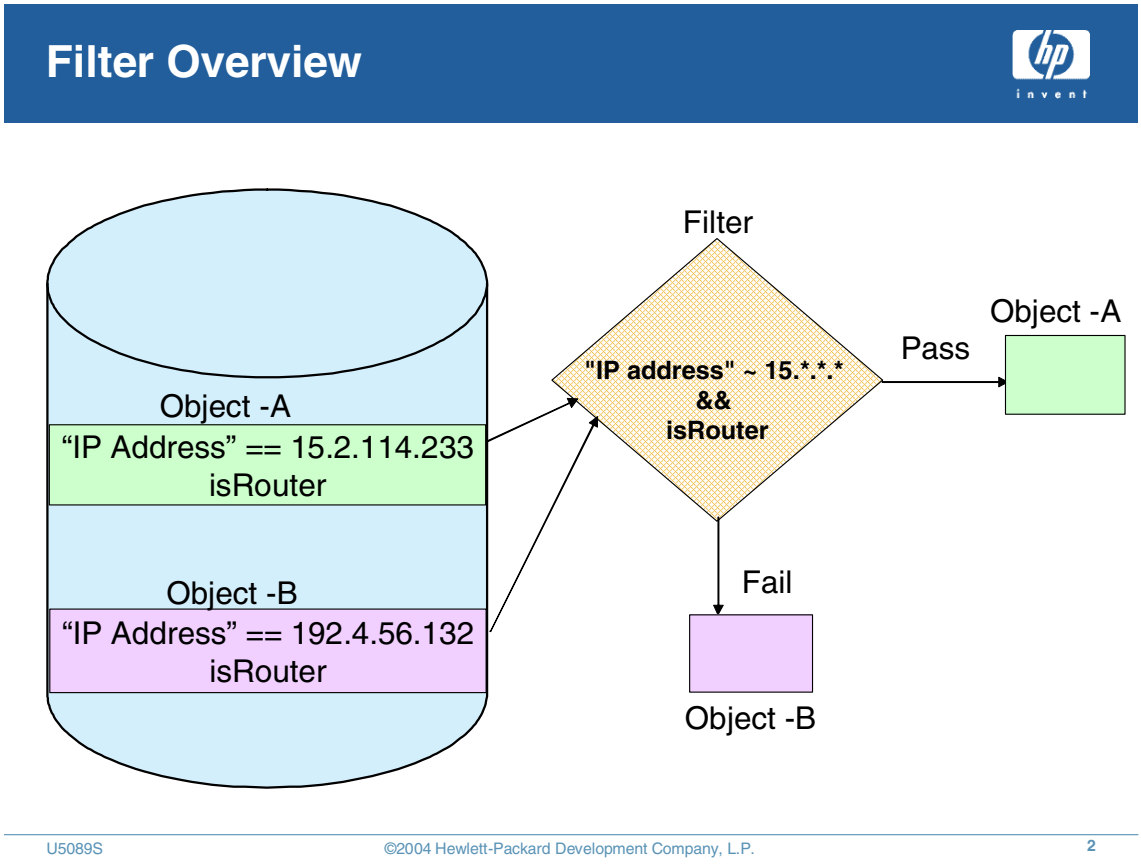
- Identify the relationship between objects and filtering.
- Identify the primary NNM filters and where each is used.
- Read and write complete filters in the `filters` file, including
  - — Sets.
  - — Attribute Value Assertions.
  - — Pattern Matching.
- Test filters.

References

Filter construction and examples are covered in detail in *A Guide to Scaling and Distribution for Network Node Manager*. You may also review the following online NNM reference pages (manpages on UNIX):

- *ovfiltercheck*
- *ovfiltertest*
- *ovtopodump*
- *OVfilterIntro*

Filter Overview

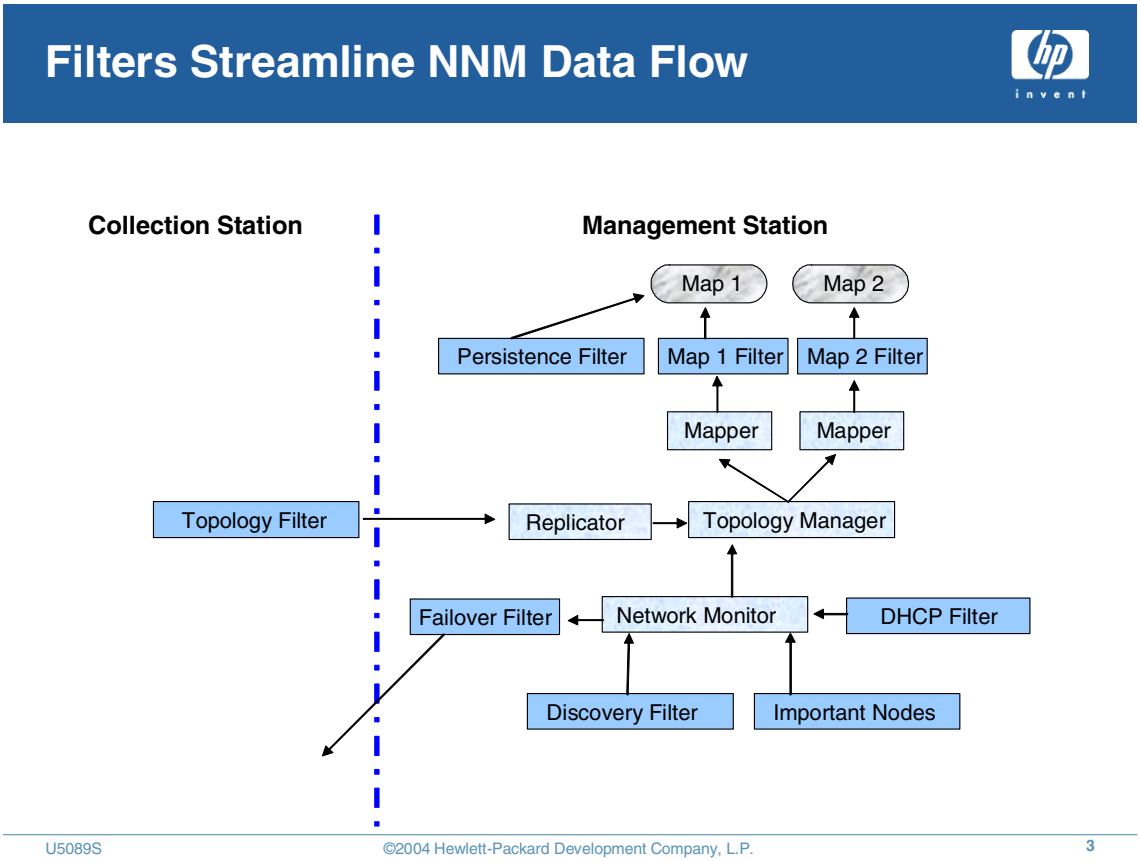


Filters are used to streamline processing and eliminate clutter on operator displays. Filters establish a specific subset of objects for a particular application or process with the intention of eliminating extraneous information.

Filters define the criteria by which objects pass through to an application or are removed from the data stream. The filter logic determines whether an object is accepted or rejected, based on the object’s attributes. If the object’s attributes match the attributes defined by the filter, the object passes. Conversely, an object is rejected if no match is found or if the attribute tested is not set.

This elimination of object data can result in a savings of memory and hard disk space, and improved performance. This allows you to scale down your management platform by eliminating unwanted overhead.

Filters Streamline NNM Data Flow



Filtering refers to the process of limiting the objects you are interested in discovering, monitoring, or displaying based on certain attributes. Filtering generally is used to reduce system load by filtering the amount of data processed by applications and reducing the amount of unused data flowing through the system.

The major types of filters are:

- Discovery Filter - Limits what `netmon` discovers.
- Topology Filter - Limits what topology information is forwarded from a collection station to a management station.
- Map Filter - Limits what is shown on the map.

In addition, you can use filters to control status polling intervals, DHCP environment handling, which submaps remain in memory, nodes for special treatment if a connector device fails, which nodes have data collected upon them, and which nodes are shown in graphical utilities.

The filters listed above represent the primary use of the NNM filtering mechanism. From the viewpoint of an administrator, the most important filters are the map, discovery and DHCP filters. It is important to note that NNM can function just fine as a stand-alone management system without using any filters.

Filters can be applied to an application or process in various ways; that is, each filter is not applied in the same manner. You can apply a filter in three possible places: 1) to an LRF file, 2) to

a registration file, and 3) via a pull-down menu.

The following list outlines where you configure a given type of filter:

- *Discovery filter*. Select `Options:Network Polling Configuration:IP`.
- *Topology filter.* Applied to the `ovtopmd.lrf` file. Run `ovtopmd -f` *`filtername`*.
- *Map filters.* Select `Map:Properties` or via direct modification to the `ipmap` registration file.
- *Persistence filters.* Select `Map:Properties` or via direct modification to the `ipmap` registration file.
- *Failover filters*.Run xnmtopoconf `-failoverFilter` *`filtername`*.
- *Important Node filter*. Select `Options:Network Polling Configuration:IP`, and the Secondary Failures area, or run `xnmpolling`.
- *DHCP filters*. Select `Options:Network Polling Configuration:Status Polling`, then select the appropriate DHCP blocks.
- *Data Collection filters*. Select `Options:Data Collection and Thresholds`.
- *Status Polling filters*. Select `Options:SNMP Configuration`, [`Poll Objects`].
- *Utility filters*. Select filters in the user interface for the utility.

Using Object Attributes

Using Object Attributes

```
ovobjprint -s "cnd-gw1"
FIELD IDFIELD       NAME                    FIELD VALUE
10                  Selection Name               "cnd-gw1"
11                  IP Hostname                  "cnd-gw1"
14                  OVW Maps Exists              1
15                  OVW Maps Managed             1
17                  vendor                       cisco Systems
27                  isNode                       TRUE
29                  isComputer                   TRUE
30                  isConnector                  FALSE
32                  isRouter                     TRUE
52                  isIP                         TRUE
53                  isIPX                        FALSE
517                 IP Status                    Normal (2)
521                 isIPRouter                   FALSE
555                 isSNMPSupported              FALSE
562                 SNMPSAgent                   Unset (0)
572                 TopM Interface Count1
579                 TopM Interface List "15.2.112.1 Normal  15.2.112.1  255.255.248.0
<None>                    0x00602FFFD581  other
581                 isCollectionStationNodeFALSE
```

(partial listing due to space constraints…)

 ©2004 Hewlett-Packard Development Company, L.P.

The example in this slide lists the output of the query on the object database for an object whose selection name is `cnd-gw1`. This example illustrates a wide variety of attributes and values associated with a particular object.

Certain objects such as `IP Hostname` return an ASCII string; whereas other objects such as `OVW Maps Exists` returns a numerical value.

For example, if you wanted to positively identify `cnd-gw1`, you could specify a logic which looked for objects identifying the `IP Hostname` as `cnd-gw1`. Or you could positively match this object by specifying all objects whose `vendor` is `cisco Systems` and `isRouter` attribute is set to `TRUE`.

There are situations when an object's value for an attribute identified by a filter is `Unset`. This situation is discussed in depth later; however, it should be pointed out that if a filter encounters an object that does not have the specified attribute set, it automatically rejects the object. For example, if the object in this slide were to be evaluated by a filter looking for the attribute SNMPAgent, the object would be rejected (in other words, evaluated to false) because that particular attribute is not set.

NOTE Remember that all objects in the database have attributes. This includes elements such as "IP Internet" and Network objects. The attributes on these objects can be used in filters also.

Objects

The universal set of objects for a specific managed domain is represented by all IP- and SNMP-managed devices seen by the management station. Assuming that no discovery filters are set, this universal set should be represented as all objects in the HP OpenView database. You can view these objects using `ovobjprint`.

Each object has a set of attributes with which it is associated. Attributes' values are gathered during the discovery process via an SNMP query to a device's MIB.

Filterable Objects & Attributes

The following is an abbreviated list of valid objects and attributes for filtering. This list is intended to assist in learning the key aspects of filters:

- Node objects:
  - isATM (boolean)
  - isCDP (boolean)
  - isConnector (boolean)
  - isDS1 (boolean)
  - isDS3 (boolean)
  - isFrameRelay (boolean)
  - isHub (boolean)
  - isIP (boolean)
  - isIPRouter (boolean)
  - isNode (boolean)
  - isRMON (boolean)
  - isRMON2 (boolean)
  - isRouter (boolean)
  - isSNMPSupported (boolean)
  - isSonet (boolean)
  - vendor (enumerated)
  - IP Address (string)
  - IP Hostname (string)
  - IP Status (enumerated)
  - Selection Name (string)
  - SNMP sysDescr (string)
  - SNMP sysObjectID (string)

- Segment objects:
  - isBusSegment (boolean)
  - isStarSegment (boolean)
  - isSegment (boolean)
  - IP Segment Name (string)
  - IP Status (enumerated)
- Network objects:
  - isIP (boolean)
  - isNetwork (boolean)
  - IP Address (string)
  - IP Subnet Mask (string)
  - IP Status (enumerated)
  - IP Network Name (string)
- Interface objects:
  - isInterface (boolean)
  - isIP (boolean)
  - IP Address (string)
  - IP Status (enumerated)
  - IP Subnet Mask (string)
  - SNMP ifPhysAddr (string)
  - SNMP ifType (enumerated)

Filtering Customized Attributes

Attributes added by applications other than HP OpenView are *not* filterable, even if the application or administrator places values in the database. The NNM filter engine only filters on the values of fields listed in Appendix A of *A Guide to Scaling and Distribution for Network Node Manager*.

Looking Inside the filters File

Looking Inside the `filters` File

- Located in
  - **`$OV_CONF/C/filters`** (UNIX)
  - **`%OV_CONF%/C/filters`** (Windows)
- Each **`filters`** file block serves a specific function:
  - Sets — A list of elements that is used for evaluation by one or more Filters.
  - Filters — The primary logic in the filtering mechanism and the only required block. Filters may reference Sets.
  - FilterExpressions — A combination of Filter elements that have already been defined.
- These blocks must occur in this order in the **`filters`** file.

©2004 Hewlett-Packard Development Company, L.P.

The terminology surrounding the use of filters can be somewhat confusing. For HP OpenView, there is one, and only one, `filters` file. However, within this one file there may be many filter definitions. For each aspect for which filtering is desired, there is a filter. All of those filters together make up the `filters` file.

The overall file has a specific definition for its syntax. This syntax is defined by the Backus-Naur Form (BNF) notation. From that definition are derived the three basic entities of the filters file:

- Sets
- Filters
- FilterExpressions

These entities may be combined in a number of ways to form the wide variety of filtering that may be desired.

Sets

Sets are a list of elements that are string items such as IP Hostnames. These lists are used by filters to aid in the evaluation of a particular object. Sets are not required for filters to function in the NNM environment, but they make filter definitions much less complex.

The Sets keyword and open brace denote the start of all of the set statements that will be used. The matching close brace defines the end of the Sets block of definitions.

Filters

Filters are the primary element of logic behind all filter operations. They are also the only required element. A fully functional `filters` file only needs to contain one filter statement. In most cases, however, many filter statements are needed to complete numerous filtering tasks in NNM.

The Filters keyword and open brace denote the start of all of the filter statements that are used. The matching close brace defines the end of the Filters block. The Filters block must come after the Sets block.

FilterExpressions

FilterExpressions complete the building blocks of the `filters` file by allowing you to combine a number of filter statements together to form more complex filter logic. While FilterExpressions are not required, they greatly simplify your ability to implement filtering mechanisms successfully. FilterExpressions can only combine filter statements that have already been defined, which is why Filters comes before FilterExpressions in the `filters` file.

The FilterExpressions keyword and open brace denote the start of the FilterExpression statements to be used. The matching close brace defines the end of the FilterExpressions block.

Defining Object Sets

Defining Object Sets

•A set is a list of strings.

•String attributes in a set:

- Sets can explicitly reference a host or description strings.
- Sets can be a file on the local host.
- Sets can reference both member strings and files.

©2004 Hewlett-Packard Development Company, L.P.

In addition to testing an attribute against a literal constant, string-valued attributes can be tested for membership in a set. A set is simply a list of strings.

Set members may be enumerated in the filter file itself, or may be listed in a separate file. Referencing a list in a file is often more practical. It tends to keep the filter file less complicated and thus, easier to document and modify.

The only operation available on sets is the test for membership. Any set member beginning with a forward slash (/) is assumed to be the name of a file where set members are listed, one per line.

Windows filter files must use the forward slash. You may include a drive specification.

UNIX Example:

```
Sets {
CriticalNodes "" { "ovt1", /etc/opt/OV/share/conf/critical.nodes }
BackboneNodes "" { /etc/opt/OV/share/conf/backbone.nodes }
}
```

Windows Example:

```
Sets {
CriticalNodes "" { "ovt1", C:/install_dir/conf/criticalNodes.txt }
BackboneNodes "" { C:/install_dir/conf/backboneNodes.txt }
}
```

In this example, the set `CriticalNodes` is comprised of the host `ovt1`, and all the hosts in the file

`/etc/opt/OV/share/conf/critical.nodes` (*`install_dir`*`\conf\criticalNodes.txt` on Windows).

The `BackboneNodes` set definition defines the set of objects located in the file `/etc/opt/OV/share/conf/backbone.nodes` (*`install_dir`*`\conf\backboneNodes.txt` on Windows).

Depending on how your environment resolves names, it may be necessary to specify the fully-qualified hostname, for example, `ovt1.cnd.hp.com` instead of just the hostname `ovt1`.

Attribute Value Assertions

- Statement about an object's value for an attribute.
- Objects are evaluated against the AVA.
- Result is either True or False.
- AVAs are available for use in filters:
  - Boolean AVAs - {**isRouter** == **TRUE**}
  - Integer AVAs - {"**TopM Interface Count** "> **1**}
  - Enum AVAs - {**vendor** == "**SynOptics**"}
  - String AVAs - {"**IP Hostname**" == "**gomez**"}

©2004 Hewlett-Packard Development Company, L.P.

An Attribute Value Assertion (AVA) is a condition that an object either does or does not meet. It is defined in terms of fields of an object and possible values of those fields. For example, `{ isRouter == TRUE }` is a filter which is only true for routers.

Fields

Fields used in creating filters are those fields contained in the `ovwdb` object database. The topology fields are mostly also acceptable in filters.

Some fields in the `ovwdb` object database have embedded white space. Such a field must be enclosed in quotes when specified in a filter. "IP Hostname" is a good example. Parentheses are used to indicate explicit precedence.

Attribute Value Assertions can be tested only against literal constant values used in the filter expression. The attributes of two different objects cannot be tested against each other.

How the AVAs Evaluate

Three possible results can occur when evaluating an AVA:

1. The object does not have that attribute set (value is "Unset")and evaluates false.
2. The object has the attribute set, but it does not match the filter value and evaluates false.
3. The object has the attribute set, and it matches the one in the filter and evaluates true.

If an object's attribute is unset, then the statement evaluates to false. For example, if you state { "IP address" ~ 192.189.127.5 }, all objects evaluated that do not have IP address as an attribute, such as a segment, do not pass the filter. This can be particularly problematic when implementing map filters.

Boolean AVAs

Boolean AVAs can be expressed with just an attribute name (with an optional leading NOT operator), or can be explicitly tested against the keyword values TRUE and FALSE. For example, testing for an object which is a router can be expressed in two ways:

{ isRouter } or { isRouter==TRUE }.

The inverse expression (in other words, testing for a nonrouter) can be expressed also:

{ ! isRouter } and { isRouter==False }.

Integer AVAs

Integer AVAs can be tested for equality, inequality, greater than, greater than or equal to, less than, and less than or equal to (==, !=, >, >=, <, and <=). For example, to test for a multihomed node, an expression using an integer AVA might be written:

{ numInterfaces > 1 }

Enum AVAs

In the ovwdb object database, enum-valued attributes have their enumerated values expressed explicitly. In enumerated AVAs, these same enumerated values are used. For example, to test whether a device is for a particular vendor, the following AVA might be used:

{ vendor == "SynOptics" }

Enum AVAs can only test for equality or inequality.

Because the integer value of a given enum value might change over time, *enum strings* must be used in enum AVAs, and not their integer equivalents.

String AVAs

String AVAs can be tested using any lexical operator (==, !=, >, >=, <, and <=). Strings are

tested using the `strcoll(3)` function, and thus take advantage of any natural language semantics provided by that function. For example, testing for a particular host might use the following AVA:

```
{ "IP Hostname" == "ovt1" }
```

In addition to lexical comparison, regular expression matching is also available. In order to distinguish between intended regular expression matching and literal expression matching, the `LIKE` operator (~) must be used. All regular expressions must be contained in double quotes (""). Thus the following example AVA matches anything with the substring `ovt5` in it:

```
{ "IP Hostname" ~ ".*ovt5.*" }
```

while the following matches only.`*ovt5.*`:

```
{ "IP Hostname" == ".*ovt5.*" }.
```

AVA Operators

•Several operators are available to be used in AVAs:

- equality (==) - {**isRouter == TRUE**}
- inequality (!=) - {**"TopM Interface Count" != 1**}
- less than (<) - {**"TopM Interface Count" <5**}
- greater than or equal (>=) - {**"TopM Node Count" >= 100**}
- less than or equal (<=) - {**"TopM Network ID" <= 300**}
- like, regex (~) - {**"IP Address" ~ 15.4.23.***}
- not like, regex (!~) - {**"IP Hostname" !~ "hp.com"}**
- in set (**in**) - {**"IP Hostname" in CriticalNodes**}

©2004 Hewlett-Packard Development Company, L.P.

Equality and Inequality

Integers, strings and enumerated values can be tested for equality or inequality. Strings compared for equality must provide an exact match.

Less Than and Greater Than

Integers and strings can be tested for relative placement using less than (<), less than or equal to (<=), greater than (>), and greater than or equal to (>=). Integer comparisons follow numerical order. String comparisons use alphabetical order according to the `strcoll` function of the C language.

Like and Not Like

Strings can be tested for regular expression matching using the like (~) and not like (!~) operators.

This allows wildcarding in fields such as IP Hostname and IP Address.

In

Sets are called using the in operator (in) to determine if a value is listed in an enumerated set definition.

Building Filter Expressions

Building Filter Expressions

- Individual filters define how an object's attributes are evaluated when the filter is applied.
- Filter expressions combine multiple, previously-defined filters.
  - && for "and"
  - || for "or"
  - may use parentheses
  - ! for "not"

filter_label **"Descriptive comment" {*attribute operator value*}**

©2004 Hewlett-Packard Development Company, L.P.

Structure

Each elementary entry in a filter contains a name for a filter, a description of its purpose (ASCII string enclosed in quotes), and a statement about the value of an attribute relative to a particular object.

The complete filter file grammar is presented in BNF format in *A Guide to Scaling and Distribution for Network Node Manager*, located in Appendix A.

Filters and Filter Expressions

A filter can be a simple attribute assertion `{ isRouter == TRUE }`, several attributes combined in a Boolean expression `{ isBridge == TRUE || isHub == TRUE }`, or a combination of filters into a filter expression: `Connectors "" { Routers && Level2Conn }`.

Examples

```
Filters {
Routers "" { isRouter == TRUE }
Level2Conn "" { isBridge == TRUE || isHub == TRUE }
}
FilterExpressions {
Connectors "" { Routers || Level2Conn }
}
```

For usage purposes, filters and filter expressions are treated exactly the same. In other words, anywhere a filter can be used, a filter expression can also be used.

The only valid operands in a filter expression are *previously defined* filters. Filters cannot be added to an existing filter file via inclusion (in other words, external reference to another or a different filters file).

Filter expressions have three valid operators: `&&` (and), `||` (or), and ! (not).

Applying Filters or FilterExpressions

Whenever you specify a filter to be used by NNM, whether through the `ovw` user interface or through a utility, such as `xnmtopoconf`, you can only specify one filter name. This means that all of the logic that you intend to have for a functional filter must be represented by one name. In many cases, the name you use is actually a filter expression since you can combine multiple filters into one filter expression. Whether you use a filter name or a filter expression name, the names are case sensitive.

Special Pattern Matching in NNM Filters

Special Pattern Matching in NNM Filters

- Applicable to IP addresses and SNMP OIDs in filters file.
- Make set specification easier.
- Wildcards.
- Ranges.

©2004 Hewlett-Packard Development Company, L.P.

Wildcard Examples

Ranges can be established using a dash (-).

```
{ "IP Address" ~ 192.189.127.1-10 }
```

Commas may not be used.

Wildcards can be used to specify a set of objects in either an IP address such as:

```
{ "IP Address" ~ 192.189.127.* }
```

or in an SNMP OID, for example:

```
{ "SNMP sysObjectID" ~ .1.4.5.13.* }
```

NOTE The special pattern *should not* be enclosed by "" (double quotes), or it will be evaluated as a regular expression.

Correct:

```
{ "IP Address" ~ 23.6.1.* }
{ "SNMP sysObjectID" ~ .1.4.5.13.* }
```

Incorrect:

```
{ "IP Address" ~ "23.6.1.*"}
```

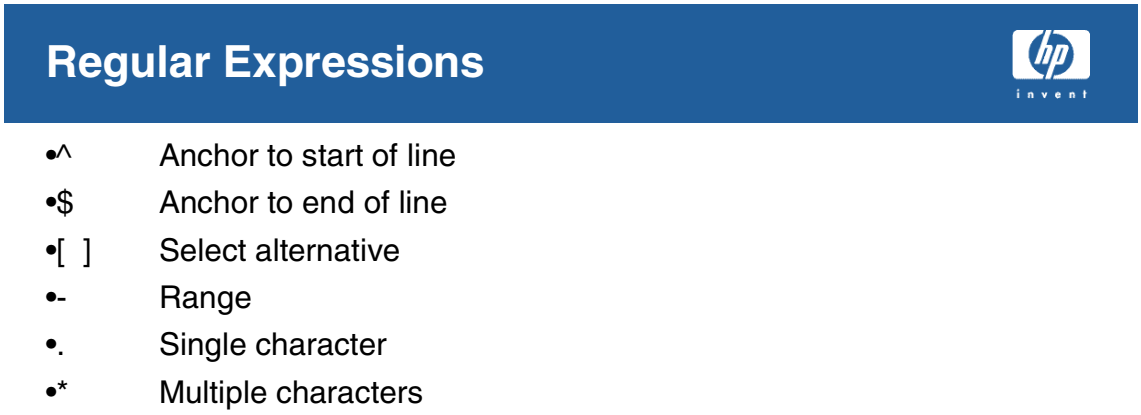
```
{ "SNMP sysObjectID" ~ ".1.4.5.13.*" }
```

These incorrect examples are legal, but lead to different, and probably unexpected, results since they are interpreted as regular expressions.

When using pattern matching on IP addresses and SNMP OID strings, *only* the (~) `LIKE` operator *or* the (!~) `NOT LIKE` operator can be used.

CAUTION When negating the "is like" operator (~), place the negation *outside* of the expression. This is necessary due to NNM discovering all level 2 interfaces, including objects like serial cards which have no IP address. When no IP address is present, the object is assigned 0.0.0.0. If you negate the operator, as in `{ "IP Address" !~ 192.189.127 }`, you will inadvertently *include* all non-IP level 2 interfaces. By negating the expression, as in `{ !("IP Address" ~ 192.189.127) }`, you get the set of all objects that *have* an IP address, and it is not like the one in the expression.

Regular Expressions



A regular expression (RE) is a mechanism for locating and manipulating patterns in text.

When you enclose the string pattern in double quotes, NNM treats it as a regular expression. You may use the following special characters to assist in the definition of your pattern matching requirements:

- ^ anchor to beginning of string
- $ anchor to end of string
- [ab-m] alternatives (ranges)
- . match a single character
- \ escape the special meaning of the following character
- * match 0 or more of the preceding regular expression

Matching a Single Character

The following REs match a single character:

Ordinary Characters An ordinary character is an RE that matches itself. An ordinary character is any character in the supported character set except <newline> and the regular expression special characters listed in Special Characters below. An ordinary character preceded by a backslash (\) is treated as the ordinary character itself, except when the character is (,), {, or }, or the digits 1 through 9.

Special Characters A regular expression special character preceded by a backslash matches the special character itself. When not preceded by a backslash, such characters have special meaning in the specification of REs. Regular expression special characters and the contexts in which they have special meaning are:

- . [\ The period, left square bracket, and backslash are special except when used in a bracket expression. A period (.), when used outside of a bracket expression, is an RE that matches any printable or nonprintable character except <newline>.
- * The asterisk is special except when used in a bracket expression, as the first character of a regular expression, or as the first character following the character pair \((see REs Matching Multiple Characters).
- ^ The circumflex is special when used as the first character of an entire RE (see Expression Anchoring) or as the first character of a bracket expression.
- $ The dollar sign is special when used as the last character of an entire RE (see Expression Anchoring).
- delimiter Any character used to bound (i.e., delimit) an entire RE is special for that RE.

Bracket Expressions

You can instruct NNM to match one element from a group by enclosing the group in square brackets ([ ]). You must place at least one item in the brackets.

The following rules apply to bracket expressions:

matching list A matching list expression specifies a list that matches any one of the characters represented in the list.

Example: `[abc]` matches any of `a`, `b`, or `c`.

non-matching list A non-matching list expression begins with a circumflex (^), and specifies a list that matches any character except the characters in the list.

Example: `[^abc]` matches any character except `a`, `b`, or `c`.

The circumflex has this special meaning only when it occurs first in the list, immediately following the left square bracket.

range expression The starting range point and the ending range point must be letters or numbers.

Both starting and ending range points must be valid elements, and the ending range point must collate equal to or higher than the starting range point; otherwise the expression is invalid.

Matching Multiple Characters

The following rules may be used to construct REs matching multiple characters from REs matching a single character:

RERE — The concatenation of REs is an RE that matches the first encountered concatenation of the strings matched by each component of the RE.

Example: `bc` matches the second and third characters of `abcdefabcdef`.

<char>* — A single character followed by an asterisk (*) matches zero or more occurrences of the character. The first encountered string that permits a match is chosen, and the matched string will encompass the maximum number of characters permitted.

Example: in the string `abbbcdeabbbbbbcde`, both `b*c` and `bbb*c` match the substring `bbbc` in the second through fifth positions.

An asterisk as the first character of an RE loses this special meaning and is treated as itself.

Expression Anchoring

An RE can be limited to matching strings that begin or end a line (i.e., anchored) according to the following rules:

- A circumflex (^) as the first character anchors the expression to the beginning of a line; only strings starting at the first character of a line are matched by the RE.

  Example: `^ab` matches the string `ab` in the line `abcdef`, but not the same string in the line `cdefab`.

- A dollar sign ($) as the last character anchors the expression to the end of a line; only strings ending at the last character of a line are matched by the RE.

  Example: `ab$` matches the string `ab` in the line `cdefab`, but not the same string in the line `abcdef`.

- An RE anchored by both ^ and $ matches only strings that are lines.

  Example: `^abcdef$` matches only lines consisting of the string `abcdef`.

The use of duplication characters (+,*) following anchors is illegal.

Precedence

The order of precedence is as follows, from high to low:

- [ ] square brackets
- * + ? asterisk, plus sign, question mark
- ^ $ anchoring
- concatenation
- | alternation

For example, `abba|cde` is interpreted as “match either abba or cde.” It does not mean “match abb followed by a or c followed in turn by de” (because concatenation has a higher order of precedence than alternation).

Pattern Matching Examples

Pattern Matching Examples

•Given the following hosts in your environment:

icicle
ice
nice
iceland
pie
pile
piece
tilled
pipe
fili.edu

•Which hosts would pass a filter using the following regular expressions?

ice	ice$
^ice	^ice$
i[cl]e	i.e
i..e	i\.e
ic*e	il*e
i[cl]*e	i.*e
i\.*e	

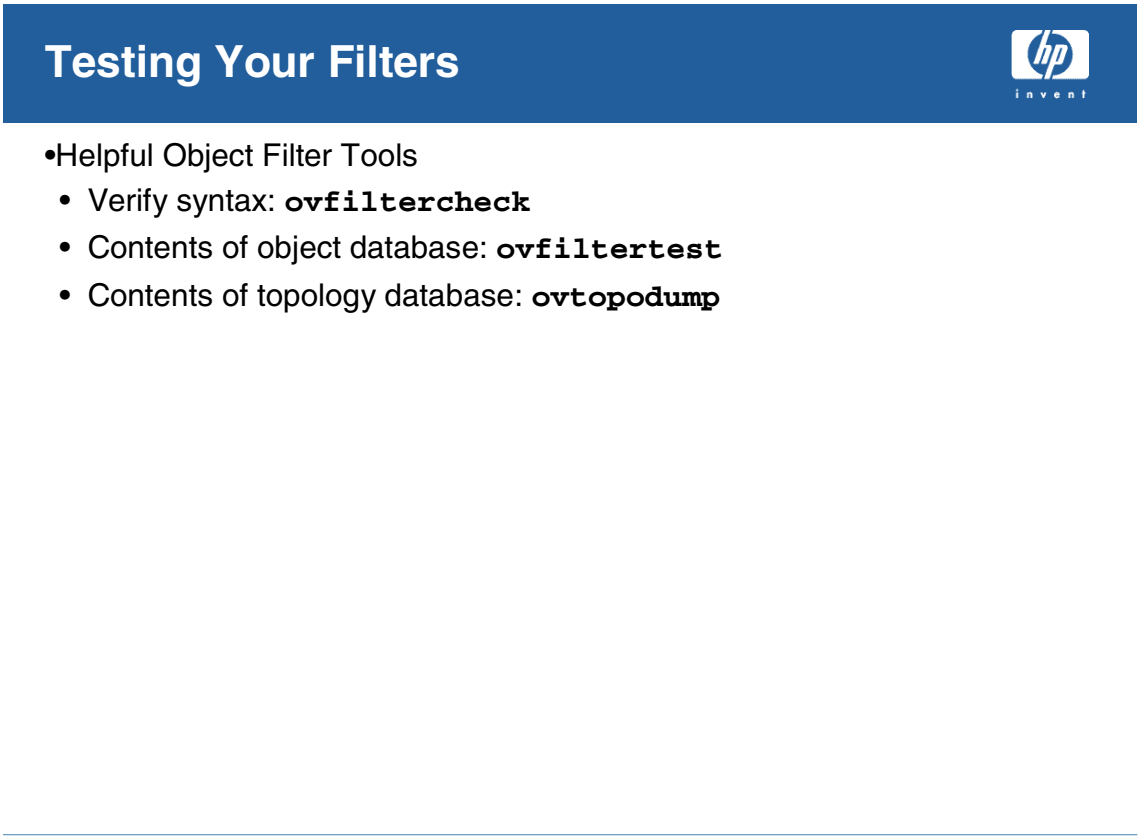
©2004 Hewlett-Packard Development Company, L.P.

Comparing Special Pattern Matching to String Regular Expressions

Compare the difference in which nodes pass the filter when you enclose the value string in quotes or not:

Node Address	“IP Address” ~ 15.*.1.9	“IP Address” ~ “15.*.1.9”
15.7.1.9	yes	yes
156.7.7.119	no	yes
7.156.1.92	no	yes
156.119.3.7	no	yes
15.179.23.75	no	yes
15.1.9.7	no	yes

Testing Your Filters



Three utilities exist for testing the `filters` file, `ovfiltercheck`, `ovfiltertest`, and `ovtopodump`.

These utilities are located in:

Windows: `%OV_BIN%`
UNIX: `$OV_BIN/`

ovfiltercheck

If the syntax of a filter file is incorrect, it cannot be applied.

This utility evaluates the **syntax** of the `filters` file using infix to parse files. If no errors are reported, a table is output along with a list of all filters, sets, and expressions defined. Key options are:

- The `-t` option. Specifies a filters file other than the default (for testing purposes only). If this is not specified, it assumes the default `filters` file.
- The `-v` option. Specifies verbose output.

ovfiltertest

This utility tests filters against the set of objects contained in the topology database. Key options are:

- The -f option. Specifies a specific filter in the filters file.
- The -a option. Allows comparison to the ovwdb object database.
- The -v option. Specifies explicit information on objects that did not pass.
- The -C option. Specifies testing only against objects from the named collection station.

When using ovfiltertest, you must be aware of the type of filter that is being tested (discovery, topology, or map) and evaluate the output of the filter application with the specific type of filter in mind.

ovfiltertest can evaluate any attribute found in ovwdb.

NOTE Keep in mind that the ovfiltertest -a command does not follow the same pattern for node/interface evaluation as the actual product. Instead of treating a node and its interfaces as a "super-object,", the ovfiltertest command treats nodes and interfaces as separate objects. See the *ovfiltertest* reference page in NNM online help (or the UNIX manpage) for more information.

ovtopodump

This utility complements the ovfiltertest command. When used with the -f option (filter), displays the contents of the topology database as evaluated by a specific filter.

Example filters File

Example Filters File

```
Sets {
      CriticalNodes "Critical Hosts" {ovdev,
/openview/criticalNodes.txt}
}
Filters {
      Routers "All Routers in the Enterprise" {isRouter == TRUE}
      Critical "Only critical hosts" {"IP Hostname" in CriticalNodes}
}
FilterExpressions {
      Backbone "" {Routers || Critical}
}
```

U5089S ©2004 Hewlett-Packard Development Company, L.P. 14

Sets:

The Sets definition is labeled `CriticalNodes` and comprises the set of objects: `ovdev` (hostname) and the hosts referenced in the `criticalNodes.txt` file located in the `/openview` on UNIX or `\openview` directory on Windows. For this example, assume that this flat file contains the hosts: `mickey`, `goofey`, `minney`, and `donald`. The `criticalNodes.txt` file would have one host per line in the following format:

```
mickey
minney
goofey
donald
```

Depending on how your environment is resolving names, it may be necessary to specify the fully-qualified hostname, for example, `mickey.cnd.hp.com` instead of just the hostname `mickey`.

Thus, the set of objects as defined by `CriticalNodes` is comprised of the five hosts: `ovdev`, `mickey`, `minney`, `goofey`, and `donald`.

Filters:

This section contains two filter definitions: `Routers` and `Critical`. `Routers` looks for all objects which have the `isRouter` attribute set to `TRUE`. `Critical` defines all objects whose `IP hostname` is part of the defined set.

Only one of these filter definitions can be specified in any given filter application.

FilterExpressions:

The `FilterExpression` definition positively identifies objects which are both routers *or* included in the critical filter definition by combining the definitions of the filters as specified in the code module shown in this slide.

This is a good way to incorporate several filter definitions together.

All filters referenced in a given `FilterExpression` definition must have been previously defined in the same filter file.

Each place in NNM where you may apply a filter may impose additional requirements on the contents of the filter.

Lab Exercises

•In this lab you will:

- Describe the functions of basic NNM filters.
- List and describe the basic block of the filters file.
- Describe Attribute Value Assertions.
- Create and apply a filter for Node View.
- Create and apply a DHCP filter.

©2004 Hewlett-Packard Development Company, L.P.

Objective: The purpose of this lab is to review the fundamental concepts behind NNM filters.

Review Questions

1. What object attributes are never available for filtering in NNM?

2. List and briefly describe the filters available in NNM.

3. Define an AVA, list the types of AVAs and give a brief description.

4. What is the proper way to use "wild cards" with the AVAs, and on what are these "wild cards" based?

5. What tools are available for testing filters, and what does each one test?

6. Using the partial listing of hosts and IP Addresses shown below, define a filter definition for each of the following:

   Example host information:

   ```
   jim.hp.com.uk  192.6.249.21
   jim.hpicome    192.6.249.22
   jim.hpicom     192.6.249.23
   jim.hpcom      192.6.249.24
   jim.hp.come    192.6.249.25
   ```

   a. Pass only hosts in the hp.com domain.

   b. Pass only hosts that have an odd IP Address in the last octet.

Lab Exercises

Preparation

1. Review the default `filters` file, then copy the `filters` file to `filters.orig`. You may want to leverage portions of the file during these exercises. Note: Do NOT remove or clear the original `filters` file. It contains filters that are in use by NNM.
2. To prepare for class labs, you will create some special purpose filters in this lab. In later labs you will build additional filters, so not all combinattions are covered here.

   Part of successful filter operations depend upon understanding the environment to be managed. Examples throughout this workbook may use example configurations that do not match your actual training environment. With the help of your instructor, be sure you can

identify the following items:

DNS Domain (if applicable): _____

IP Address Range for classroom systems: _____

IP Hostnames for the classroom: _____

Additional classroom devices: _____

3. Create and test a filter, without using a set definition, that passes only the systems from your classroom. Verify its operation using `ovfiltertest` and view the results in a Node View.

4. Modify the above filter to use a Sets definition.

5. (Optional Exercises) Write and verify a filter to do each of the following:

   a. Show just topology equipment.

   b. Show only PCs (requires sysObject ID).

   c. Show a particular subnet, where the network mask is not at a byte boundary.

   d. Show HP Series 800 systems usint the `sysDescr` attribute.

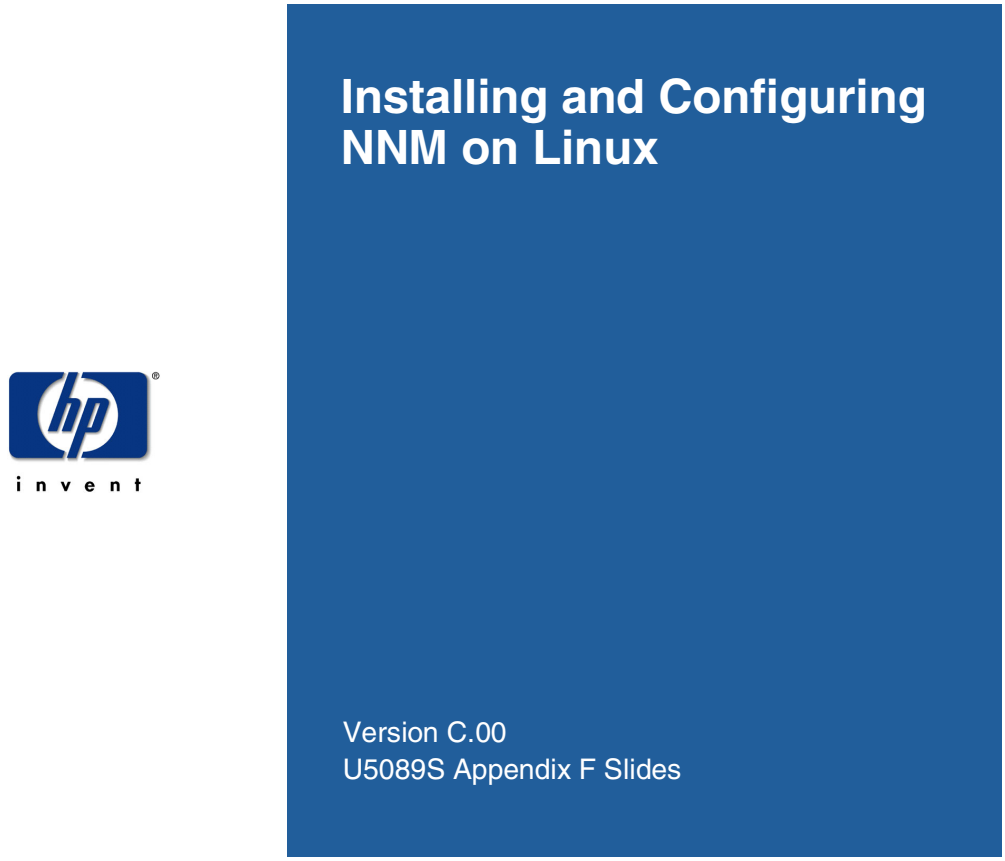
6. (Advanced Optional Exercises) Write and verify a filter to perform the following:

   a. Show only critical systems.

b. Show only systems whose IP address in the last octet is an EVEN number (trickier, since some objects not of interest have an empty IP address attribute, i.e. is 0, and you don't want those.)

F Installing and Configuring NNM on Linux

Module Objectives



At the end of this module the student will be able to:

- Locate the hardware and software requirements for NNM.
- Install and start NNM.
- Configure Oracle for use with the Data Warehouse.
- Verify that NNM is running.
- Remove NNM.

Installation Overview

Installation Overview

- Initial Installation
  - Hardware Requirements
  - Software Requirements
  - Install NNM
  - NNM “Universal” Pathnames
- After Installation
  - Run with Instant-On License Password (60 day expiration).
  - Register to obtain a Permanent License Password.
  - Install Permanent License Password.

©2004 Hewlett-Packard Development Company, L.P.

This guide will help you install HP OpenView Network Node Manager on a Red Hat Linux Advanced Server 2.1 system.

There are several tasks to complete during the installation process:

- Match your system’s setup to the NNM minimum requirements.
- Complete any necessary pre-installation steps.
- Install the NNM software.
- Configure SNMP agents.
- Run the NNM software.
- Obtain a permanent password for your NNM software.
- Configure NNM to access the HP OpenView Web interface.

System Requirements

System Requirements

•Please refer to the Release Notes, Supported Configurations for:

- Computer
- Graphics Display
- RAM
- CD-ROM Drive (required installation device)
- Disk Space
- Operating System, including patches
- Networking Subsystem
- Windowing Subsystem
- SNMP Agent (NNM management station)
- RDBMS (Optional)
- Web Browser

 ©2004 Hewlett-Packard Development Company, L.P.

Please refer to the most recent Release Notes for the latest information on hardware platforms, operating system, and supporting software requirements. The release notes can be found at `http://ovweb.external.hp.com/lpe/doc_serv`, through your Online Help, or in *`install_dir`*`\www\htdocs\C\ReleaseNotes` on an installed system.

The amount of RAM in your NNM management station should be based on the number of nodes which you wish to manage. Additional RAM may also be required to run third-party OpenView applications on top of NNM. See the *Network Node Manager Performance and Configuration Guide* for assistance in calculating for the optimum amount of RAM.

NOTE The NNM management station requires the use of UDP socket ports 161 and 162. (The `ovtrapd` background process communicates via port 162, and the SNMP agent communicates via port 161.) On Sun systems, occasionally these ports have already been reserved by another application (e.g. SunNet Manager). Prior to installing NNM, make sure that these ports are not in use.

Java Plug-In

Dynamic Views require the Java Plug-in and Java Runtime Environment. See the Release Notes, Supported Configurations for the current requirements.

For further information visit the Runtime Plug-in for HP-UX, JavaTM Edition, home page at: `http://www.hp.com/go/java`.

Pre-Installation Steps

- Set DISPLAY variable.
- Set up DHCP if necessary.
- Ensure proper web browser is installed.
- Other products that NNM requires.
- Set up /var partition if it doesn't already exist.
- Move the native SNMP agent to a different port.
- Check specified RPMs.

 ©2004 Hewlett-Packard Development Company, L.P.

To ensure that all products installed on your system run compatibly and efficiently with NNM, there are several preparatory steps you need to do before you actually install NNM.

General Configuration

1. Set the DISPLAY environment variable to the appropriate value. Also, run the xhost command to allow connection from the system running NNM to the system displaying NNM.

DHCP Setup

Users of the Dynamic Host Configuration Protocol (DHCP) need to ensure that their NNM management station is assigned the same IP address each time it runs NNM.

You can specify a range of IP addresses that your network is configured to assign dynamically for mobile devices. NNM keeps the map clean and the Alarm Browser list free of unnecessary messages about devices within this address range as they are repeatedly attached and detached from your network. See *Managing Your Network with HP OpenView Network Node Manager* for more information.

Web Browser Installation

Several of the NNM features are web-based, and require a web browser to be installed on the same system where NNM is installed. Furthermore, the Java-based graphical interfaces require a Java plug-in (JPI) for the browser.

Refer to the Release Notes for supported web browsers and JPI installation information. You can access the Release Notes from the CD. The Release Notes are in the `README.html` file.

From the left pane of the Release Notes, double-click on `Supported Configurations`.

To install the browser, follow the instructions that came with it. Be sure to configure any web proxies according to the browser's instructions.

Pre-Installation Requirements

The following are the pre-installation requirements for installing NNM:

- Red Hat Linux Advanced Server 2.1.
- Creation of `/var` partition with adequate disk space to accommodate NNM databases. Disk space depends on usage.
- NNM starts the `Emanate snmp agent` at udp 161 port. You should either stop the system-supplied native `snmp agent` or restart it at a different port, for example 50161.

  For more information on how to start the `snmp agent` at a different port, see the `snmpd.1` man page. If the snmp agent is not restarted at a different port, some of the `native snmp agent` features may be missing.
- If you are using Oracle for data warehousing, you need to install Oracle on Red Hat Linux Advanced Server 2.1 after installing NNM. Refer to the Installation guide supplied by Oracle for information on how to install Oracle for Red Hat Linux Advanced Server 2.1. For information on the supported versions of Oracle, refer to the release notes.
- For smooth installation of NNM, make sure the following RPMs are installed on the system:

  `ncompress-4.2.4` and `db3x-3.2.9-3`

  To check whether these RPMs are installed on the system or not, run the following commands:

  — `rpm -qa | grep ncompress-4.2.4`

  The output of this command should show the following:

  `ncompress-4.2.4-XX where XX can be any number.`

  — `rpm -qa | grep db3x-3.2.9-3`

  The output of this command should show the following:

  `db3x-3.2.9-3`
- Do not upgrade, downgrade, or unistall any of the following system RPM packages:

  — `glibc2.2.4-26`

  — `Xfree86-libs-4.1.0-25`

  — `libstdc++2.96-108.1`

  — `lesstif-0.93.15.3`

— `openmotif-2.1.30-11`

- Log on to the system as root user to install NNM.
- Configure the X server with 24-bit true colors or more than 512 colors.
- Ensure that the host name resolution is configured properly.

Installing the NNM Software

Installing the NNM Software

- Login as **root**.
- Mount the NNM CD-ROM.
- Change to the mount-point directory.
- Type command: **./install**

©2004 Hewlett-Packard Development Company, L.P.

Verifying Name Resolution

NNM depends on your network name resolution scheme being in good working order (DNS, NIS, NIS+, NetBios, local hosts files). NNM cannot conduct successful discovery if it is receiving inaccurate information. NNM will exhibit performance problems and erratic behavior when your name resolution scheme is not working. If you have name resolution in place prior to NNM's initial discovery process, hostnames are used for map symbol labels rather than IP addresses. Hostnames generally make the NNM maps and alarm messages more meaningful to your team.

If you are using DNS, HP recommends that the management station be a caching or secondary name server. Ensure that the management station secondary name server is not serving any other clients.

To test the health of the name resolution implementation before installing NNM, use the following procedure.

1. Copy the `gethost.exe` tool from your NNM CD to your hard drive.

   mount the CD drive,
   `/cdrom/OVDEPOT/OVNNMgr/OVNNM-RUN/opt/OV/support/gethost`

2. Choose an IP address that is not in use on your network.
3. Navigate to the directory where you copied `gethost`. At the command prompt, type either:

   ```
   gethost -a -i bogusIPaddress
   ```

   ```
   gethost -i realHostName
   ```

4. If you receive an immediate answer, your name resolution scheme is in good shape. If your system hangs while trying to formulate an answer, clean up your network name resolution before installing NNM.

Installing NNM Software

A single CD contains the files required to install on Red Hat Linux Advanced Server 2.1. This CD contains the following files and directories:

- `install.nnm`
- `OVAGENTDEPOT_LINUX`
- `OVNNMDEPOT_LINUX`
- `OVECSDEPOT_LINUX`
- `remove.nnm`
- Release notes in `.pdf` and `.html` formats

IMPORTANT The release notes contain an addendum that describes the changes incorporated into online and user documentation.

NOTE NNM requires specific fonts. After you have installed NNM, you can see which fonts are required by viewing the appropriate app-defaults files. The app-defaults files are located in `/usr/lib/X11/app-defaults`.

To install NNM on Red Hat Linux Advanced Server 2.1, complete the following steps:

1. Insert the CD-ROM in the CD-ROM drive.
2. If the CD-ROM is not mounted, you will need to mount it on to the system by using the following command:

   ```
   mount /dev/cdrom /mnt/cdrom
   ```

3. Use the `cd` command to change to the `/mnt/cdrom` directory as follows:

   ```
   cd /mnt/cdrom
   ```

4. To install NNM, execute the install.nnm script as follows:

   ```
   ./install.nnm
   ```

5. Provide appropriate answers to the questions that the installation software requires while installing the software.

NNM on Red Hat Linux Advanced Server 2.1 is now ready for use.

After Installing

After Installing

- Add **$OV_BIN** to your **$PATH** environment variable.
- Add **$OV_MAN** to your **$MANPATH** environment variable.
- Until Permanent License is installed. NNM uses an Instant-On License.
- Instant-On License clock starts ticking at installation, and expires 60 days later.
- System boot scripts have been modified to start NNM at system boot time.

- Shared library name extension in Linux is '.so', as in Solaris.
- The installation is based on native 'rpm' and not Software Deport (SD).

©2004 Hewlett-Packard Development Company, L.P.

Invoking Universal Pathnames

To use the universal path names, you must configure them.

You can configure these variables into your environment by running the appropriate command below, or by adding the line to your login start-up file (for example, `.profile`, `.cshrc`):

```
. /opt/OV/bin/ov.envvars.sh # Korn shell users
source /opt/OV/bin/ov.envvars.csh # C shell users
```

Instant-on License

When NNM starts running, the clock begins ticking on the Instant-On License, such that this license will expire in exactly 60 days from this moment.

The proper system startup files have been added such that `ovstart` is executed automatically at system boot time.

Configuring Oracle

Configuring Oracle

•NNM for Linux supports only Oracle.

•NNM does not provide installation, only configuration.

ovdbsetup -o

•This is optional for NNM; users may choose to use Oracle's administration tools.

 ©2004 Hewlett-Packard Development Company, L.P.

`ovdbsetup -o`

- Creates a new OpenView instance
- Configures system table spaces
- Creates NNM table spaces
- Creates `ovdb` user
- Instantiates tables (NNM schemas)
- Validates with `ovdbcheck`
- Changes Data Warehouse defaults to use Oracle
- ODBC DataSource configuration:
  `/etc/opt/OV/share/conf/analysis/system_odbc.ini`

Authentication

NNM uses the Oracle User and Oracle Password authentication. NNM does not rely on UNIX Login and User Password configuration and authentication for database configuration.

Verifying NNM

- **$OV_BIN/ovstart -v**
- **$OV_BIN/ovstatus**
- **$OV_BIN/ovw**
- HP OpenView Launcher
- Dynamic Views

©2004 Hewlett-Packard Development Company, L.P.

$OV_BIN/ovstart -v

Start all the background processes.

$OV_BIN/ovstatus

The `ovstatus` command provides status information about the various OpenView background processes started by `ovspmd`.

Sample output:

```
object manager name:netmon
state:RUNNING
PID:12738
last message:Initialization complete.
exit status:-

object manager name:ovwdb
state:                RUNNING
PID:                  12733
exit status:          -
```

Included in each status entry are the following fields:

- `object manager name` - name of the process as registered in the first field of that daemon's LRF (local registration file).
- `state` - last known state of the process (e.g. `RUNNING`, `NEVER_RUN`).
- `PID` - process's operating system process ID.
- `last message` - last message `ovspmd` received from the process. (Only `OVs_WELL_BEHAVED` processes have this field, because only these processes have an open communication channel to `ovspmd`.)
- `exit status` - process exit status (`-` if still running).

$OV_BIN/ovw

Verify that the NNM graphical user interface starts.

Home Base

Verify that Home Base is working:

From your web browser, specify the following URL:`http://<server>:7510`.

HP OpenView Launcher

Verify that the HP OpenView Launcher is working:

- From the NNM menu bar, select `Tools:HP OpenView Launcher`.

  OR

- From your web browser, specify the following URL:
  `http://<server>[:3443]/OvCgi/ovlaunch.exe`

Troubleshooting and Known Issues

Troubleshooting and Known Issues

- The installation log is written to /tmp/nnm_install.log
- Nettl tracing is supported on Linux as well.

- NNM-Linux does not have following:
  - Problem Diagnostics
  - Extended Topology and related views
- NNM-Linux is not tested to run under any High availability solution.
- NNM-Linux is not integrated with Customer views and Multi Cast.
- NNM-Linux Integration with Cisco Works is not available.
- NNM-Linux is not tested to co-exist other OV product like OVO agents, Performance agents
- NNM-Linux is not tested against Oracle for Openview 9i because it is still available on Redhat AS2.1. Rather NNM is tested with the native Oracle9i

 ©2004 Hewlett-Packard Development Company, L.P.

- `netmon` doesn't start

  `Ucd-snmpd` is installed as part of the operating system installation. This may occupy udp-161 port, and clash with the Emanate Snmp agent shipped with NNM Linux. This results in `netmon` not starting up.

  Solution : Stop the native `snmpd` and see that the Emanate Master Agent and `mib2agt` are started. After that, the command `ovstart -c netmon` should start `netmon`.

- `SNMP` tools don't work properly

  `Ucd-snmpd` is installed as part of the operating system installation. This also installs `snmpget`, `snmpset`, `snmpwalk` in `/usr/bin`. So, if your search path is not set to pick up these tools from `/opt/OV/bin`, the SNMP tools supplied with NNM will not be executed.

  Solution: Set your `PATH` in such a way that `/opt/OV/bin` is picked up before `/usr/bin`.

  The same reason is applicable for not getting correct man pages. As the solution, make sure `MANPATH` is set to pick up `/opt/OV/man` before `/usr/man`.

- `ovw` colors are not pleasing

  NNM-Linux requires 24 bit based true colors or at least more than 512 colors. If the X configuration of the monitor is not properly set (that is, less than 24 bits or 256/128 colors) the `ovw` maps, icons etc. will appear in monochrome colors.

Solution: Use 'X configuration' option of setup to properly set the colors of the monitor. Here, setup refers to the text-mode setup utility of RedHat available under usr/sbin.

- Failing processes don'tgenerate 'core' files.

  Core files are very important for debugging and CPE.

  In Linux, core file generation depends on the '`ulimit`' of the system. '`ulimit`' for core files is set to 0 bytes by default and because of this, core files are not getting generated.

  Solution: run '`ulimit -c unlimited`' command in the shell and start the executables. This will enable core file generation.

- Multiple instances of `snmpd`, `ovas`, `ovalarmsrv`

  All the threaded executables in NNM-Linux appear to run multiple instances.

  Reason: The theading model of Linux is not Posix compatible as on RH AS2.1. As the result the 2.4 Linux Kernel creates one process for every thread with separate pid.

  Solution: Currently none, but this does not hinder the operation in any way.

- Number of rpms

  NNM-Linux is packaged as 'rpms' and each file set is translated in to one rpm.

  The installation would see more than 30 rpms being installed. This is not a concern for support.

  Reason: The patching uses `tar`, so, there won't be a need to go back to these list of rpms as part of CPE for patch information.

- `snmpColDump` not producing expected results

  `snmpColDump` relies on `awk` to format and present output to the user.

  In case commands do not work as described in the man page, use '`awk -compat`'.

  Use `awk -f'\t'` instead of `awk -F\t`

Removing NNM

- Login as **root**.
- Type the command: **$OV_BIN/ovstop**.
- **cd /**
- Type the command: **/opt/OV/bin/remove.nnm**

 ©2004 Hewlett-Packard Development Company, L.P.

When you remove NNM components at the product, subproduct, or fileset level, the fileset dependencies are not necessarily enforced. Because of this, it is recommended that you remove product bundles only.

Steps to Remove

1. Log in as `root`.
2. Type `$OV_BIN/ovstop`.
3. Change to the root (/) directory.
4. Type the command:

   `/opt/OV/bin/remove.nnm`

   You will be prompted with the following question:

```
Enter an option:
(S) Remove NNM Software only.
(D) Remove NNM Data and Software.
(F) Remove NNM Data and Software and Force the removal of
```

```
     all files and directories under /opt/OV, /etc/opt/OV
     and /var/opt/OV.
(H/?) Help - reprint the option descriptions.
(X) Exit.
Enter one of (S|D|F|H|X).
```

5. If you have used Oracle to store your data warehouse, you must remove the NNM data manually.

Lab Exercises: Installing NNM on Linux

Installing NNM Lab

1. Install the NNM product
2. Modify the .profile
3. Start an HP OpenView Session

©2004 Hewlett-Packard Development Company, L.P.

G Device Managment Details

Module Objectives

At the completion of this module the student will be able to:

- Define the use of boards on Cisco devices.
- Describe how Extended Topology discovers and monitors Cisco boards.
- Define the use of aggregated ports on Cisco devices.
- Describe how Extended Topology discovers and monitors aggregated ports.
- Describe support for stacked HP ProCurve switches.
- Describe how Extended Topology connects layer 3 devices at the edge of the network.
- List types of duplicate IP address situations and how NNM supports each one.

Status Determination for Switches

Default Status Determination for Switches

Port Connection	Admin Status	Polling Configuration	Status
Unconnected	Up	Disabled	Not Monitored
		Enabled	Operational status
	Down at Last Discovery	Disabled	Not Monitored
		Enabled	Disabled
	Down Since Last Discovery (up at discovery)	Disabled	Not Monitored
		Enabled	Disabled
Connected	Up	Disabled	Not Monitored
		Enabled	Operational status
	Down at Last Discovery	Disabled	Not Monitored
		Enabled	Disabled
	Down Since Last Discovery (up at discovery)	Disabled	Not Monitored
		Enabled	Disabled

Configuration Default Visualization Status Color

U5089S ©2004 Hewlett-Packard Development Company, L.P. 2

The heuristics used by APA to determine polling and status are:

1. Unconnected switch ports are never polled by default.
   - Status is No Status.
   - Visualization is Not Monitored.
   - If you configure APA to poll unconnected ports, then it proceeds to the next rule.
2. Ports that are administratively down at the time of discovery are not polled.
   - Status is No Status.
   - Visualization is Not Monitored.
   - If you configure APA to poll administratively down ports,
     - They appear Disabled.
     - They are polled slowly (every 6 hours by default).
     - If one changes to administratively up and a trap comes from the device directly or through syslog, its status changes immediately.
     - If one changes to administratively up and there is no trap, it is recognized at the next polling cycle.

— A port which changes to administratively up status goes to the operational status, and it keeps the slow polling cycle. At the next discovery, it will be discovered in an administratively up state and that configuration takes effect.

— The polling interval is also recalculated if you restart `ovet_poll`.

3. Ports that are administratively up at the time of discovery are polled.

- Status reflects Operational Status as long as administrative status is up.
- If administrative status changes to down after discovery,

— The device is still polled at the short interval until APA is restarted.

— Its status is Disabled.

— An alarm is sent to the Alarm Browser indicating that the port has become Disabled.

Additionally, objects which were monitored, but which are no longer monitored, should have any outstanding events cleared.

Enable or Disable SNMP Polling for Unconnected Switch Ports

Control Polling of Unconnected Ports

•Configure ifAdminStatus manually on the managed device

•By default:
- Switches: do not poll unconnected interfaces
- Routers: do not poll unconnected interfaces that are adminDown
- Routers: DO poll unconnected interfaces that are adminUp

•Edit paConfig.xml

•Find UnconnectedAdminUpSwitchIF

•Change to false or true

•Note: Switch interfaces default to adminUp when unconnected, so you must commit to manually controlling the status on the switch.

•Admin status is only updated during discovery. Status polling only updates operStatus.

U5089S ©2004 Hewlett-Packard Development Company, L.P. 3

There are times when some devices will not be in the extended topology database, or will not otherwise connect correctly. An example of this is if Extended Topology discovers information from an OAD environment, but cannot talk to some of the end nodes using SNMP. The result is confusion about whether there are any nodes connected to certain ports on a switch.

APA provides a solution for this problem. APA decides whether to poll a device using attributes from the node and interface. For example, APA knows if an interface's port is connected to another node in the extended topology and knows the class of the device it is polling. You can configure APA to SNMP poll switch ports that are *either* known to be connected to another node in the extended topology or have an `ifAdminStatus` of `up`.

This solution involves editing the `paConfig.xml` file. This solution assumes that you manually configure the `ifAdminStatus` parameter on the switches you want to poll using SNMP.

To implement this solution, use the following procedure:

1. Manually configure the `ifAdminStatus` parameter on your switches. For example, if you want APA monitor a switch port using SNMP, you must manually set its `ifAdminStatus` to `up`.
2. Make sure you have APA enabled.
3. As `Administrator` or `root`, edit `$OV_CONF/nnmet/paConfig.xml`.

4. Search for `UnconnectedAdminUpSwitchIF`

   You should see the following:

```
 <classSpecification>
<filterName>UnconnectedAdminUpSwitchIF</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>
```

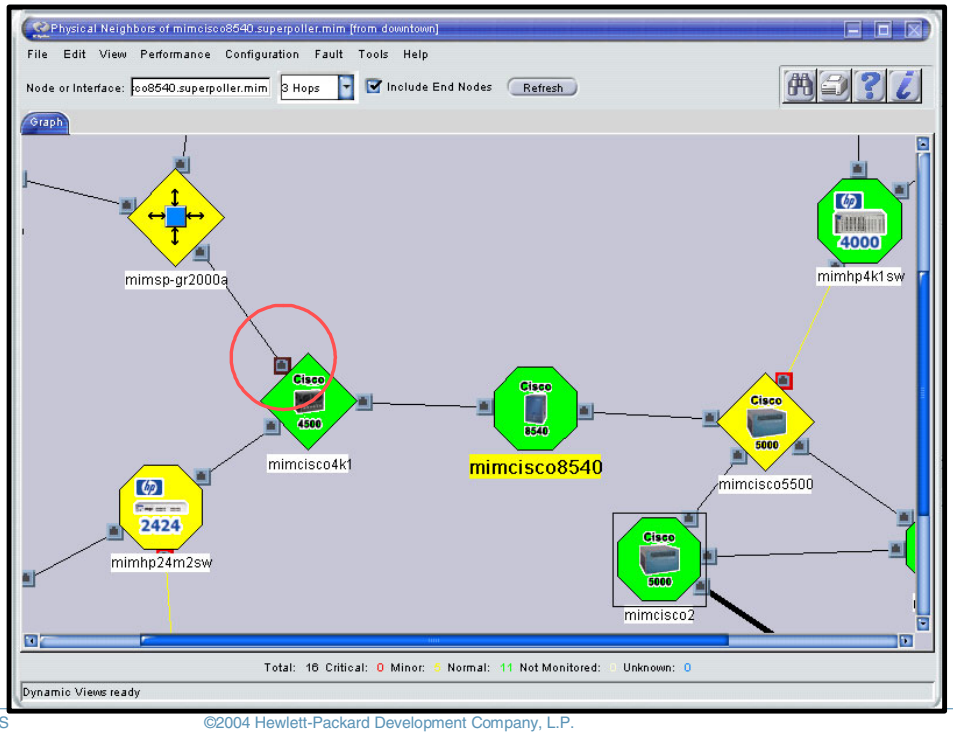
5. Modify the bold `false` to `true`.
6. As `Administrator` or `root`, run the following commands:

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

     This applies your changes to the APA polling process.

Example Admin Down Interface

Example Admin Down Interface



©2004 Hewlett-Packard Development Company, L.P.

Dynamic Handling of Unconnected Ports

An internet service provider may want to monitor the network infrastructure of customer networks. They typically have the customer switches and routers directly in their topology, but do not have access to the end nodes in the customers' environments. ISPs desire the capability of monitoring ports that are connected in the environment without the ports being connected in the Extended Topology database. Further, when a port transitions connected state in the environment, they want status reflected "correctly" on the map and alarms to be generated correctly.

For the NNM AE 7.01 release, these requirements were partially met:

- Interface Admin status was set as part of the discovery process. By default, unconnected ports were not polled. Polling of administrative up but still unconnected ports was enabled via configuration.
- Anything discovered as administratively down remained unpolled.
- If an interface's administrative status changed, you had to rediscover the network (or zone) to initiate a change in polling.

In NNM 7.5, you can configure APA to start polling connected ports immediately without waiting for a rediscovery. When an interface is marked administratively up, its status is based on its operational status and/or ping status. If you configure APA to monitor (slowly) interfaces marked administratively down, then its status is an administratively down status (disabled).

APA polls administratively down interfaces slowly and relies on event triggered polling to maintain the status until the next rediscovery. This allows for dynamic transitions, which disabling polling does not allow.

If an interface has polling enabled, APA initiates a status update based on link up/link down events immediately, independent of the polling cycle. Thus, an administratively down interface could still be polled and still see status updates if trap handling is enabled (if the device is configured to send traps or syslog messages to the management station).

If one end of a connection is administratively down, and the other end is operationally down, the connection is considered “down”.

Configuring Handling of Unconnected Ports

To enable monitoring of unconnected ports on switches, edit the `UnconnectedAdminUpOrTestSwitchIF` portion of `paConfig.xml` and set `snmpEnable` to true.

To enable monitoring of administratively down ports on switches, edit the `UnconnectedAdminDownSwitchIF` portion of `paConfig.xml` and set `snmpEnable` to true.

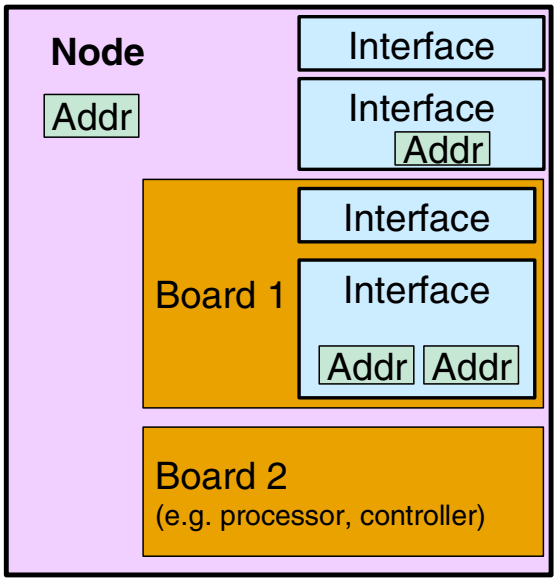
Stop and restart `ovet_poll` for your changes to take effect.

Managing Cisco Boards

Managing Cisco Boards

- Cisco-only feature
- Referred to as board, card, module
- Multiple proprietary MIBs

- Fewer alarms
- More specific fault indication



U5089S ©2004 Hewlett-Packard Development Company, L.P. 6

Communication devices such as routers or switches are composed of multiple components. The commonly seen components are Board, Chassis, Power Supply, Fan, Port etc.

Chassis contain slots where boards are plugged-in. There are many different types of boards: Processor board, Controller board, Memory board, I/O board etc. I/O boards contain ports which interface with the physical media to transmit and receive data packets. Boards can be specialized software modules such as Cisco's Route Switch Module, which provides layer 3 routing. Boards can contain sub-boards.

From network management model perspective, Board, Chassis, Power Supply, Port etc. are all physical components in the sense that the component can be physically identified in a switch or router.

Some of these components have logical counterparts. For instance, Port is the physical connection point to the wire whereas Interface is the logical correspondence; it models the communication characteristic of the Port.

In prior NNM releases, these two concepts, logical and physical, are often mixed in the same topology object. NNM 7.5 begins to distinguish these concepts into separate objects in the Extended Topology database.

Frequently, the terms card, module, and board are used interchangeably.

Discovering Cisco Boards

Board discovery is a Cisco-only feature. NNM models boards as objects.

Each port has a board ID which indicates its container. The board IDs were “derived” in NNM 7.0; now they are obtained via SNMP directly.

In NNM 7.5, the Cisco switch agent has been changed to support both switches and routers.

A board may have multiple sub-boards and a sub-board can have multiple ports. Cisco routers support this type of containment by CISCO-RHINO-MIB (marginally), but switches don’t have similar support in their MIBs. Therefore, sub-boards appear as boards. If a Cisco device has a board:sub-boards:ports relationship, the sub-boards are treated as top-level boards with their ports contained on them.

Cisco Routing-Switch Modules (RSMs) which have their own IP address and SNMP agent are modeled as a separate node

Monitoring Cisco Boards

- Differentiate between interfaceDown and boardDown
- Configure polling frequency in **`paConfig.xml`**
- Board events and their symptoms are correlated

 ©2004 Hewlett-Packard Development Company, L.P.

APA `ovet_poll` can monitor and analyze interface failures associated with a board and generate the appropriate status and alarms to reflect a fault. For example, if a board goes down which contains 10 interfaces and the node is reachable, then a single board alarm is emitted instead of 10 interface down alarms.

Because the board hierarchy is not modeled in NNM 7.5, when a board goes down, an APA_Board_Down alarm is generated for the board and all subBoards.

APA uses the appropriate Cisco MIB to obtain the device's board or subBoard status. The OverallStatus (what the GUI displays) may or may not directly correspond to what the device communicates because APA considers the context of the failure during analysis.

No board or subBoard degradation alarm will be emitted by APA as is done with the aggregated port support. However, the status of the board or subBoard may be set to Minor.

In this release, APA does not communicate the notion of ports to the user. APA will always drill down through the port to get to the interface of interest.

Board Status propagates to Node Status just like interface status propagates.

Configuring Board Polling

By default polling is enabled for boards.

You can disable polling of boards by editing `paConfig.xml` and searching for AllBoards. Uncomment the block to disable polling.

You cannot configure the frequency of board polling.

If you disable monitoring of boards, the board status will be No Status (Not Monitored in displays) and the interface handling appears as in NNM 7.01 where the interfaces appear directly contained in the node.

NOTE The number of boards being polled shows in the APA statistics tab of Home Base.

The APA statistics tab shows the last update time of the information. On a system with no load, this defaults to every 5 minutes. However, during busy polling times, the statistics may only be updated hourly.

Correlation of Board Events

When a board fails on a Cisco device, board and link failure traps or syslog messages may also be generated by the device. The customer would like to see the root cause APA event identifying the board that failed at the top level of the Alarm Browser. The Cisco board down traps and syslog messages, as well as related APA status events, are correlated under the root cause APA board failure event. (If both a trap and a syslog message are received, the second to arrive is correlated under the first to arrive.) NNM also monitors the rate of board failures and notifies you if a board if flapping.

APA sets board status for display in the GUI and issues board status alarms. The new APA alarms are:

OV_APA_boardDown

OV_APA_boardUnreachable

OV_APA_boardUp

OV_APA_boardRemoved

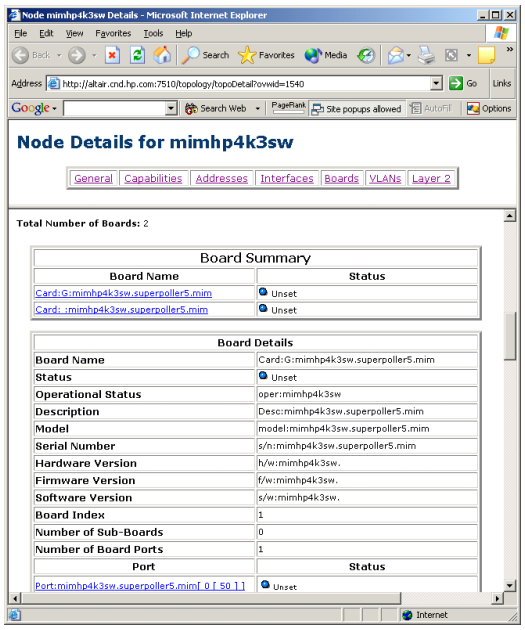
The PairWise ECS circuit processes board events such that boardUp cancels the other board events. (Events which are log only by default are not correlated to save processing time.) The deDup configuration has been updated too.

Note: The interaction of board analysis and aggregated port analysis can be quite complicated with the suite of devices and configurations in OpenView customer networks.

Board Visualization

Board Visualization

- Display on Node Details page of board data
- Board to sub-board relationship is not modeled.
- Show board attributes, status and associated ports.
- Show board count on Topology Summary page.



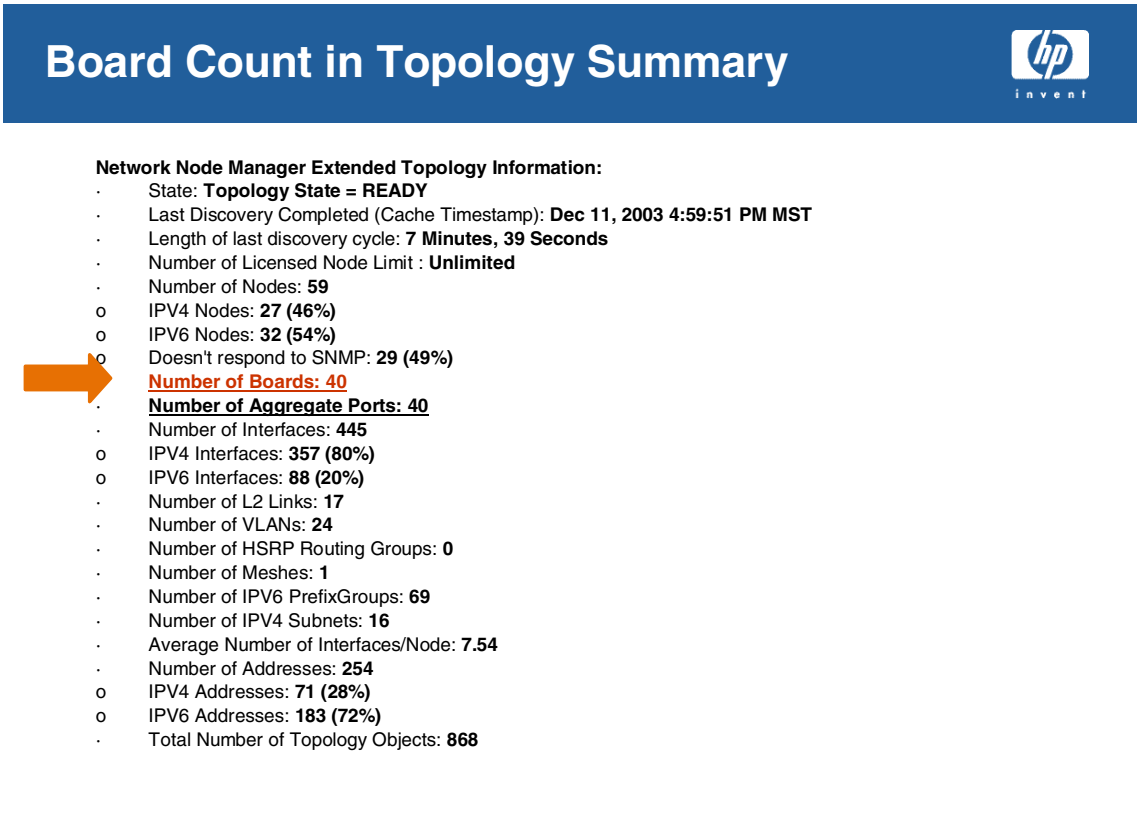
 ©2004 Hewlett-Packard Development Company, L.P.

Board visualization is presented on the Node Details web page and as a board count on the Topology Summary page.

If a board has a malfunction, APA emits an event which is displayed in the Alarm Browser. The source field of the trap contains the node that has the malfunctioning board. You can select the alarm and launch a Neighbor View focused on the malfunctioning node.

Alternatively, an open Neighbor View may alert an operator to a malfunctioning node containing a Cisco board. Opening the node takes you to Node Details where you can select Boards from the hyperlinks listed in the top frame and view the board status and details.

Board Count in Topology Summary



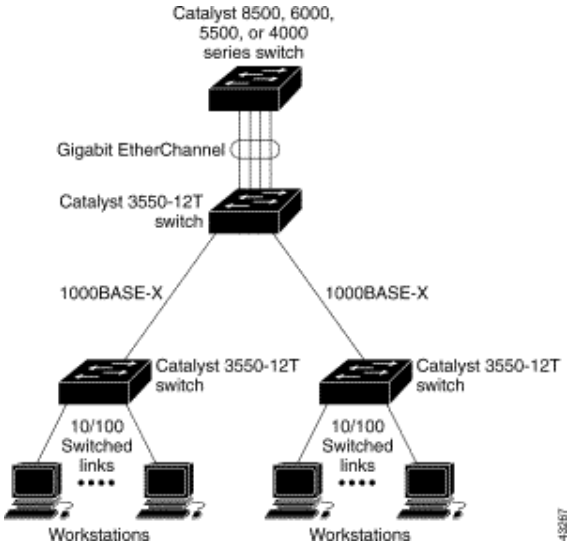
To determine how many boards are known to Extended Topology, launch a Topology Summary. The board count is listed in the displayed information.

Aggregated Ports

Aggregated Ports

- Cisco-only support
- Combining multiple physical ports into one logical link for increased bandwidth, load sharing, load balancing, and high availability
- NOT trunks, which are single links carrying multiple VLANs

- The following terms can be used interchangeably
  - Link Aggregation : IEEE 802.3ad / LACP
  - Port Aggregation : Cisco / PAgP
  - MultiLink Trunk : Nortel
- MultiLink Trunking VS. VLAN Trunking
  - MultiLink Trunking : One logical interface with multiple physical port members
  - VLAN Trunking : One single physical interface carries multiple VLANs.



 ©2004 Hewlett-Packard Development Company, L.P.

Cisco port aggregation is the combining of multiple ports into one logical port-channel or aggregate port (sometimes referred to as an AP).

Conversely, Cisco trunks are single links that carry traffic for multiple VLANs. So, in a Cisco terminology, multiple trunks acting as one logical trunk form a port aggregation. NNM is not supplying Cisco trunk management.

Port aggregation is a Cisco-only feature. PAGP is the only supported form.

Supported protocols:

- PAGP

  According to Cisco.com, “PAgP (Port Aggregation Protocol) [is] A protocol that aids in the automatic creation of Fast EtherChannel links. PAgP packets are sent between Fast EtherChannel-capable ports in order to negotiate the forming of a channel.”

  NNM supports this for switches and routers. You need the 2004 version of IOS to get the best PAGP MIB accuracy.

- LACP (not used)

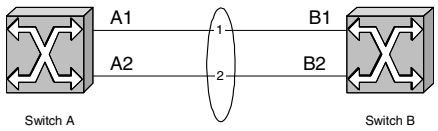
  According to Cisco.com, “Link Aggregation Control Protocol (LACP) is part of an IEEE specification (802.3ad) that allows you to bundle several physical ports together to form a single logical channel. LACP allows a switch to negotiate an automatic bundle by sending

LACP packets to the peer. It performs a similar function as Port Aggregation Protocol (PAgP) with Cisco EtherChannel."

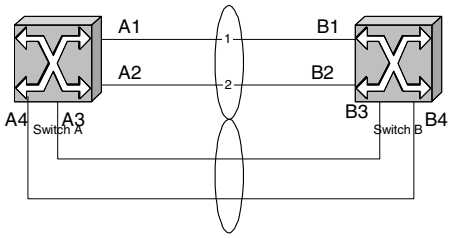
This MIB is currently supported by only one Cisco device model, and that support is incomplete.

Monitoring Aggregated Ports

Monitoring Aggregated Ports



One port aggregation
between two switches



Two port aggregations
between two switches

- Fewer Alarms
- More specific fault indication
- Higher priority fault indicator
- Show degradation

 ©2004 Hewlett-Packard Development Company, L.P.

Extended Topology discovers each interface (A1, A2, etc., B1, B2, etc.) and two logical aggregated interfaces: LA and LB. The neighbor information is:

[2]A1 – B1

[2]A2 – B2

[2]LA – LB

Note that the physical connectivity is not reported by the MIBs. NNM pairs the physical interfaces by their interface speed and suggests the connectivity for them. The MIBs return only the logical connection.

The logical interfaces LA and LB are a type of interface, and have ifIndex, ifName, ifDesc, ifOperStatus, ifAdminStatus etc. (MAC address is not present.).

APA reconciles the status values obtained from the physical interfaces with the values obtained from the logical interface to develop an OverallStatus value for the logical interface. APA sets overall status on the logical interface for display and issues new aggregated port status alarms.

The status of the virtual interface goes to Minor if the ifOperStatus of the virtual interface is non-UP or if any of the physical interfaces are non-UP.

The status of the virtual interface goes to Critical only when ALL physical (polled) interfaces go to

critical or unreachable.

	Local Port	Remote Port	Local Logical IF	Remote Logical IF	Alarms
One port goes down	Critical		Minor	Normal	aggregatedPortDegraded vIf1 – vIf2 // vIf2 is virtual IF on the other node ‡ If3 Down // Coorelation from APA & ConnectorDown
Port is fixed	Normal		Normal	Normal	aggregatedPortUp vif1 – vif2 ‡aggregatedPortDegraded vif1 – vif2 // Correlation from PairWise ‡ If3 Down // Correlation from APA & ConnectorDown ‡ If3 Up // Correlation from APA & ConnectorDown
Both ports down on one link	Critical	Critical	Minor	Minor	aggregatedPortDegraded vIf1 – vIf2 // vIdf2 is on the other node ‡ connDown If3-If4 // Coorelation from APA & ConnectorDown
Both ports fixed	Normal	Normal	Normal	Normal	aggregatedPortUp vif1 – vif2 ‡aggregatedPortDegraded vif1 – vif2 // Correlation from PairWise ‡ connDown If3-If4 // Coorelation from APA & ConnectorDown ‡ connUp If3-If4 // Coorelation from APA & ConnectorDown

Configuring Aggregated Port Polling

You can create APA polling configurations for aggregated interfaces using Extended Topology Topology filters based upon the following new attributes and corresponding out-of-the box filters:

```
isAggregatedIF// logical interface
```

```
isPartOfAggregatedIF// physical interface
```

For example, you may want to poll aggregated interfaces more or less frequently than non-aggregated interfaces.

Correlation of Aggregated Port Events

When an aggregated link fails or its performance is degraded because of underlying interface failures on a device, the customer would like to see an APA aggregated port status event identifying the aggregated port problem. However, there are a number of other events that can be generated with an aggregated link failure scenario. The customer only wants to see the APA aggregated port status event. The other events, such as LinkDown traps and aggregated link

related syslog messages, are correlated under the APA aggregated port status event. These sympathetic events are deleted from the Alarm Browser if they were being displayed.

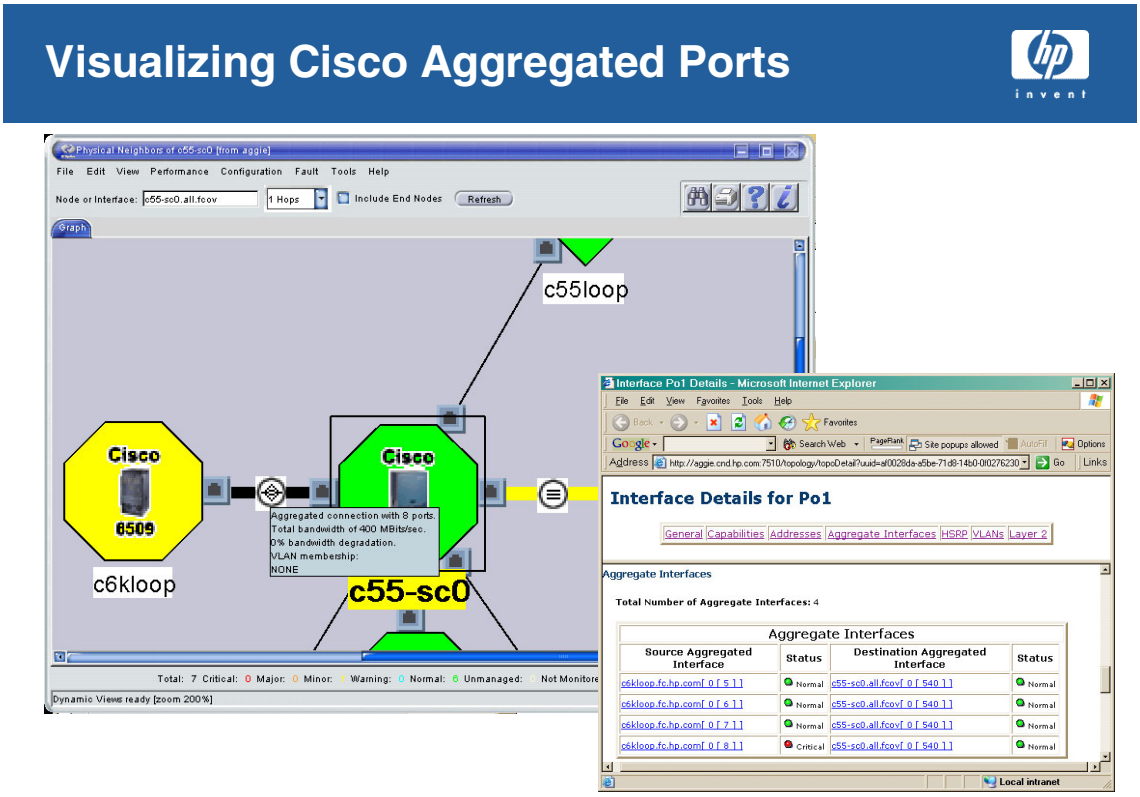
Logical aggregated port alarms and physical interface alarms are correlated using the ECS PairWise correlation, a MultiSource correlator, and deDup. They provide one top level alarm per fault.

Aggregations are used for increasing bandwidth or reliability. Therefore these connections tend to be very important to the customer. The customer needs to know about degradation of these aggregated links. Individual ifDown alarms are rolled up into the aggregated port alarm to minimize clutter in the Alarm Browser.

The new aggregation alarms are:

```
OV_APA_AggregatedPortUp  E1 E2

OV_APA_AggregatedPortDown   E1 E2

OV_APA_AggregatedPortDegraded E1 E2// Specifies how many physical Ifs are Up/Down on each node.
```

Visualizing Cisco Aggregated Ports



Dynamic views represent aggregated ports as thick L2 edges. They have status and the port tool-tip shows some details about the aggregation. Opening one port of the aggregated port displays the Interface Details view.

Hovering over the edge representing the aggregated port displays the count of ports, maximum bandwidth, percent degraded, and associated VLAN. When you hover over an aggregated port port, the tool-tip shows details about the aggregation, including interface information as well as the participating physical interface names, bandwidth maximum, and media type.

Aggregate links are identified by a circle icon with a single line branching to three lines and reforming on the opposite side as a single line. Double-clicking on an aggregate port (or any other non-redundant port) launches an interface details page in a new window.

Redundant L2 links (that is, separate connections between the same nodes) are displayed as thick lines in order to differentiate them from aggregated ports. Hovering over the edge shows port and edge count. Hovering over port shows the group of individual interfaces. Redundant edges are identified by a circle icon with three parallel lines in it. Double-clicking on a redundant port launches an interface detail page (in a new browser) for each member interface.

You may see an APA Aggregated Port Status alarm in the Alarm Browser. The source field of the alarm contains the node that has the malfunctioning aggregated port. You can select the alarm and launch a Neighbor View with the node highlighted.

From the dynamic view, you can double-click the aggregated port to display interface details. The

details include the list of participating interfaces. You can also open the node details and click the Aggregated Ports hyperlink at the top. This displays a list of aggregated port interfaces. Selecting one starts the interface details display. The Aggregated Ports table does not show port-to-port connections as these are unavailable from the device. The ports on each device participating in aggregation are sorted by interface speed.

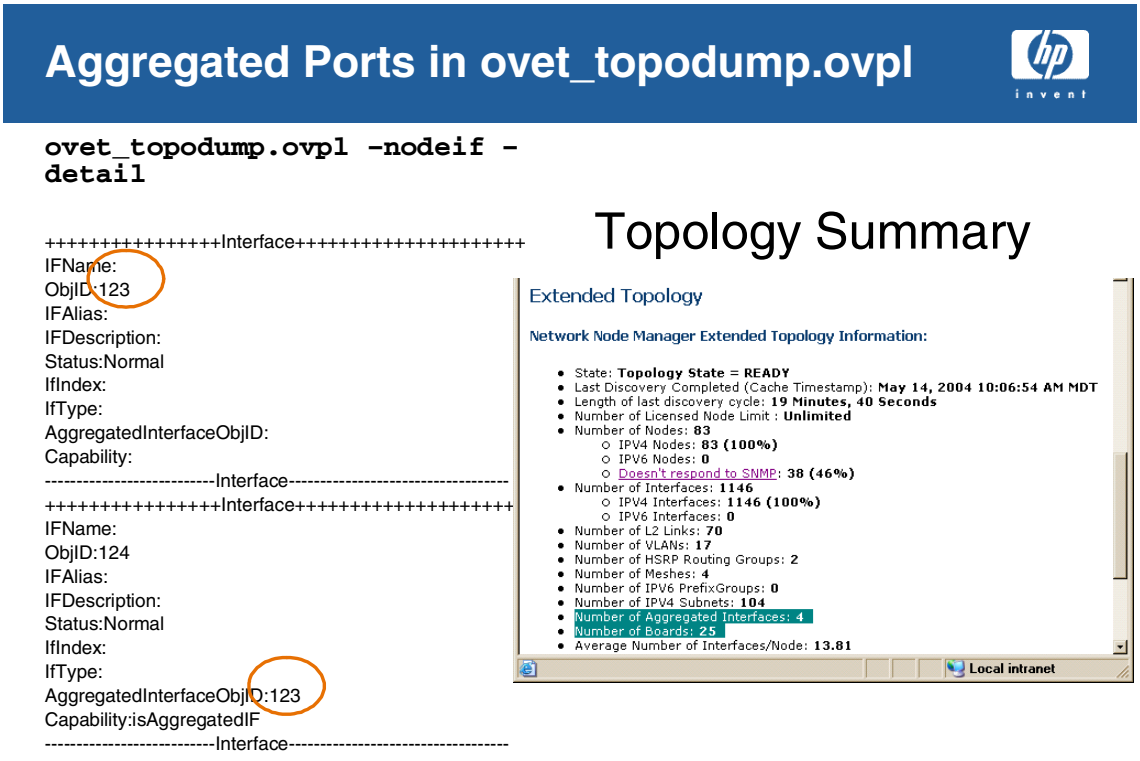
Alternatively, you may have a Neighbor View open when the edge representing the aggregated port turns yellow or red. Hovering over the edge shows its port count, maximum physical bandwidth and its current degradation percentage (e.g., 25% degraded). Where a mismatch in local-remote port speeds can be determined, a caution will also be printed in the tool-tip.

Changes to Previous Behavior:

You can now open all ports, which displays an Interface Details page. This is functionality works for both aggregated and non-aggregated ports.

Hovering over edges shows a tool-tip listing the maximum and degraded level of bandwidth. This information differs from the NNM 7.01 product where hovering popped up a tool-tip showing the VLAN and number of ports associated with the edge.

Aggregated Ports in ovet_topodump.ovpl



 ©2004 Hewlett-Packard Development Company, L.P.

In order to display the aggregate interface associations in `ovet_topodump.ovpl`, the Object ID of the interfaces are required to be shown. You can match the Object IDs to determine which ports participate in the same aggregation.

`ovet_topodump.ovpl -nodeif -detail` shows these Object IDs.

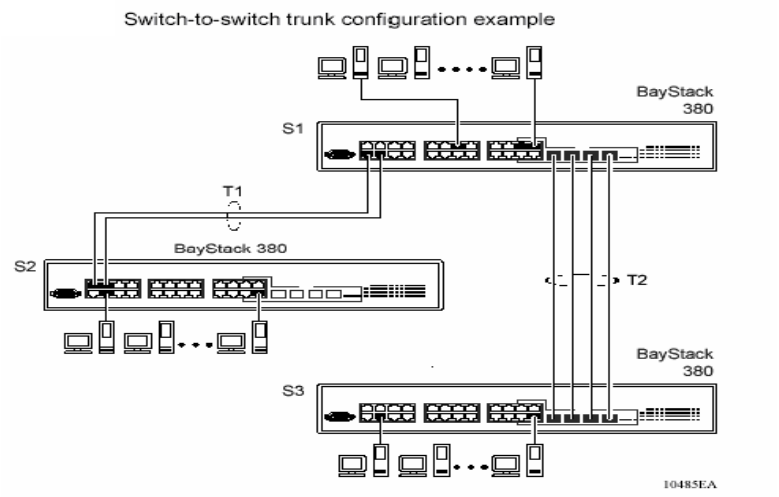
```
++++++++++++++++Node++++++++++++++++++++
NodeName:tshp51.cnd.hp.com
IPProtocolSupported:IPv4
ObjID:
OADId:0
SysOID:1.2.3.4.4.6
SysContact:
SysLocation:
Description:
Status:Normal
Capability:isLanSwicth isRouter
-------------------------ManagementAddress---------------------
AddressType:IPv4
ObjID
Address:
PingState:
---------------------------ManagementAddress-----------------------------------
++++++++++++++++Interface++++++++++++++++++++
IFName:
ObjID:123
```

```
IFAlias:
IFDescription:
Status:Normal
IfIndex:
IfType:
AggregatedInterfaceObjID:
Capability:
+++++++++++++++Address++++++++++++++++++++
AddressType:IPv4
ObjID
Address:
PingState:
---------------------------Address-----------------------------------
--------------------------Interface-----------------------------------
+++++++++++++++Interface++++++++++++++++++++
IFName:
ObjID:124
IFAlias:
IFDescription:
Status:Normal
IfIndex:
IfType:
AggregatedInterfaceObjID:123
Capability:isAggregatedIF
--------------------------Interface-----------------------------------
--------------------------Node----------------------------------
```

Nortel MultiLink Trunk Support

Nortel MultiLink Trunk (MLT)

Discover all layer 2 connections via MultiLink Trunk



U5089S ©2004 Hewlett-Packard Development Company, L.P. 15

Nortel switches use MultiLink Trunk (MLT), a point-to-point connection that aggregates multiple ports so that they logically act like a single port with the aggregated bandwidth. Grouping multiple ports into a logical link provides higher aggregate throughput on a switch-to-switch or switch-to-server application.

The MLT shows up in 802.1D forwarding table with a big port number, such as 4096, which cannot be found from MIB-2's ifTable. So all remote neighbors associated with MLTs will not have corresponding local neighbors.

For Nortel Passport switches, if there is remote neighbor found for a MLT, Extended Topology duplicates the same remote neighbor to each port member of this MLT. This makes a L2 connection between two MLTs discovered and the GUI shows it as a port aggregation link.

This feature is available for IPv4 only.

Switch Stack Device Features

- Varies with vendors: Cisco, Nortel, 3Com, and Procurve

- Only HP Procurve supported at this time
  - Stacks have a commander and one or more members
  - Procurve Stack members may or may not have IP address
  - Commander always has an IP address
  - Proprietory MIBs support for member access through the Commander
  - Members accessed from Commander using community strings of pre-defined formats.
    - <Commander's Comm.Str>@sw<MemberID>

U5089S ©2004 Hewlett-Packard Development Company, L.P. 16

Each vendor who implements stacked switches uses proprietary technology and terminology.

At this time NNM Extended Topology supports only HP Procurve switches in a stacked configuration.

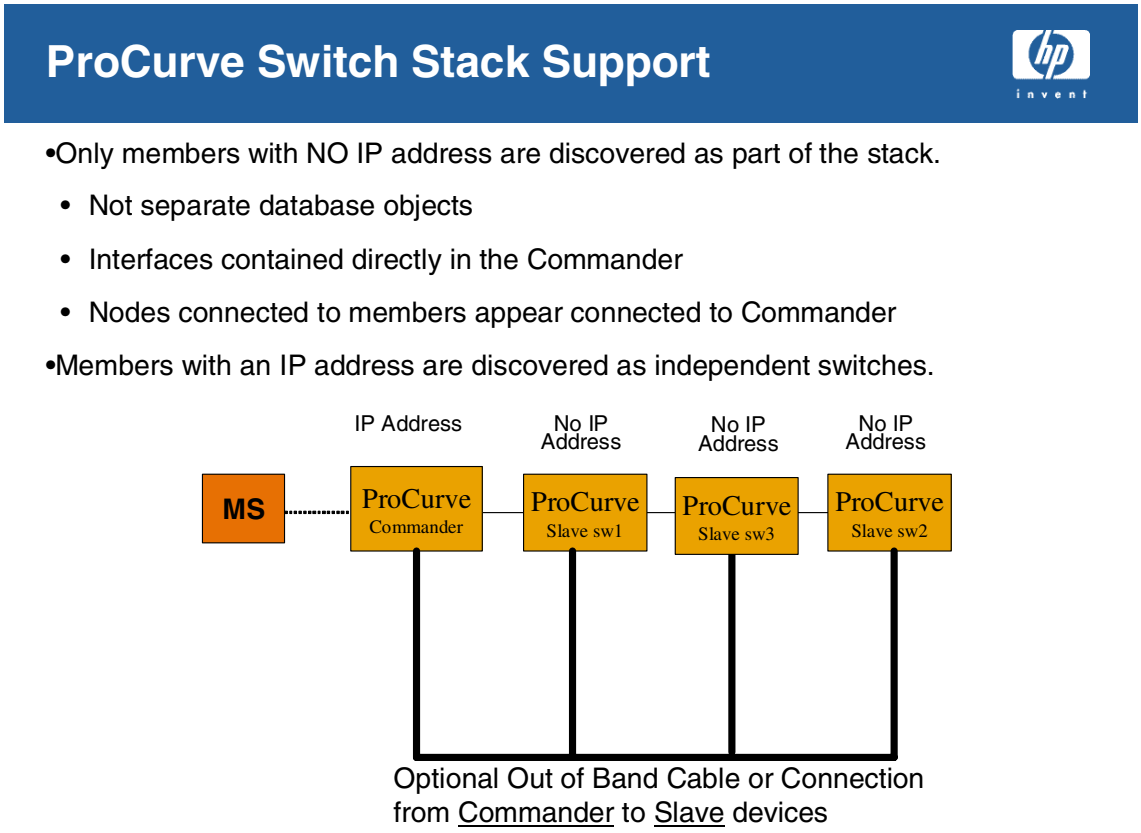
NNM Stack Support

NNM Stack Support

- Discover and represent the stack members as separate nodes in the Extended Topology database
- Monitor (status poll) the stack members as separate nodes

©2004 Hewlett-Packard Development Company, L.P.

ProCurve Switch Stack Support



The HP Procurve family of switches support virtual stacking of physically distributed switches in a single subnet, into a single logical stack. The virtual stack consists of a single commander switch and one or more participating member switches.

The commander switch is assigned an IP address and member switches may or may not have IP addresses assigned to them. The commander switch communicates to the member switch through layer-2 communication when they do not have IP addresses.

The following are the primary reasons for customers to use stacked switches:

- Reduces number of IP addresses needed for the network and enables adding switches to the network without having to do IP address management tasks
- Eliminates the need for any specialized cables for connections and removes the distance barrier between switches when using other stacking technologies
- Simplifies management of small work groups while scaling the network for bandwidth requirements.

The following are the series of switches that currently support stacking:

Series 2500 Switches, such as HP Procurve switch 2512, HP Procurve switch 2524

Series 4100 Switches such as HP Procurve switch 4108GL, 4104GL

Series 2400 Switches such as HP Procurve switch 2424, 2400

Series 8000 Switches such as HP Procurve switch 8000M

Series 4000 Switches such as HP Procurve switch 4000M

Series 1600 Switches such as HP Procurve switch 1600M

For a more exhaustive and up to date information please refer to the HP Procurve Website, http://www.hp.com/go/Procurve.

As the members of a stack may or may not have an IP address assigned to them, only those members that do not have an IP address assigned to them would be discovered as part of the Commander Switch of the stack.

All switches in the stack appear as separate devices, correctly connected to each other and to end nodes. The Commander's switch number is by default assigned to be “0” by the stack.

The Commander switch of the Stack is represented by the same switch icon that is used to represent other non-stack switches.

Those members of the stack that *have* an IP address assigned to them would not be discovered as part of the stack but would be discovered as any other normal ProCurve switches. Stack members with an IP address are discovered as independent nodes through `netmon` and passed to Extended Topology discovery as independent nodes. In order to avoid duplication of discovery, these members with IP addresses are not rediscovered transparently by the Extended Topology device agent as stack members, but are discovered explicitly as independent nodes.

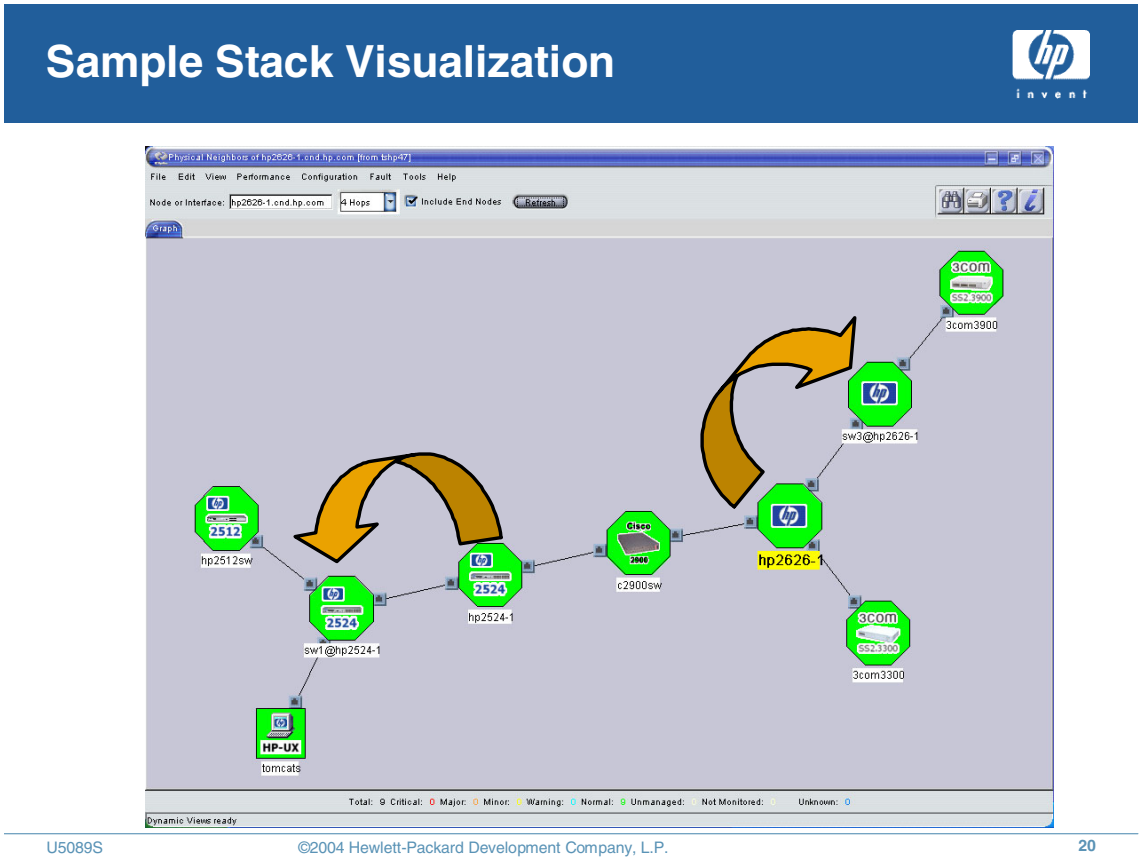
Stacked Switch Management

Stacked Switch Management

- Discovery creates an SNMP Proxy configuration entry for each slave device
  - identifies the commander’s node name as the proxy destination
  - the community string is a prefixed version of the commander’s community string
- Also updates SnmpNoLookupConf with the name of the slave device
- Since the SNMP configuration entries for the slave devices have the proxy set to the commander, the SNMP query actually goes to the commander.

©2004 Hewlett-Packard Development Company, L.P.

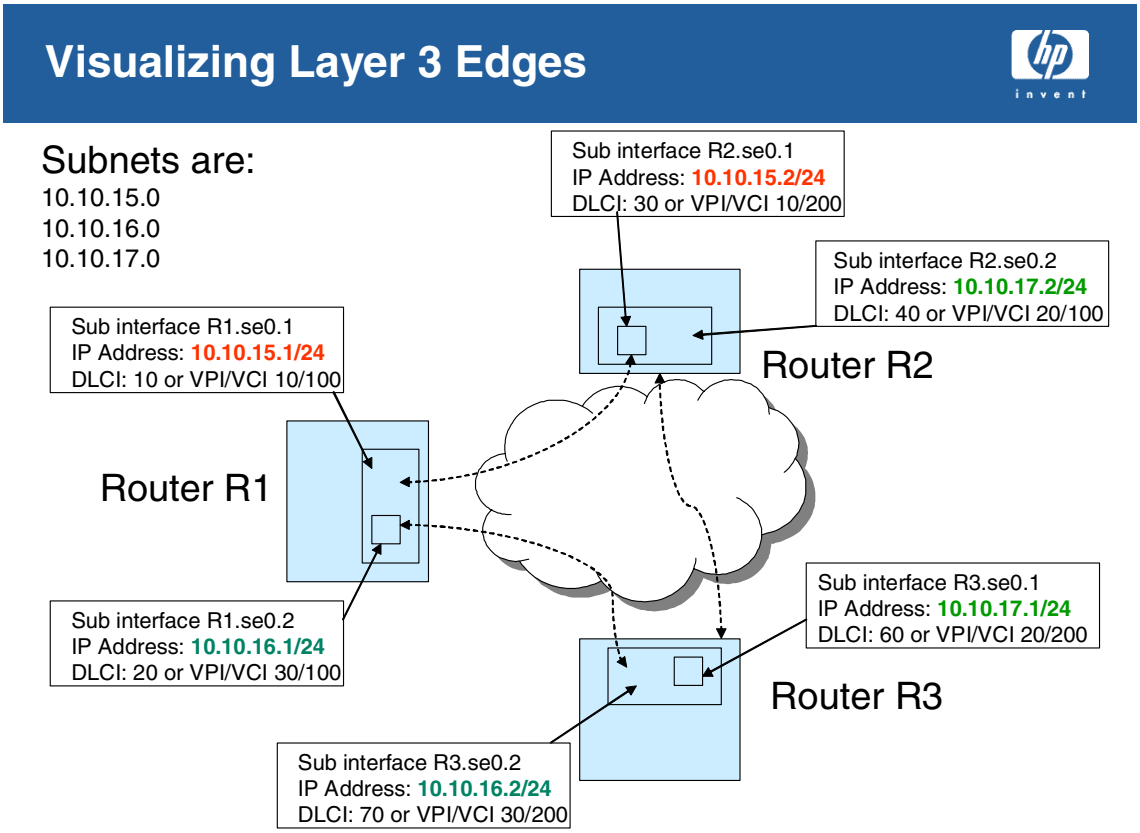
Sample Stack Visualization



This display shot shows two different stacks. Each stack has one commander that is directly connected to one member which does not have an IP address. End nodes and other switches are connected to stack members. Other non-stack and non-Procurve switches areconnected to commanders.

Double clicking any stack member node displays the Node Details Page for that node, similar to the support for Node Details page for any other, non-stack switches.

Visualizing Layer 3 Edges



At the LAN/WAN boundary, it is common to have routers with ATM or Frame Relay interfaces that connect to a service provider VPN. These interfaces are often configured to be in an IP subnet with a 31, 30 or 29 bit subnet mask, with only two nodes existing in the subnet.

NNM AE 7.01 focused on layer-2 discovery.

Customers would like to see NNM AE connect these two interfaces together physically. This would allow for correlation of related events for the interface pairs and also for accurate neighbor and path view creation.

It could be argued that the layer-3 connection between two routers should not be modeled as a physical connection. From an architectural purity standpoint, this is probably correct. The benefits of connectivity analysis for polling and correlation outweigh this aspect.

NNM 7.0 Handling of Layer 3 Connectivity

NNM 7.0 Handling of Layer 3 Connectivity

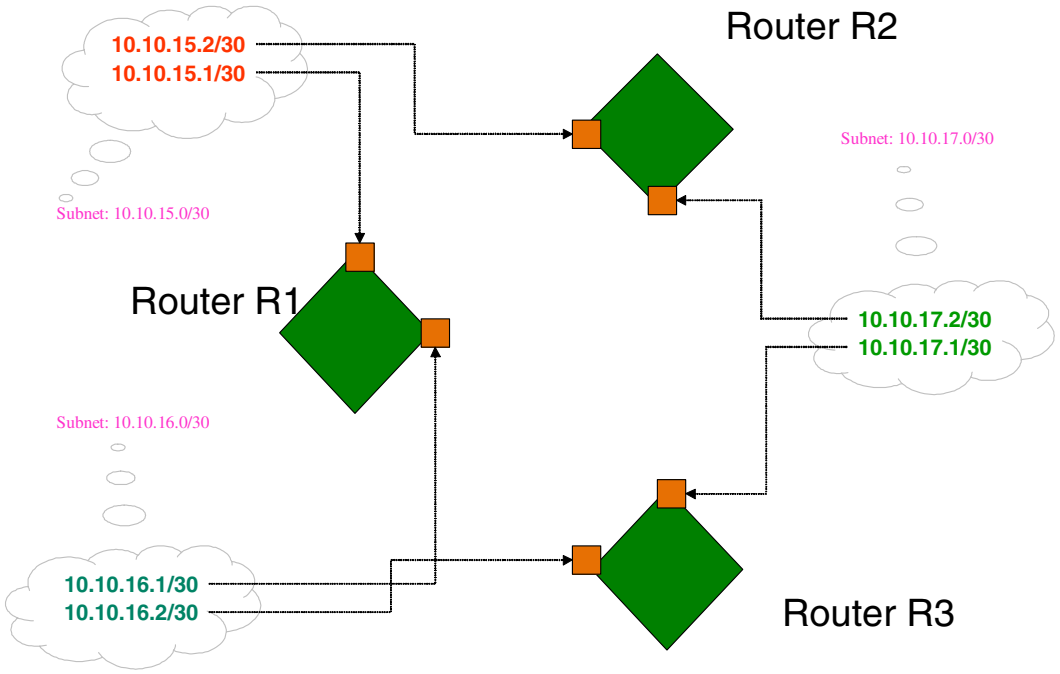
- Layer 2 centric (physical connectivity)
- Weak on Layer 3 connectivity.
- Typically occur in connections between routers (or other devices) with WAN interfaces
  - Routers with mainly ATM or FrameRelay interfaces, connecting to VPNs
  - T1/T3 Circuits and SONET etc.

- For Layer 3 devices (e.g. routers)
  - Vendor proprietary protocols used for connectivity analysis
  - E.g. CDP for Cisco, EDP for Extreme.
  - No Vendor protocol means No connectivity!
  - Result in edge interfaces left unconnected … wrongly!
  - Inhibits effective Root Cause Analysis

©2004 Hewlett-Packard Development Company, L.P.

Visualization Without Connectivity Information

Current Topology Visualization (7.01) when Connectivity Info NOT Available



©2004 Hewlett-Packard Development Company, L.P.

NNM 7.5 Addresses Layer 3 Edges

NNM 7.5 Addresses Layer 3 Edges

- Edge connectivity only
  - Layer 3 Core connectivity NOT addressed!
- Enhance connectivity accuracy
- Enable better Root Cause Analysis for polling and correlation.

- Also applicable to other generic WAN interfaces.
- 2-node Subnet - usually 30-bit subnet Mask
  - Utilize Subnet + Subnet Mask to derive Point-to-Point edge connectivity
  - Configurable Subnet Mask supported

- There are two scenario types:
  - Point to point
  - Point to multi-point
  - NNM 7.5 addresses ONLY Point to Point scenario!

 ©2004 Hewlett-Packard Development Company, L.P.

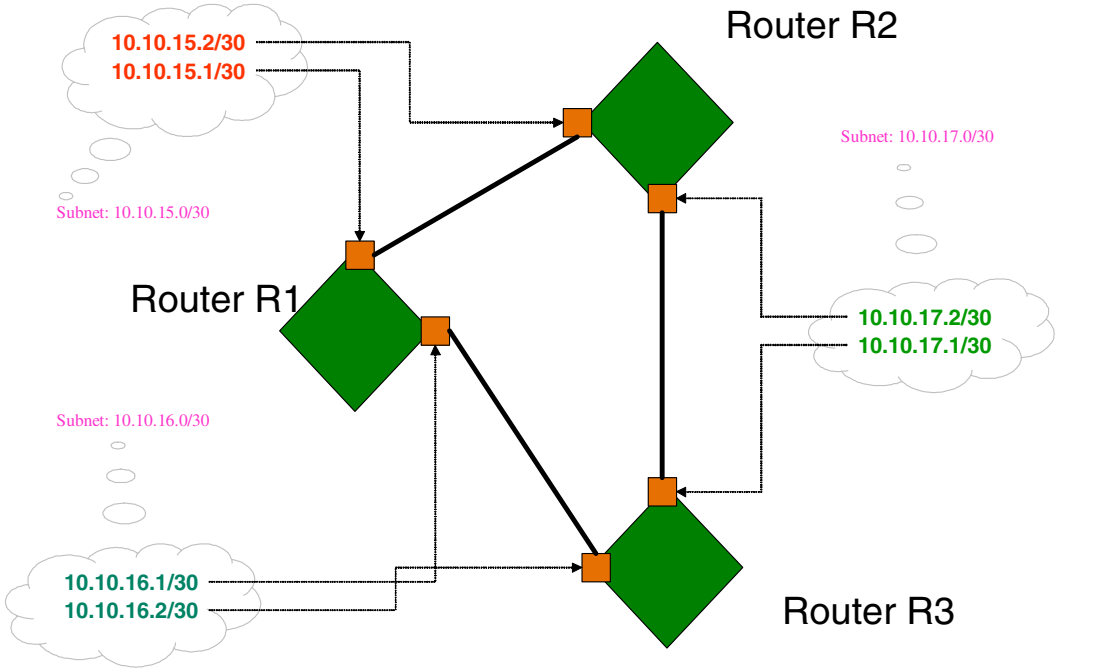
NNM analyzes the IP interfaces on routers. When exactly two router interfaces exist in the same subnet, as identified by IP address and subnet mask, and both interfaces are not already connected to some other device via the existing layer-2 connectivity analysis, the interfaces will be connected.

Connectivity information sources:

- Fdb Tables (Switches)
- CDP (Cisco + Procurve Devices)
- EDP (Extreme Devices)
- ILMI (ATM MIB supported devices)
- FDP (Foundry devices) – new!

Visualization When Connectivity Info is Available

Accurate Topology Visualization when Connectivity Info is Available



©2004 Hewlett-Packard Development Company, L.P.

Configuring Layer 3 Edge Discovery

Configuration Details

•Provide out-of-the-box value with the most commonly used configuration setting as default – yet configurable!

•Configuration through $OV_CONF/nnmet/EdgeL3Conn.cfg

- **`enableConnectivity=1`**. 1 = enable (default), 0 = disable
- **`minAllowedBitmask=29`**. [0-31] (Default = 29)

©2004 Hewlett-Packard Development Company, L.P.

Layer 3 edge visualization is enabled by default. To disable it, create the following file.

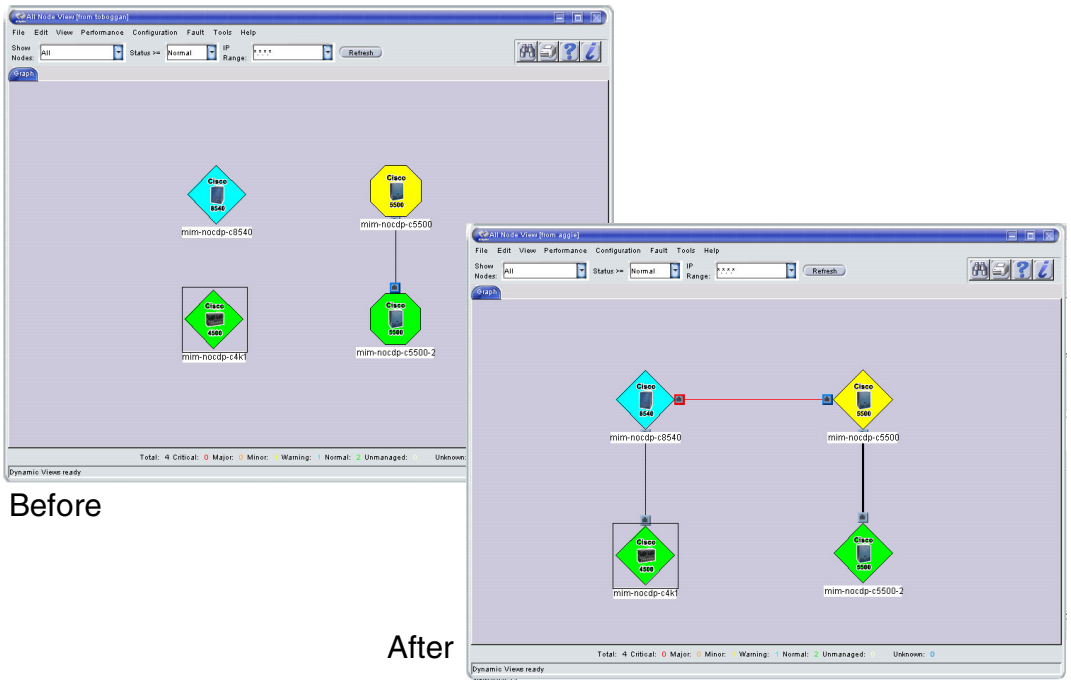
You can disable this capability or change the minimum allowable subnet bitmasks to identify participating routers. Edit `$OV_CONF/nnmet/EdgeL3Conn.cfg` and modify the `m_enableConnectivity` and `m_minAllowedBitmask` variables. The changes take effect as part of the next discovery (full or incremental) cycle.

- enableConnectivity. 1 = enable (default), 0 = disable
- minAllowedBitmask. [0-31] (Default = 29)

Note: Although configuration is carried through the configuration file, by default, the file does not exist.

Visualization Example

Visualization Example



©2004 Hewlett-Packard Development Company, L.P.

Duplicate IP Address Support

NNM AE Extended Topology manages Overlapping Address Domains where two separate environments use overlapping IP address ranges, but which need to be monitored from a single manager.

Allowing netmon to Encounter Duplicate IP Addresses

In current networks `netmon` can expect to encounter the same IP address in use by multiple interfaces and/or systems. When this occurs, `netmon`

- prevents duplicate IP addresses from making it to `ovtopmd`'s topology
- prevents duplicate IP addresses from being picked as management addresses.

Whenever `netmon` finds that a newly discovered or added IP address is a duplicate, `netmon`:

1. drops the new as well as existing (old) interfaces which have the same IP Address.
2. if this address was earlier set as the management address, picks a new management address for that node/interface.
3. stores all the dropped IP addresses in the `netmon.noDiscover` file to prevent the same Anycast IP address being re-discovered. (`netmon` creates the file if it does not already exist.)

No external configuration is required.

Anycast Address Support

AnyCast Address Support

- An address that exists on multiple nodes, any one of which can respond to communications.
- A node sends a request to an anycast address and the “closest” node responds.
- IPv6 introduced the concept
  - In IPv6, anycast has its own reserved address space.
  - NNM 7.5 does not support IPv6 anycast
- IPv4 does not have a reserved anycast address space. An anycast address:
  - exists on multiple routers which broadcast the anycast address into the routing protocols.
  - is usually a software loopback address (other than 127.0.0.1).
  - usually has a network mask of 255.255.255.255.
- Monitored via SNMP query on the interface (not ICMP)

 ©2004 Hewlett-Packard Development Company, L.P.

IPv6 introduced the concept of an “anycast” address; an address that exists on multiple nodes, any one of which can respond to communications with that IPv6 address. In IPv6, this concept is directly in the addressing architecture, with its own reserved address space. The theory is that a node can simply send a request to an anycast address and the “closest” node will respond.

In IPv4, this concept is used by convention, and is gaining popularity within the multicast community in particular as a method of locating rendezvous points. Since IPv4 does not have a reserved anycast address space, in practical terms you can not look at an IPv4 address and distinguish it from a regular unicast address. However, for an anycast address to work in the IPv4 space, it generally has to follow these conventions:

- The anycast address will exist on multiple nodes, usually routers since the node must be able to broadcast the anycast address into the routing protocols.
- The anycast address will be a software loopback address (other than 127.0.0.1).
- The anycast address will have a network mask of 255.255.255.255.

Thus, multiple routers will all advertise the same IP address into the routing protocols. The routing protocols take care of routing packets to the “closest” router serving the address based on the routing metrics.

In NNM 7.5, anycast addresses are only discovered and monitored in IPv4 environments. Since anycast address information is not available via SNMP MIB queries, discovery determines if

incoming addresses are qualified as anycast addresses using the following heuristic:

For any two or more interfaces with identical IPv4 addresses with the following characteristics:

- The interface is a software loop-back interface with an address other than 127.0.0.1.
- The matching address is in the same OAD
- The address has a subnet mask of 255.255.255.255
- Does not meet the characteristics for IP addresses as a backup/primary pair.

then categorize each address in the group as ANYCAST.

By default, anycast addresses are not monitored via ping, as it is generally impossible to tell if any given anycast node's address is responding to a particular ping. You can configure APA monitoring of AnycastAddresses in `paConfig.xml`. By default, ICMP polling is disabled and these addresses are only monitored via their interface via SNMP.

If an anycast address is monitored, then all IP address objects should generally reflect the same state. NNM concludes that all interfaces are in trouble if no node is responding to the anycast address. The interfaces containing that address are monitored by SNMP and become Minor if one of their addresses is down. If you disable SNMP polling as well, the interface becomes Critical because the failure of the address propagates directly to the interface with no way to verify it.

Anycast address should NOT generate any duplicate address warnings.

If an anycast address is enabled (by changing `paConfig.xml`) for ICMP polling, it will be treated like other IP addresses, with ping state set based on whether any response is received to the ICMP request, with no distinction of which node responded.

Backup Address Support

- Two interfaces have the same IP address.
- Must be on different devices
- One interface is administratively down (backup)
- IPv4 only

 ©2004 Hewlett-Packard Development Company, L.P.

It is common to have an interface configured as a “backup” to another interface, with only one interface enabled.

In this scenario, one of the interfaces will be in an administrative down state. An administratively down interface typically takes user action to bring to an active state, though sometimes this can be an automatic action when an attempt is made to communicate with the address.

You must configure `netmon` to ignore such addresses via the `netmon.noDiscover` configuration file.

Extended Topology can discover and manage duplicate IP addresses configured on different routers (in the same subnet), where one of the interfaces is administratively down. NNM discovers both the interfaces, and if possible creates a relationship of primary-backup between them.

NOTE NNM does not support for detecting duplicate IP addresses on the same node.

NOTE This feature covers IPv4 only.

Discovering Backup Addresses

Since backup/primary address information is not available via SNMP MIB queries, discovery determines whether incoming addresses are backup duplicate IPv4 addresses using the following heuristic:

For any two and only two interfaces with identical IPv4 addresses in the same subnet, in the same OAD,

- Both interfaces must be operational (accessible for SNMP) for the following administrative determination. Otherwise, a primary-backup relationship cannot be determined.
- If one of the interfaces is administratively up and the other interface is administratively down
  - — The address that is administratively up is qualified as the primary of a backup duplicate IP address pair
  - — The address that is administratively down is qualified as the backup of a backup duplicate IP address pair.

Backup interfaces do not generate extraneous warnings about duplicate IP addresses.

Monitoring Backup Addresses

Extended Topology allows duplicate IP addresses in the same OAD, and sets the “isDuplicated” attribute when detected. Additionally, “isBackup” indicates a backup interface. Filtering is allowed on either or both attributes.

When the interface becomes active, it will generate an Interface Up event. This is not correlated with the primary interface failure event in NNM 7.5.

The IP Address of the primary/backup interfaces is monitored based on the settings. Assuming the IP address is monitored, the ping state will be updated based on whether the IP address is responding. This may mean that an IP address on an administratively or operationally down interface may be marked as responding if the secondary or backup link is activated.

NOTE: In NNM 7.5, we are assuming that the two interfaces have the same IP address. There are other backup/failover scenarios where interfaces are activated to provide alternate routes without having the same IP address. This would be more typical with a dialup ISDN link as a backup to a higher bandwidth WAN link.

For backup interfaces that do not have a duplicate IP address, it is recommended that the configuration be adjusted such that pingEnable is set to false for those interfaces and/or IP addresses, so that pinging the address does not inadvertently activate the address.

Other Duplicate IP Address Support

Duplicate External Addresses

Duplicate external addresses exist when a service provider has multiple external connections to customers that have overlapping addresses. Technically, this is the realm of the Overlapping Address Domain support. However, the distinction here is that the routers primarily exist in the service provider's domain, and it is only the external interfaces that have overlapping addresses. The service provider may not be monitoring beyond the edge routers. In this case, the interfaces are all active, but will not be ping-able or routable.

ICMP polling is disabled.

Duplicate WAN Link Addresses

Under older WAN architectures, some customers configured their serial WAN links to all have the same IP addresses. For all intents and purposes, these can be treated the same as the external address case: they routable via ICMP, they are administratively up, etc.

Misconfigured Duplicate Addresses

Finally, there is the use case where there are two nodes that unintentionally have the same IP address configured. This is typically an end-node issue, but is not specifically restricted by type of node. Under this scenario, multiple nodes have operationally up interfaces that have the same IP address configured, AND that IP address is expected to respond to communication via ping or other methods.

NOTE: End-nodes with a single IP address are difficult to distinguish as both attempt to respond to the same address. These scenarios are typically handled by the nodes themselves reporting error conditions.

Requirements for Misconfigured Duplicate Addresses:

- Multiple nodes may have the same IP address in the same management domain.
- netmon will warn the user when multiple active monitored interfaces are detected.

How Extended Topology Handles Other Duplicate Address Scenarios

For the other duplicate IP address scenarios, the behavior is very similar to the duplicate backup IP address scenario.

For any known unreachable addresses, you should create an Extended Topology filter that describes these addresses. A matching class, “UnreachableAddresses”, can be created in the `paConfig.xml`, and the “pingEnable” setting should be “false”.

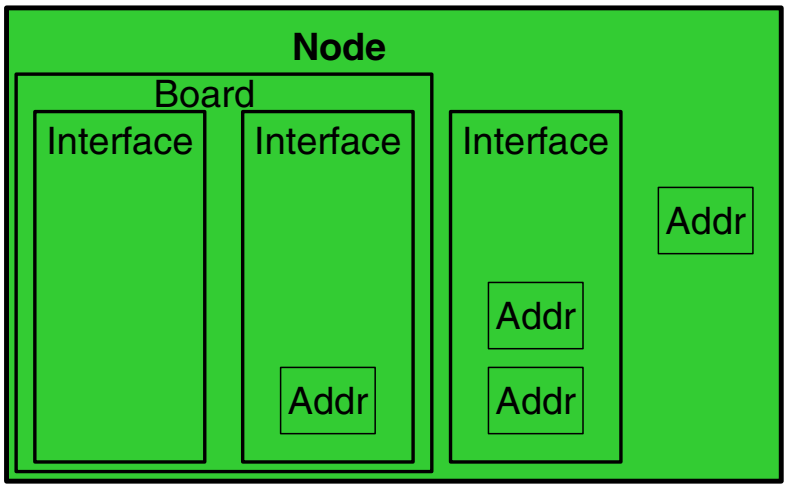
For other duplicate addresses, `netmon` will continue to generate warnings about the duplicates unless the address is put in the `netmon.noDiscover` file.

NOTE: When APA is your primary poller, `netmon` does not generate these warnings.

If polling is enabled on an address that exists on multiple nodes, the IP address state will be maintained based on whether ANY node is responding to ping on that IP address.

Simple Extended Topology Object Model

Simple Extended Topology Model



- Node may or may not have interfaces.
- Interfaces may or may not contain addresses.
- Interfaces and addresses may or may not be polled.
- Interface may be a virtual aggregated interface.

 ©2004 Hewlett-Packard Development Company, L.P.

Extended Topology models multiple levels of containment. This allows it to capture the variety of configurations that are possible on devices.

The smallest unit is the address. An address may be associated with an interface, a board, an aggregated (virtual) interface, or the node itself. Address health comes from a ping.

An interface may contain zero, one, or multiple addresses. The interface may be contained on a board or by the node. The health of an interface is determined by an SNMP query to the node using the MIB II variable IFOperStatus in the IFTable.

A board may contain zero or more interfaces and/or addresses. The board is contained by the node.

Interface and address status are tracked separately. It is possible for an address on an interface not to respond to a ping due to a routing error. However, when you query the node using SNMP (through a different management address) the operational status of the interface is up. Similarly, you could be able to ping an address, but the SNMP query to the management address could fail.

The Polling Engine has a large quantity of mixed requests in its queue -- pings for addresses, SNMP queries for interfaces, SNMP queries for boards, SNMP queries for aggregated ports, varied by system type, IFAdminStatus, connectedness, and current configuration entries. These queries happen in a random order. The Status Analyzer must be prepared to handle failures which arrive to it in any order.

For any failure, the Status Analyzer drives the search for root cause up through the containers to the largest element which fails health verification. For example, if an address does not answer a ping, the Status Analyzer verifies the health of the interface. (These verifications do not pass through the Polling Engine.) If the interface is IFOperUp, the board and/or aggregated port health is verified, if there is one. If they are healthy, the node health is verified by an SNMP query to its management address followed by a ping to the management address.

Note that the Status Analyzer may have already confirmed that a container is down from a previous poll and the drive upward can leap forward to the current step in that previously initiated analysis.

As the search drives upward, an element fails the health verification. That is determined to be the **root cause** of the failure, and an alarm is generated for that element only. At that point, all **child components** are deemed to be unreachable. Their status in the Extended Topology database is set to Unknown (appears as dark blue), and a correlated event is generated for each that the component is unreachable. The **parent containers** receive a propagated status (silently) in the Extended Topology database of Minor and appear as yellow. An interface may propagate its status to multiple parents, for example a board and an aggregated port.

If it is determined that the entire node is down, APA begins Neighbor Analysis to determine whether this node's failure is the root cause or another node is at fault.

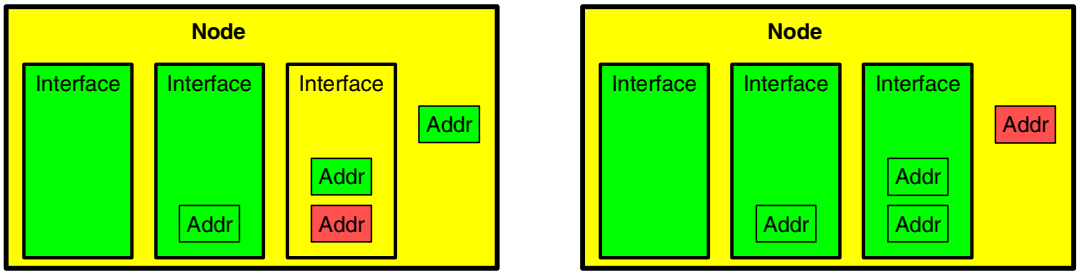
Note that this drive upward differs significantly from `netmon`'s approach of querying each interface and providing a stepwise node degradation. In APA's case, once the root cause is determined, the status is immediately set to Critical.

When status is forwarded to `ovtopmd`, only the root cause failure is forwarded (addresses are forwarded as interfaces). Secondary failures are not forwarded. The `ovw` display system propagates the status from the failed component to its containers according to the `ovw` rules.

1: Address Down, Interface Up, Node Up

1: Address Down

- Polling Engine in APA observes ping failure.
- Sends to Status Analyzer in APA
- Determines interface (or node) is not down
- Ping_NotResponding status in Extended Topology database
- OV_APA_Address_Down alarm
- Pairwise utilized when address comes back up

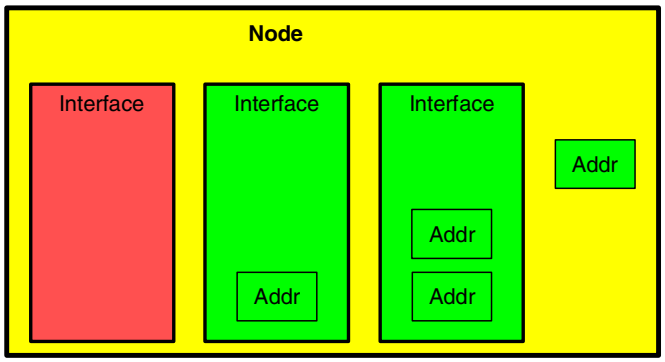


©2004 Hewlett-Packard Development Company, L.P.

2: Interface Down, Node Up

2: Interface with No Address Down

- Polling Engine has nothing to ping.
- Polling Engine observes ifOperStatus is Down
- Sends to Status Analyzer
- Determines node is not down
- Critical status in Extended Topology database
- OV_APA_Interface_Down alarm
- Pairwise utilized when interface comes back up

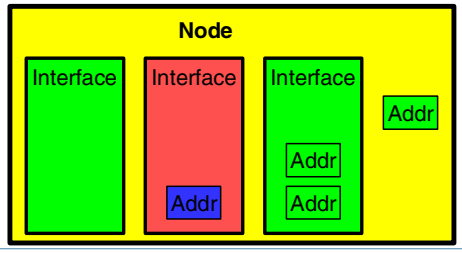


©2004 Hewlett-Packard Development Company, L.P.

3: Address Down, Interface Down, Node Up

3: Interface Down

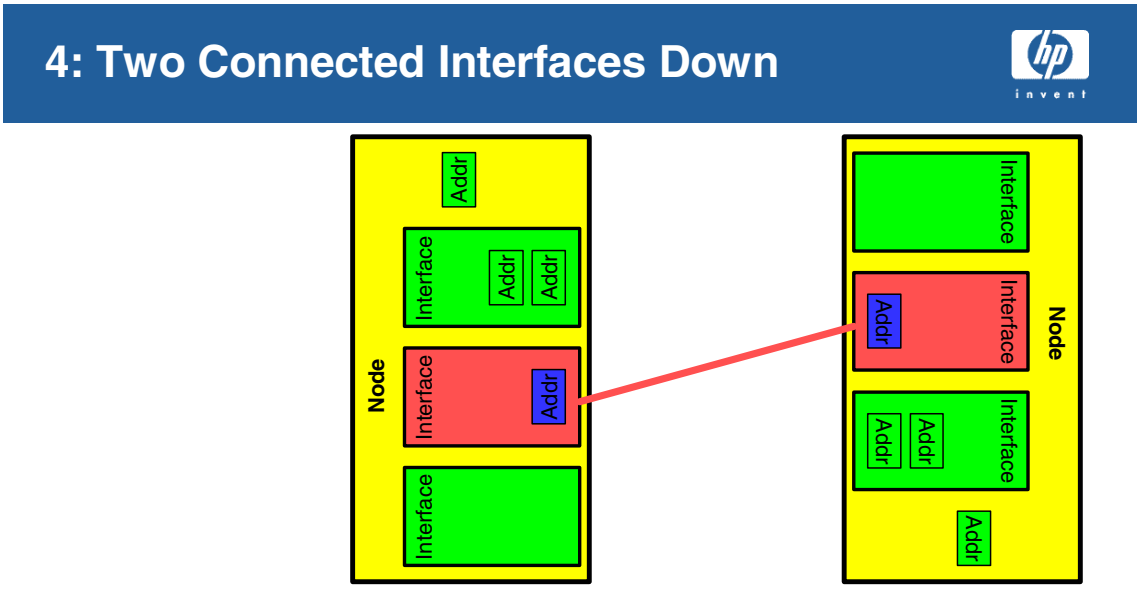
- Polling Engine observes ping failure and ifOperStatus is Down
- Sends to Status Analyzer
- Determines node is not down
- Critical interface status in Extended Topology database
- OV_APA_Interface_Down alarm
- Ping_Unreachable status for address
- Address unreachable alarm correlated with interface down alarm
- Pairwise utilized when interface comes back up
- Pairwise not needed for secondary failure (unreachable) events.



U5089S ©2004 Hewlett-Packard Development Company, L.P. 35

When you read the Alarm Browser, the Down alarm always indicates the root cause. Unreachable events all indicate secondary failures.

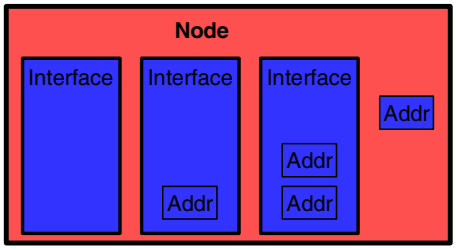
4: Two Connected Interfaces Down



Since APA can still contact the node, no Neighbor Analysis is triggered.

5: Node Down

5: Node Down



- Polling Engine observes ping failure or no SNMP response
- Sends to Status Analyzer
- Determines node is down
  - SNMP then ping to management address
- Critical node status in database
- Interface status is Unknown
- Address status is ping_unreachable
- Begin Neighbor Analysis
  - No alarms generated until Neighbor Analysis completes

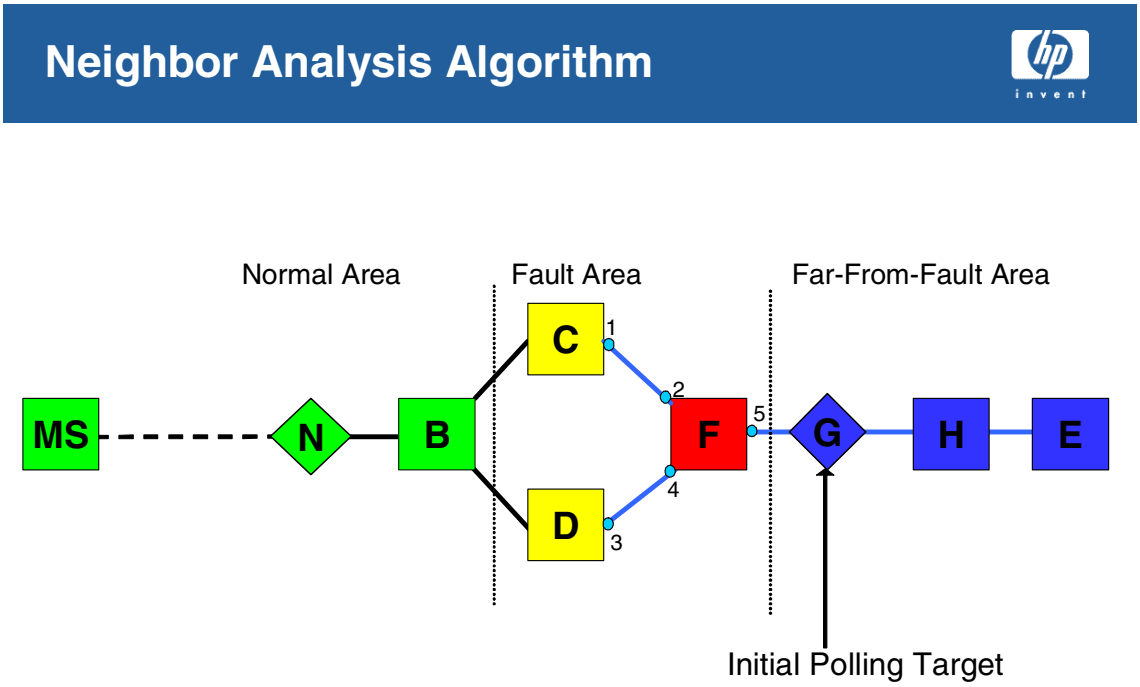
 ©2004 Hewlett-Packard Development Company, L.P.

Once APA determines that the entire node is down, it begins a Neighbor Analysis to determine if this node is unreachable due to an upstream failure.

If the SNMP query fails but the follow-on ping succeeds, an SNMP_Not_Responding alarm is issued. So, for example, if you have an incorrect community string (after a correct discovery somehow), the node turns red, but the alarm in the Alarm Browser is SNMP_Not_Responding, not Node_Down.

It is possible that the management address is unreachable due to a routing error, but other interfaces are in fact up. You can get a false positive on Node_Down. It is important to set your management address to the one most reachable from the management station, not one further downstream.

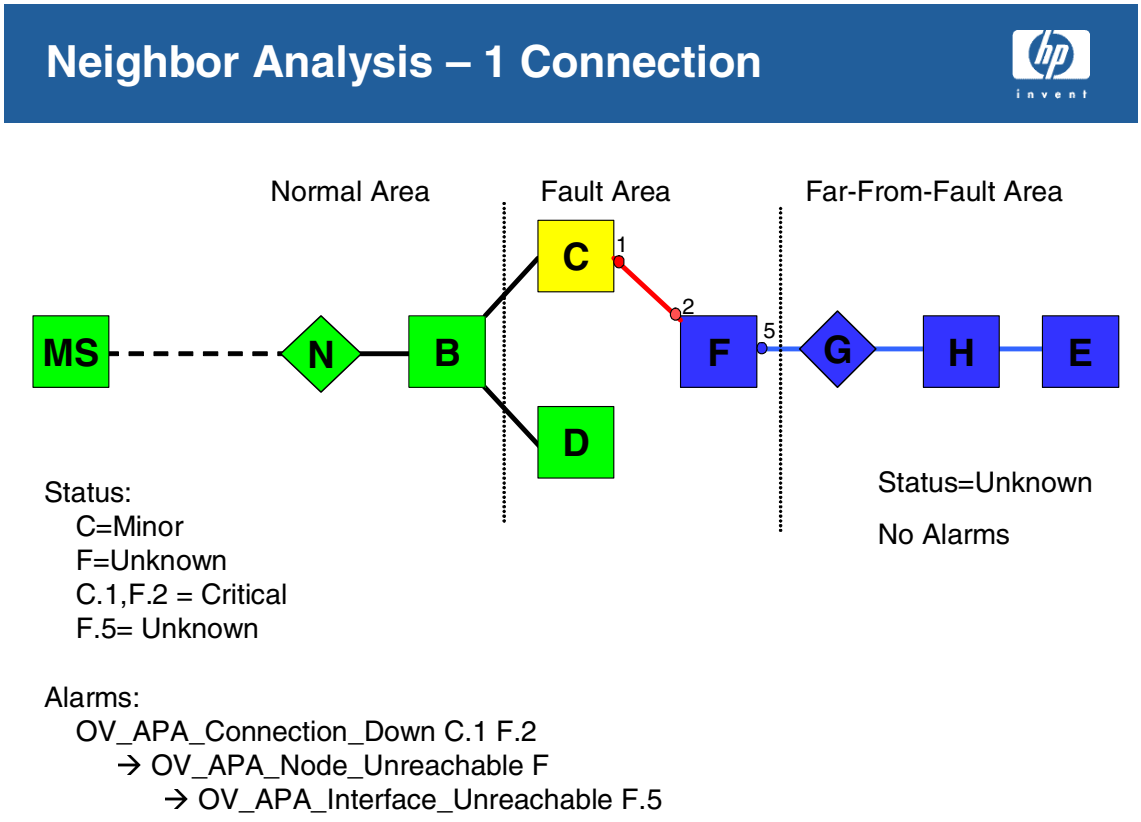
Neighbor Analysis Algorithm



Neighbor Analysis begins by determining the area in which the fault occurs. APA examines the Extended Topology database to determine all the nodes connected to this node. For each of those, the node health is verified. If all of the neighbors are Node_Down, APA is looking beyond the fault area, surrounded by secondary failures. APA examines the neighbors of the node nearest the management station, working backward until at least one node is up.

The Fault Area is defined as the area where at least one node is Node_Up and one node is Node_Down.

Neighbor Analysis: One Connection

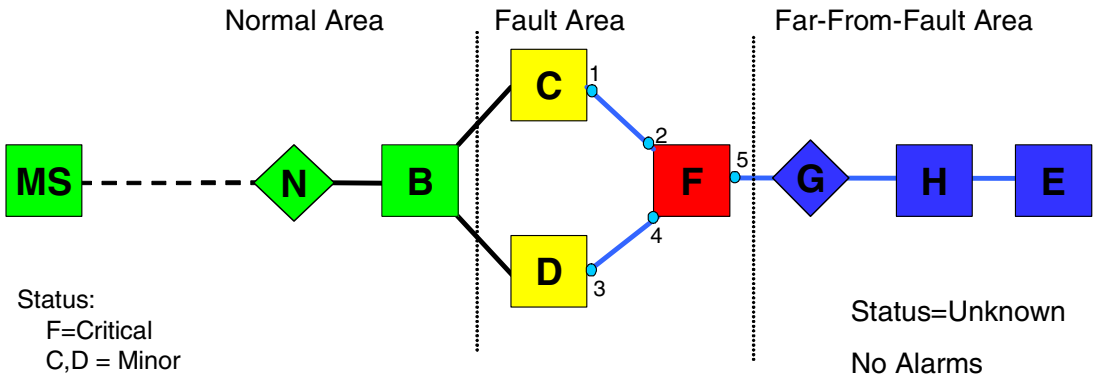


When the nearest node which is Node_Down has only one neighbor, APA queries that neighbor to determine the IFOperStatus of the interface connected to the down node. If the interface is down, APA marks the connection as the root cause, and sets the downstream node to Unknown.

is it possible for the IFOperStatus to be up? Is this really the check that's done? Or does APA just assume its the connection that's down?

Neighbor Analysis: Two Connections

Neighbor Analysis – Multiple Connections



Status:
- F=Critical
- C,D = Minor
- F.2, F.4, F.5 = Unknown
- C.1, D.3 = Unknown

Alarms:
- OV_APA_Node_Down F
  - → OV_APA_Connection_Unreachable C.1 F.2
  - → OV_APA_Connection_Unreachable D.3 F.4
  - → OV_APA_Interface_Unreachable F.5
    - → OV_APA_Address_Unreachable

Status=Unknown

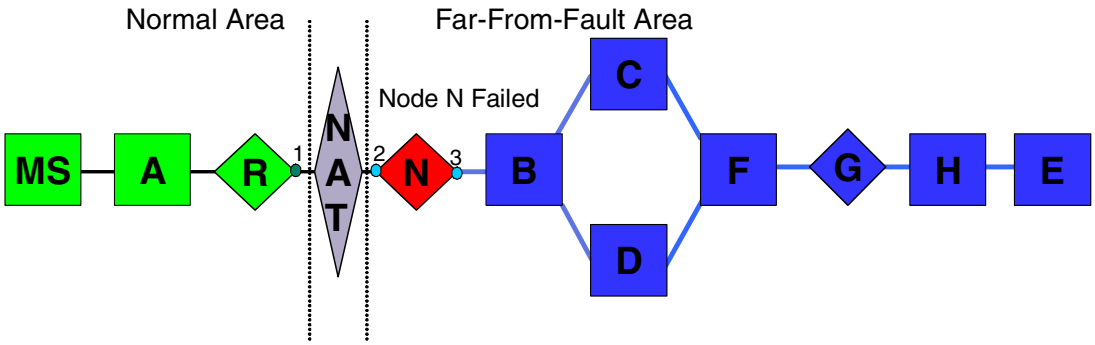
No Alarms

©2004 Hewlett-Packard Development Company, L.P.

When a down node has more than one neighbor, APA queries to see whether either neighbor has a working connection to the node in question. If neither neighbor can access the node, the node is marked as the root cause and the connections become secondary failures.

Neighbor Analysis: OAD NextHop

Neighbor Analysis – DupIP NextHop



Not Discovered

or

Not Properly Connected

Solution: Define N as NextHop in dupip.conf

©2004 Hewlett-Packard Development Company, L.P.

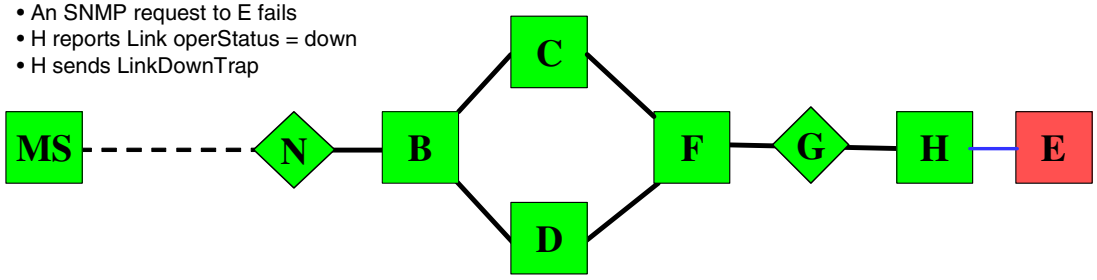
OAD environments present a special concern to NNM because the NAT device tries to be invisible. In this case, if the NAT device fails or the inbound interface on the router fails, it has no neighbors to query. The inbound interface on the router is seen as unconnected, and therefore unpolled. Because the management address cannot respond the node could be marked as the root cause. To avoid this, designate the first router(s) in the domain with the NextHop keyword in the `dupip.conf` file.

Neighbor Analysis: End Node Down

Neighbor Analysis – End Node Down

- End Nodes and Access ports are handled differently.
- Behavior is Configurable.

Symptoms:
- Ping E Fails
- An SNMP request to E fails
- H reports Link operStatus = down
- H sends LinkDownTrap



OV_APA_Node_Down E
→ OV_APA_Connection_Unreachable H-E

©2004 Hewlett-Packard Development Company, L.P.

Turning off desktop systems at night or having servers fail should not create unnecessary troubleshooting for switch operators. Rather than have the Neighbor Analysis return a Connection_Down (because the node in question has only one neighbor), APA contains special code for end nodes that returns a Node_Down.

Lab Exercises

Cisco Board Discovery

Assumptions:

- `$OV_CONTRIB/NNM7labs/initialsetup.ovpl` has been executed.

Directions

1. Change working directory to `$OV_CONTRIB/NNM7labs/Lab_deploy`:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_deploy
   ```

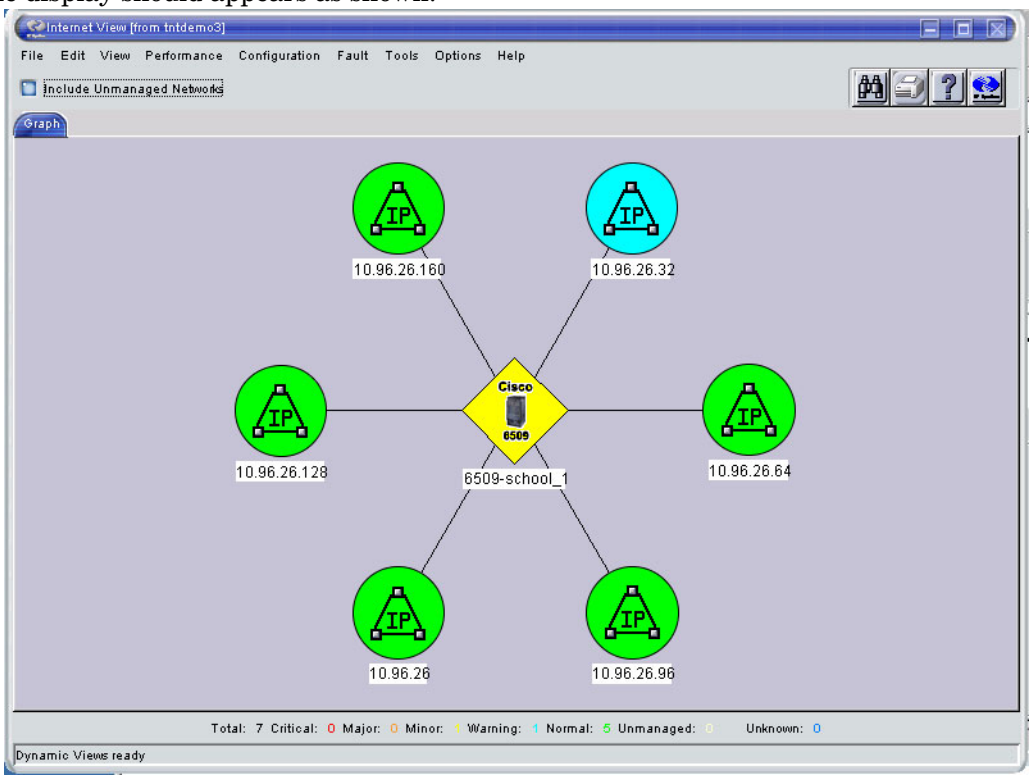
2. When the instructor has given you the go-ahead, execute the setup script:

   ```
   setupLab_deploy.ovpl
   ```

3. When this completes, open an Internet View from Home Base.
4. Verify proper and complete discovery of the simulated network. Locate any symbol that may

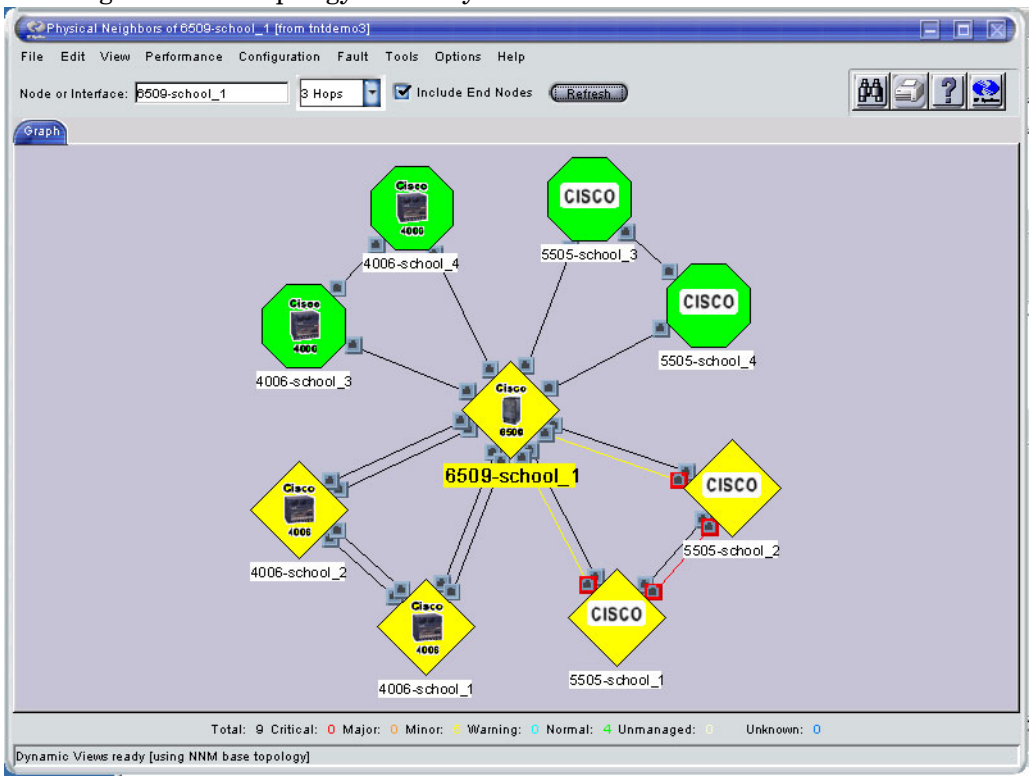
be unmanaged or unknown. If unmanaged, select it, and then use the ovw menu `Edit:Manage`, or the dynamic view menu `File:Topology:Manage`. For any node which shows as a blank square, select the node and `Fault:Network Connectivity:Poll Node`.

5. The display should appears as shown.



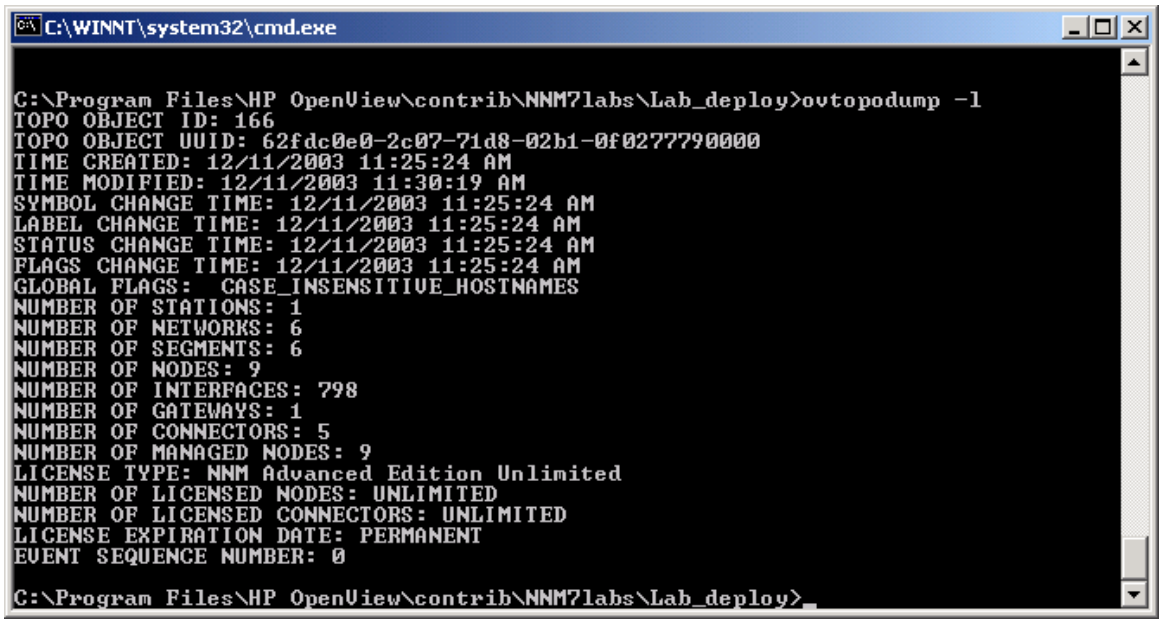
6. Execute a Neighbor view of the device 6509-school_1 with 3 hops and include end nodes, prior

to executing Extended Topology discovery.



7. Confirm correct topology has been discovered:

```
ovtopodump -l
```



8. When you have confirmed that all devices are managed and appear as shown in the above figures, commence the Extended Topology discovery from Home Base.

NOTE You will run several iterations of Extended Topology discovery throughout these labs. Each execution of the discovery will give you an opportunity to use the support tools to view progress of the discovery processes and agents. These

tools will be especially important in the labs that simulate discovery problems.

9. From the NNM Home Base, bring up a Node View. Select “All” for the Show Nodes field and “Normal” for the Status >= field, then hit the Refresh button. Note the status of the nodes and links in the Node View.
10. Right click on the 6509-school_1 node, and select Details.
11. In the Node Details window, click on the Boards link in the top frame.
12. Note the board detail information in this section of the Node Details window. Note the status of the board with interface ID 205.
13. Close the Node Details window.
14. When the instructor gives you the go-ahead, right click on the 6509-school_1 node. Select “APA Status Poll” from the resulting pop-up menu. Wait for the APA status poll to complete in the resulting APA Demand Poll window.
15. Repeat step 9to bring up a Node View of your entire network. Note the new status of the nodes and links in the Node View.
16. Right click on the 6509-school_1 node, and select Details.
17. In the Node Details window, click on the Boards link in the top frame.

Status Determination for Switches

Default Status Determination for Switches

Port Connection	Admin Status	Polling Configuration	Status
Unconnected	Up	Disabled	Not Monitored
		Enabled	Operational status
	Down at Last Discovery	Disabled	Not Monitored
		Enabled	Disabled
	Down Since Last Discovery (up at discovery)	Disabled	Not Monitored
		Enabled	Disabled
Connected	Up	Disabled	Not Monitored
		Enabled	Operational status
	Down at Last Discovery	Disabled	Not Monitored
		Enabled	Disabled
	Down Since Last Discovery (up at discovery)	Disabled	Not Monitored
		Enabled	Disabled

Configuration Default Visualization Status Color

U5089S ©2004 Hewlett-Packard Development Company, L.P. 2

The heuristics used by APA to determine polling and status are:

1. Unconnected switch ports are never polled by default.
   - Status is No Status.
   - Visualization is Not Monitored.
   - If you configure APA to poll unconnected ports, then it proceeds to the next rule.
2. Ports that are administratively down at the time of discovery are not polled.
   - Status is No Status.
   - Visualization is Not Monitored.
   - If you configure APA to poll administratively down ports,
     - They appear Disabled.
     - They are polled slowly (every 6 hours by default).
     - If one changes to administratively up and a trap comes from the device directly or through syslog, its status changes immediately.
     - If one changes to administratively up and there is no trap, it is recognized at the next polling cycle.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

— A port which changes to administratively up status goes to the operational status, and it keeps the slow polling cycle. At the next discovery, it will be discovered in an administratively up state and that configuration takes effect.

— The polling interval is also recalculated if you restart `ovet_poll`.

3. Ports that are administratively up at the time of discovery are polled.

- Status reflects Operational Status as long as administrative status is up.
- If administrative status changes to down after discovery,

— The device is still polled at the short interval until APA is restarted.

— Its status is Disabled.

— An alarm is sent to the Alarm Browser indicating that the port has become Disabled.

Additionally, objects which were monitored, but which are no longer monitored, should have any outstanding events cleared.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Enable or Disable SNMP Polling for Unconnected Switch Ports

Control Polling of Unconnected Ports

•Configure ifAdminStatus manually on the managed device

•By default:
- Switches: do not poll unconnected interfaces
- Routers: do not poll unconnected interfaces that are adminDown
- Routers: DO poll unconnected interfaces that are adminUp

•Edit paConfig.xml

•Find UnconnectedAdminUpSwitchIF

•Change to false or true

•Note: Switch interfaces default to adminUp when unconnected, so you must commit to manually controlling the status on the switch.

•Admin status is only updated during discovery. Status polling only updates operStatus.

 ©2004 Hewlett-Packard Development Company, L.P.

There are times when some devices will not be in the extended topology database, or will not otherwise connect correctly. An example of this is if Extended Topology discovers information from an OAD environment, but cannot talk to some of the end nodes using SNMP. The result is confusion about whether there are any nodes connected to certain ports on a switch.

APA provides a solution for this problem. APA decides whether to poll a device using attributes from the node and interface. For example, APA knows if an interface's port is connected to another node in the extended topology and knows the class of the device it is polling. You can configure APA to SNMP poll switch ports that are *either* known to be connected to another node in the extended topology or have an `ifAdminStatus` of `up`.

This solution involves editing the `paConfig.xml` file. This solution assumes that you manually configure the `ifAdminStatus` parameter on the switches you want to poll using SNMP.

To implement this solution, use the following procedure:

1. Manually configure the `ifAdminStatus` parameter on your switches. For example, if you want APA monitor a switch port using SNMP, you must manually set its `ifAdminStatus` to `up`.
2. Make sure you have APA enabled.
3. As `Administrator` or `root`, edit `$OV_CONF/nnmet/paConfig.xml`.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

4. Search for `UnconnectedAdminUpSwitchIF`

   You should see the following:

```
 <classSpecification>
<filterName>UnconnectedAdminUpSwitchIF</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>
```

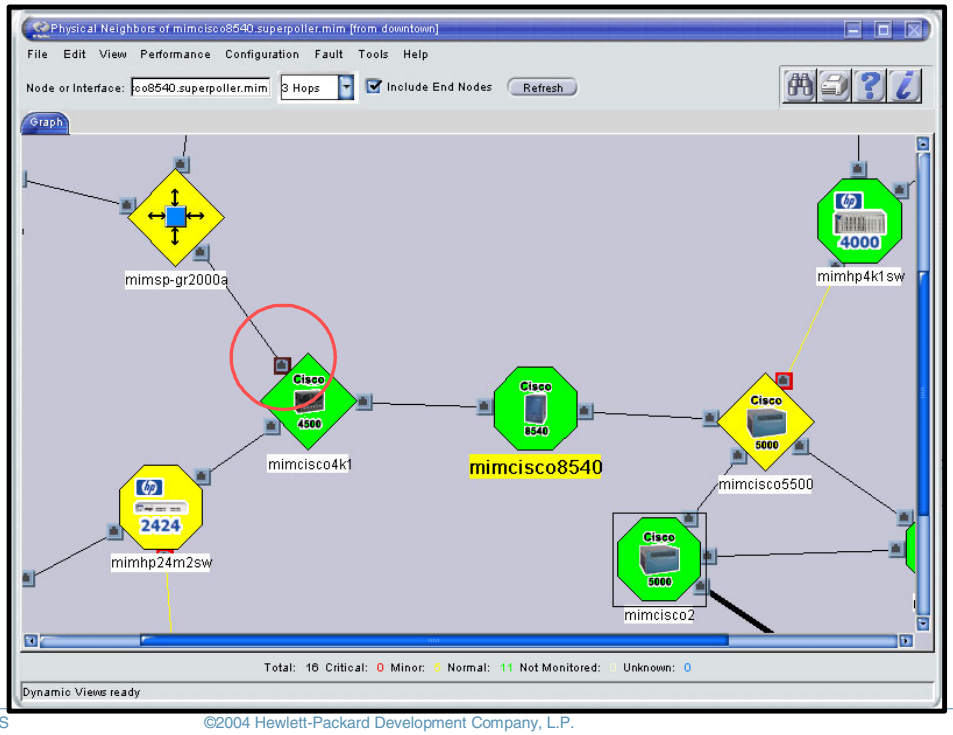
5. Modify the bold `false` to `true`.
6. As `Administrator` or `root`, run the following commands:
   - `ovstop -c ovet_poll`
   - `ovstart -c ovet_poll`

     This applies your changes to the APA polling process.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Example Admin Down Interface

Example Admin Down Interface



©2004 Hewlett-Packard Development Company, L.P.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Dynamic Handling of Unconnected Ports

An internet service provider may want to monitor the network infrastructure of customer networks. They typically have the customer switches and routers directly in their topology, but do not have access to the end nodes in the customers' environments. ISPs desire the capability of monitoring ports that are connected in the environment without the ports being connected in the Extended Topology database. Further, when a port transitions connected state in the environment, they want status reflected "correctly" on the map and alarms to be generated correctly.

For the NNM AE 7.01 release, these requirements were partially met:

- Interface Admin status was set as part of the discovery process. By default, unconnected ports were not polled. Polling of administrative up but still unconnected ports was enabled via configuration.
- Anything discovered as administratively down remained unpolled.
- If an interface's administrative status changed, you had to rediscover the network (or zone) to initiate a change in polling.

In NNM 7.5, you can configure APA to start polling connected ports immediately without waiting for a rediscovery. When an interface is marked administratively up, its status is based on its operational status and/or ping status. If you configure APA to monitor (slowly) interfaces marked administratively down, then its status is an administratively down status (disabled).

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

APA polls administratively down interfaces slowly and relies on event triggered polling to maintain the status until the next rediscovery. This allows for dynamic transitions, which disabling polling does not allow.

If an interface has polling enabled, APA initiates a status update based on link up/link down events immediately, independent of the polling cycle. Thus, an administratively down interface could still be polled and still see status updates if trap handling is enabled (if the device is configured to send traps or syslog messages to the management station).

If one end of a connection is administratively down, and the other end is operationally down, the connection is considered “down”.

Configuring Handling of Unconnected Ports

To enable monitoring of unconnected ports on switches, edit the `UnconnectedAdminUpOrTestSwitchIF` portion of `paConfig.xml` and set `snmpEnable` to true.

To enable monitoring of administratively down ports on switches, edit the `UnconnectedAdminDownSwitchIF` portion of `paConfig.xml` and set `snmpEnable` to true.

Stop and restart `ovet_poll` for your changes to take effect.

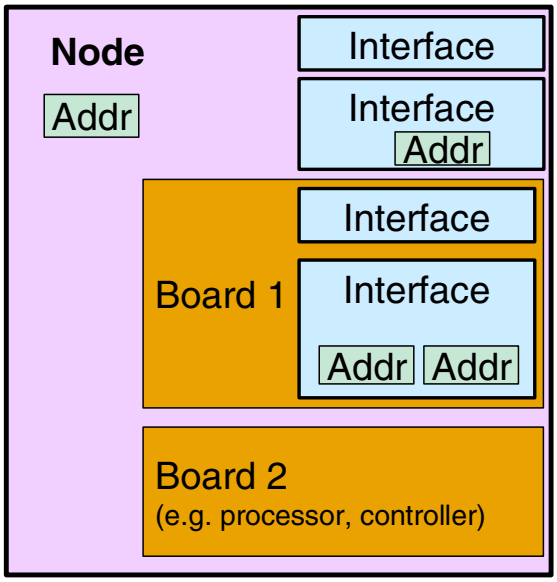
Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Managing Cisco Boards

Managing Cisco Boards

- Cisco-only feature
- Referred to as board, card, module
- Multiple proprietary MIBs

- Fewer alarms
- More specific fault indication



U5089S ©2004 Hewlett-Packard Development Company, L.P. 6

Communication devices such as routers or switches are composed of multiple components. The commonly seen components are Board, Chassis, Power Supply, Fan, Port etc.

Chassis contain slots where boards are plugged-in. There are many different types of boards: Processor board, Controller board, Memory board, I/O board etc. I/O boards contain ports which interface with the physical media to transmit and receive data packets. Boards can be specialized software modules such as Cisco's Route Switch Module, which provides layer 3 routing. Boards can contain sub-boards.

From network management model perspective, Board, Chassis, Power Supply, Port etc. are all physical components in the sense that the component can be physically identified in a switch or router.

Some of these components have logical counterparts. For instance, Port is the physical connection point to the wire whereas Interface is the logical correspondence; it models the communication characteristic of the Port.

In prior NNM releases, these two concepts, logical and physical, are often mixed in the same topology object. NNM 7.5 begins to distinguish these concepts into separate objects in the Extended Topology database.

Frequently, the terms card, module, and board are used interchangeably.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Discovering Cisco Boards

- Model boards as objects in Extended Topology database
- Obtain board ID via SNMP for each port
- Cisco switch agent discovers switches and routers
- No sub-board modeling. Ports on a sub-board appear to be contained by the board.

 ©2004 Hewlett-Packard Development Company, L.P.

Board discovery is a Cisco-only feature. NNM models boards as objects.

Each port has a board ID which indicates its container. The board IDs were “derived” in NNM 7.0; now they are obtained via SNMP directly.

In NNM 7.5, the Cisco switch agent has been changed to support both switches and routers.

A board may have multiple sub-boards and a sub-board can have multiple ports. Cisco routers support this type of containment by CISCO-RHINO-MIB (marginally), but switches don’t have similar support in their MIBs. Therefore, sub-boards appear as boards. If a Cisco device has a board:sub-boards:ports relationship, the sub-boards are treated as top-level boards with their ports contained on them.

Cisco Routing-Switch Modules (RSMs) which have their own IP address and SNMP agent are modeled as a separate node

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Monitoring Cisco Boards

APA `ovet_poll` can monitor and analyze interface failures associated with a board and generate the appropriate status and alarms to reflect a fault. For example, if a board goes down which contains 10 interfaces and the node is reachable, then a single board alarm is emitted instead of 10 interface down alarms.

Because the board hierarchy is not modeled in NNM 7.5, when a board goes down, an APA_Board_Down alarm is generated for the board and all subBoards.

APA uses the appropriate Cisco MIB to obtain the device's board or subBoard status. The OverallStatus (what the GUI displays) may or may not directly correspond to what the device communicates because APA considers the context of the failure during analysis.

No board or subBoard degradation alarm will be emitted by APA as is done with the aggregated port support. However, the status of the board or subBoard may be set to Minor.

In this release, APA does not communicate the notion of ports to the user. APA will always drill down through the port to get to the interface of interest.

Board Status propagates to Node Status just like interface status propagates.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Configuring Board Polling

By default polling is enabled for boards.

You can disable polling of boards by editing `paConfig.xml` and searching for AllBoards. Uncomment the block to disable polling.

You cannot configure the frequency of board polling.

If you disable monitoring of boards, the board status will be No Status (Not Monitored in displays) and the interface handling appears as in NNM 7.01 where the interfaces appear directly contained in the node.

NOTE The number of boards being polled shows in the APA statistics tab of Home Base.

The APA statistics tab shows the last update time of the information. On a system with no load, this defaults to every 5 minutes. However, during busy polling times, the statistics may only be updated hourly.

Correlation of Board Events

When a board fails on a Cisco device, board and link failure traps or syslog messages may also be generated by the device. The customer would like to see the root cause APA event identifying the board that failed at the top level of the Alarm Browser. The Cisco board down traps and syslog messages, as well as related APA status events, are correlated under the root cause APA board failure event. (If both a trap and a syslog message are received, the second to arrive is correlated under the first to arrive.) NNM also monitors the rate of board failures and notifies you if a board if flapping.

APA sets board status for display in the GUI and issues board status alarms. The new APA alarms are:

OV_APA_boardDown

OV_APA_boardUnreachable

OV_APA_boardUp

OV_APA_boardRemoved

The PairWise ECS circuit processes board events such that boardUp cancels the other board events. (Events which are log only by default are not correlated to save processing time.) The deDup configuration has been updated too.

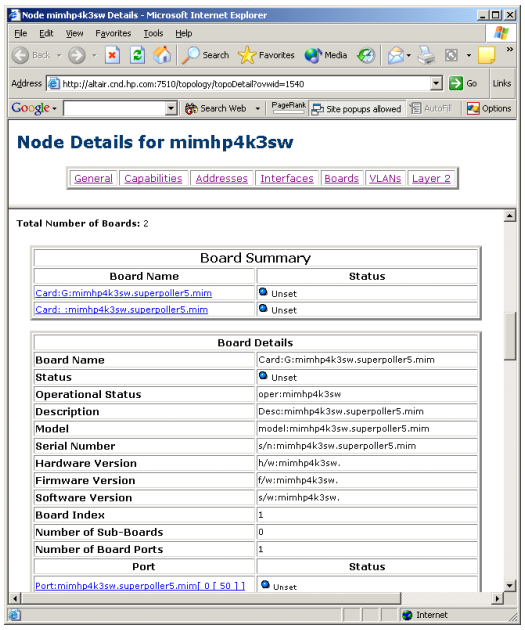
Note: The interaction of board analysis and aggregated port analysis can be quite complicated with the suite of devices and configurations in OpenView customer networks.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Board Visualization

Board Visualization

- Display on Node Details page of board data
- Board to sub-board relationship is not modeled.
- Show board attributes, status and associated ports.
- Show board count on Topology Summary page.



 ©2004 Hewlett-Packard Development Company, L.P.

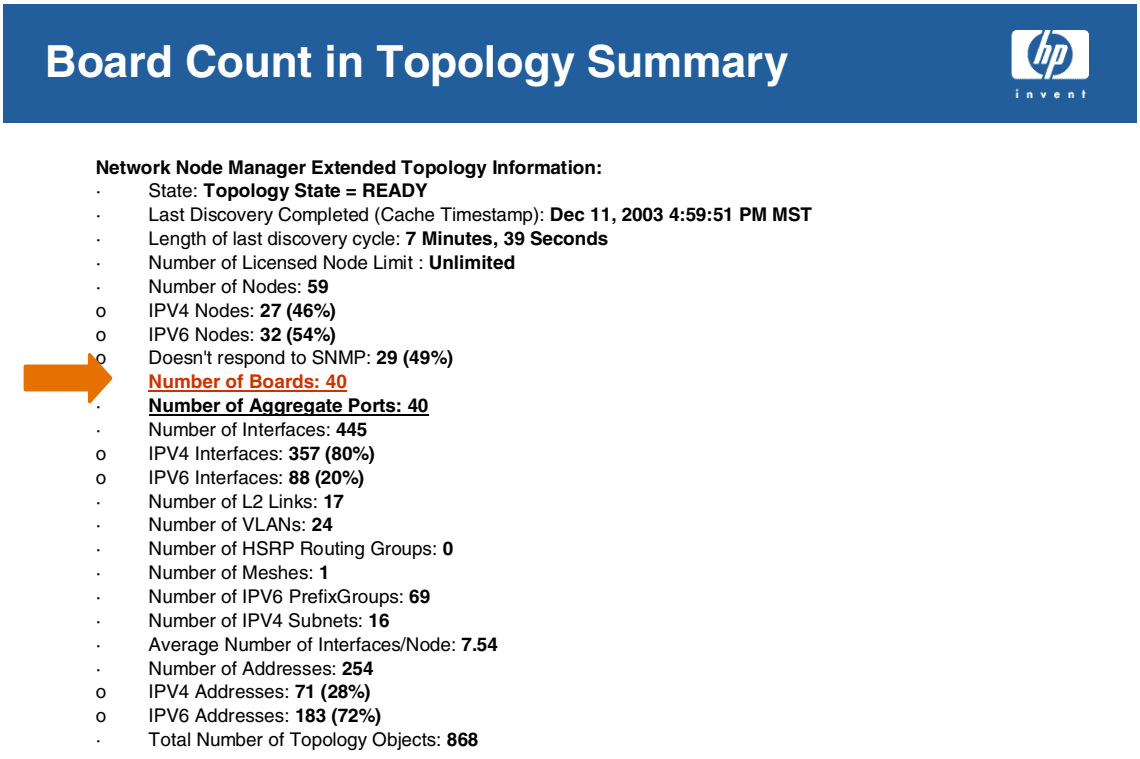
Board visualization is presented on the Node Details web page and as a board count on the Topology Summary page.

If a board has a malfunction, APA emits an event which is displayed in the Alarm Browser. The source field of the trap contains the node that has the malfunctioning board. You can select the alarm and launch a Neighbor View focused on the malfunctioning node.

Alternatively, an open Neighbor View may alert an operator to a malfunctioning node containing a Cisco board. Opening the node takes you to Node Details where you can select Boards from the hyperlinks listed in the top frame and view the board status and details.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Board Count in Topology Summary



To determine how many boards are known to Extended Topology, launch a Topology Summary. The board count is listed in the displayed information.

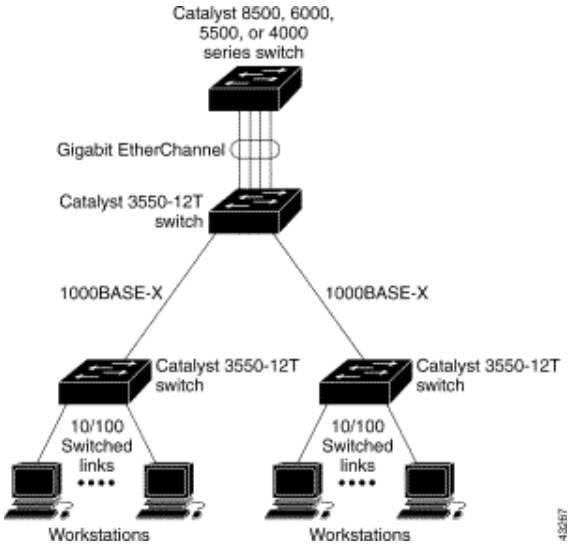
Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Aggregated Ports

Aggregated Ports

- Cisco-only support
- Combining multiple physical ports into one logical link for increased bandwidth, load sharing, load balancing, and high availability
- NOT trunks, which are single links carrying multiple VLANs

- The following terms can be used interchangeably
  - Link Aggregation : IEEE 802.3ad / LACP
  - Port Aggregation : Cisco / PAgP
  - MultiLink Trunk : Nortel
- MultiLink Trunking VS. VLAN Trunking
  - MultiLink Trunking : One logical interface with multiple physical port members
  - VLAN Trunking : One single physical interface carries multiple VLANs.



U5089S ©2004 Hewlett-Packard Development Company, L.P. 11

Cisco port aggregation is the combining of multiple ports into one logical port-channel or aggregate port (sometimes referred to as an AP).

Conversely, Cisco trunks are single links that carry traffic for multiple VLANs. So, in a Cisco terminology, multiple trunks acting as one logical trunk form a port aggregation. NNM is not supplying Cisco trunk management.

Port aggregation is a Cisco-only feature. PAGP is the only supported form.

Supported protocols:

- PAGP

  According to Cisco.com, “PAgP (Port Aggregation Protocol) [is] A protocol that aids in the automatic creation of Fast EtherChannel links. PAgP packets are sent between Fast EtherChannel-capable ports in order to negotiate the forming of a channel.”

  NNM supports this for switches and routers. You need the 2004 version of IOS to get the best PAGP MIB accuracy.

- LACP (not used)

  According to Cisco.com, “Link Aggregation Control Protocol (LACP) is part of an IEEE specification (802.3ad) that allows you to bundle several physical ports together to form a single logical channel. LACP allows a switch to negotiate an automatic bundle by sending

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

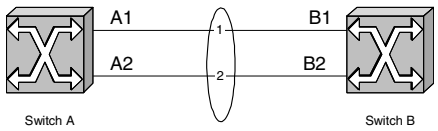
LACP packets to the peer. It performs a similar function as Port Aggregation Protocol (PAgP) with Cisco EtherChannel."

This MIB is currently supported by only one Cisco device model, and that support is incomplete.

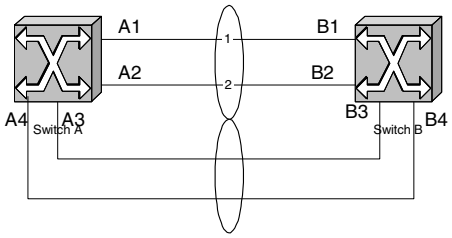
Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Monitoring Aggregated Ports

Monitoring Aggregated Ports



One port aggregation
between two switches



Two port aggregations
between two switches

- Fewer Alarms
- More specific fault indication
- Higher priority fault indicator
- Show degradation

©2004 Hewlett-Packard Development Company, L.P.

Extended Topology discovers each interface (A1, A2, etc., B1, B2, etc.) and two logical aggregated interfaces: LA and LB. The neighbor information is:

[2]A1 – B1

[2]A2 – B2

[2]LA – LB

Note that the physical connectivity is not reported by the MIBs. NNM pairs the physical interfaces by their interface speed and suggests the connectivity for them. The MIBs return only the logical connection.

The logical interfaces LA and LB are a type of interface, and have ifIndex, ifName, ifDesc, ifOperStatus, ifAdminStatus etc. (MAC address is not present.).

APA reconciles the status values obtained from the physical interfaces with the values obtained from the logical interface to develop an OverallStatus value for the logical interface. APA sets overall status on the logical interface for display and issues new aggregated port status alarms.

The status of the virtual interface goes to Minor if the ifOperStatus of the virtual interface is non-UP or if any of the physical interfaces are non-UP.

The status of the virtual interface goes to Critical only when ALL physical (polled) interfaces go to

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

critical or unreachable.

	Local Port	Remote Port	Local Logical IF	Remote Logical IF	Alarms
One port goes down	Critical		Minor	Normal	aggregatedPortDegraded vIf1 – vIf2 // vIf2 is virtual IF on the other node ‡ If3 Down // Coorelation from APA & ConnectorDown
Port is fixed	Normal		Normal	Normal	aggregatedPortUp vif1 – vif2 ‡aggregatedPortDegraded vif1 – vif2 // Correlation from PairWise ‡ If3 Down // Correlation from APA & ConnectorDown ‡ If3 Up // Correlation from APA & ConnectorDown
Both ports down on one link	Critical	Critical	Minor	Minor	aggregatedPortDegraded vIf1 – vIf2 // vIdf2 is on the other node ‡ connDown If3-If4 // Coorelation from APA & ConnectorDown
Both ports fixed	Normal	Normal	Normal	Normal	aggregatedPortUp vif1 – vif2 ‡aggregatedPortDegraded vif1 – vif2 // Correlation from PairWise ‡ connDown If3-If4 // Coorelation from APA & ConnectorDown ‡ connUp If3-If4 // Coorelation from APA & ConnectorDown

Configuring Aggregated Port Polling

You can create APA polling configurations for aggregated interfaces using Extended Topology Topology filters based upon the following new attributes and corresponding out-of-the box filters:

```
isAggregatedIF// logical interface
```

```
isPartOfAggregatedIF// physical interface
```

For example, you may want to poll aggregated interfaces more or less frequently than non-aggregated interfaces.

Correlation of Aggregated Port Events

When an aggregated link fails or its performance is degraded because of underlying interface failures on a device, the customer would like to see an APA aggregated port status event identifying the aggregated port problem. However, there are a number of other events that can be generated with an aggregated link failure scenario. The customer only wants to see the APA aggregated port status event. The other events, such as LinkDown traps and aggregated link

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

related syslog messages, are correlated under the APA aggregated port status event. These sympathetic events are deleted from the Alarm Browser if they were being displayed.

Logical aggregated port alarms and physical interface alarms are correlated using the ECS PairWise correlation, a MultiSource correlator, and deDup. They provide one top level alarm per fault.

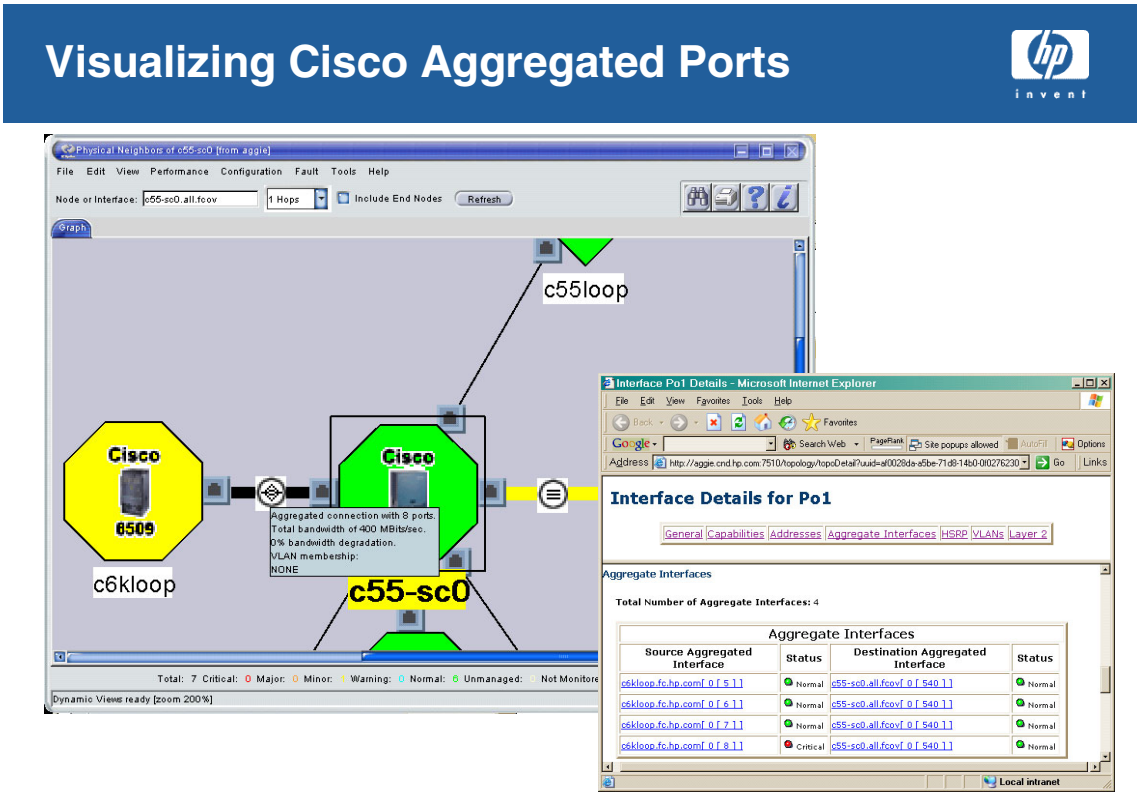
Aggregations are used for increasing bandwidth or reliability. Therefore these connections tend to be very important to the customer. The customer needs to know about degradation of these aggregated links. Individual ifDown alarms are rolled up into the aggregated port alarm to minimize clutter in the Alarm Browser.

The new aggregation alarms are:

```
OV_APA_AggregatedPortUp  E1 E2

OV_APA_AggregatedPortDown   E1 E2

OV_APA_AggregatedPortDegraded E1 E2// Specifies how many physical Ifs are Up/Down on each node.
```


Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Visualizing Cisco Aggregated Ports



 ©2004 Hewlett-Packard Development Company, L.P.

Dynamic views represent aggregated ports as thick L2 edges. They have status and the port tool-tip shows some details about the aggregation. Opening one port of the aggregated port displays the Interface Details view.

Hovering over the edge representing the aggregated port displays the count of ports, maximum bandwidth, percent degraded, and associated VLAN. When you hover over an aggregated port port, the tool-tip shows details about the aggregation, including interface information as well as the participating physical interface names, bandwidth maximum, and media type.

Aggregate links are identified by a circle icon with a single line branching to three lines and reforming on the opposite side as a single line. Double-clicking on an aggregate port (or any other non-redundant port) launches an interface details page in a new window.

Redundant L2 links (that is, separate connections between the same nodes) are displayed as thick lines in order to differentiate them from aggregated ports. Hovering over the edge shows port and edge count. Hovering over port shows the group of individual interfaces. Redundant edges are identified by a circle icon with three parallel lines in it. Double-clicking on a redundant port launches an interface detail page (in a new browser) for each member interface.

You may see an APA Aggregated Port Status alarm in the Alarm Browser. The source field of the alarm contains the node that has the malfunctioning aggregated port. You can select the alarm and launch a Neighbor View with the node highlighted.

From the dynamic view, you can double-click the aggregated port to display interface details. The

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

details include the list of participating interfaces. You can also open the node details and click the Aggregated Ports hyperlink at the top. This displays a list of aggregated port interfaces. Selecting one starts the interface details display. The Aggregated Ports table does not show port-to-port connections as these are unavailable from the device. The ports on each device participating in aggregation are sorted by interface speed.

Alternatively, you may have a Neighbor View open when the edge representing the aggregated port turns yellow or red. Hovering over the edge shows its port count, maximum physical bandwidth and its current degradation percentage (e.g., 25% degraded). Where a mismatch in local-remote port speeds can be determined, a caution will also be printed in the tool-tip.

Changes to Previous Behavior:

You can now open all ports, which displays an Interface Details page. This is functionality works for both aggregated and non-aggregated ports.

Hovering over edges shows a tool-tip listing the maximum and degraded level of bandwidth. This information differs from the NNM 7.01 product where hovering popped up a tool-tip showing the VLAN and number of ports associated with the edge.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

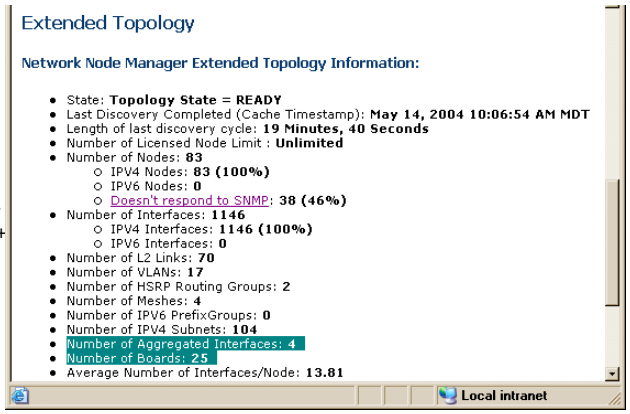
Aggregated Ports in ovet_topodump.ovpl

Aggregated Ports in ovet_topodump.ovpl

ovet_topodump.ovpl -nodeif -detail

++++++++++++++++Interface++++++++++++++++++++
IFName:
ObjID:123
IFAlias:
IFDescription:
Status:Normal
IfIndex:
IfType:
AggregatedInterfaceObjID:
Capability:
------------------------Interface----------------------------------
++++++++++++++++Interface++++++++++++++++++++
IFName:
ObjID:124
IFAlias:
IFDescription:
Status:Normal
IfIndex:
IfType:
AggregatedInterfaceObjID:123
Capability:isAggregatedIF
------------------------Interface----------------------------------

Topology Summary



 ©2004 Hewlett-Packard Development Company, L.P.

In order to display the aggregate interface associations in `ovet_topodump.ovpl`, the Object ID of the interfaces are required to be shown. You can match the Object IDs to determine which ports participate in the same aggregation.

`ovet_topodump.ovpl -nodeif -detail` shows these Object IDs.

```
++++++++++++++++Node++++++++++++++++++++
NodeName:tshp51.cnd.hp.com
IPProtocolSupported:IPv4
ObjID:
OADId:0
SysOID:1.2.3.4.4.6
SysContact:
SysLocation:
Description:
Status:Normal
Capability:isLanSwicth isRouter
-------------------------ManagementAddress---------------------
AddressType:IPv4
ObjID
Address:
PingState:
---------------------------ManagementAddress-----------------------------------
++++++++++++++++Interface++++++++++++++++++++
IFName:
ObjID:123
```


Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

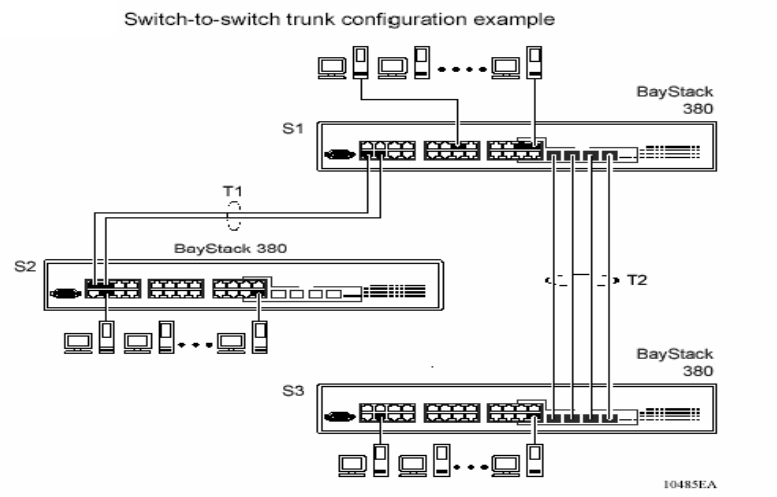
```
IFAlias:
IFDescription:
Status:Normal
IfIndex:
IfType:
AggregatedInterfaceObjID:
Capability:
+++++++++++++++Address++++++++++++++++++++
AddressType:IPv4
ObjID
Address:
PingState:
---------------------------Address-----------------------------------
--------------------------Interface----------------------------------
+++++++++++++++Interface++++++++++++++++++++
IFName:
ObjID:124
IFAlias:
IFDescription:
Status:Normal
IfIndex:
IfType:
AggregatedInterfaceObjID:123
Capability:isAggregatedIF
--------------------------Interface----------------------------------
--------------------------Node---------------------------------
```


Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Nortel MultiLink Trunk Support

Nortel MultiLink Trunk (MLT)

Discover all layer 2 connections via MultiLink Trunk



U5089S ©2004 Hewlett-Packard Development Company, L.P. 15

Nortel switches use MultiLink Trunk (MLT), a point-to-point connection that aggregates multiple ports so that they logically act like a single port with the aggregated bandwidth. Grouping multiple ports into a logical link provides higher aggregate throughput on a switch-to-switch or switch-to-server application.

The MLT shows up in 802.1D forwarding table with a big port number, such as 4096, which cannot be found from MIB-2's ifTable. So all remote neighbors associated with MLTs will not have corresponding local neighbors.

For Nortel Passport switches, if there is remote neighbor found for a MLT, Extended Topology duplicates the same remote neighbor to each port member of this MLT. This makes a L2 connection between two MLTs discovered and the GUI shows it as a port aggregation link.

This feature is available for IPv4 only.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Switch Stack Device Features

- Varies with vendors: Cisco, Nortel, 3Com, and Procurve

- Only HP Procurve supported at this time
  - Stacks have a commander and one or more members
  - Procurve Stack members may or may not have IP address
  - Commander always has an IP address
  - Proprietory MIBs support for member access through the Commander
  - Members accessed from Commander using community strings of pre-defined formats.
    - <Commander's Comm.Str>@sw<MemberID>

©2004 Hewlett-Packard Development Company, L.P.

Each vendor who implements stacked switches uses proprietary technology and terminology.

At this time NNM Extended Topology supports only HP Procurve switches in a stacked configuration.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

NNM Stack Support

NNM Stack Support

- Discover and represent the stack members as separate nodes in the Extended Topology database
- Monitor (status poll) the stack members as separate nodes

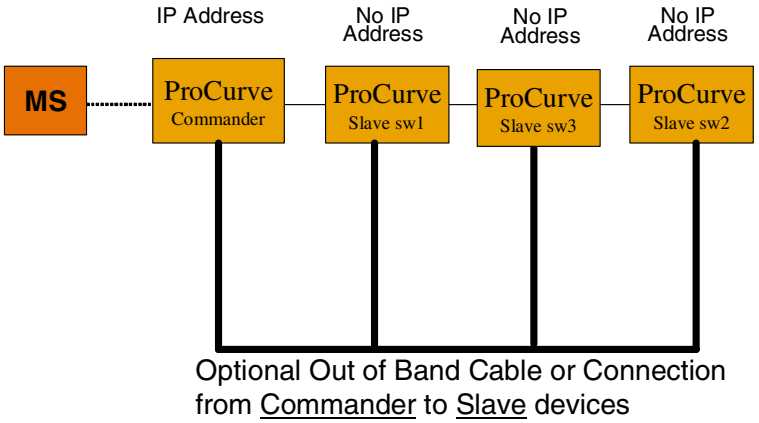
©2004 Hewlett-Packard Development Company, L.P.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

ProCurve Switch Stack Support

ProCurve Switch Stack Support

- Only members with NO IP address are discovered as part of the stack.
  - Not separate database objects
  - Interfaces contained directly in the Commander
  - Nodes connected to members appear connected to Commander
- Members with an IP address are discovered as independent switches.



U5089S ©2004 Hewlett-Packard Development Company, L.P. 18

The HP Procurve family of switches support virtual stacking of physically distributed switches in a single subnet, into a single logical stack. The virtual stack consists of a single commander switch and one or more participating member switches.

The commander switch is assigned an IP address and member switches may or may not have IP addresses assigned to them. The commander switch communicates to the member switch through layer-2 communication when they do not have IP addresses.

The following are the primary reasons for customers to use stacked switches:

- Reduces number of IP addresses needed for the network and enables adding switches to the network without having to do IP address management tasks
- Eliminates the need for any specialized cables for connections and removes the distance barrier between switches when using other stacking technologies
- Simplifies management of small work groups while scaling the network for bandwidth requirements.

The following are the series of switches that currently support stacking:

Series 2500 Switches, such as HP Procurve switch 2512, HP Procurve switch 2524

Series 4100 Switches such as HP Procurve switch 4108GL, 4104GL

Series 2400 Switches such as HP Procurve switch 2424, 2400

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Series 8000 Switches such as HP Procurve switch 8000M

Series 4000 Switches such as HP Procurve switch 4000M

Series 1600 Switches such as HP Procurve switch 1600M

For a more exhaustive and up to date information please refer to the HP Procurve Website, http://www.hp.com/go/Procurve.

As the members of a stack may or may not have an IP address assigned to them, only those members that do not have an IP address assigned to them would be discovered as part of the Commander Switch of the stack.

All switches in the stack appear as separate devices, correctly connected to each other and to end nodes. The Commander's switch number is by default assigned to be “0” by the stack.

The Commander switch of the Stack is represented by the same switch icon that is used to represent other non-stack switches.

Those members of the stack that *have* an IP address assigned to them would not be discovered as part of the stack but would be discovered as any other normal ProCurve switches. Stack members with an IP address are discovered as independent nodes through `netmon` and passed to Extended Topology discovery as independent nodes. In order to avoid duplication of discovery, these members with IP addresses are not rediscovered transparently by the Extended Topology device agent as stack members, but are discovered explicitly as independent nodes.

Stacked Switch Management

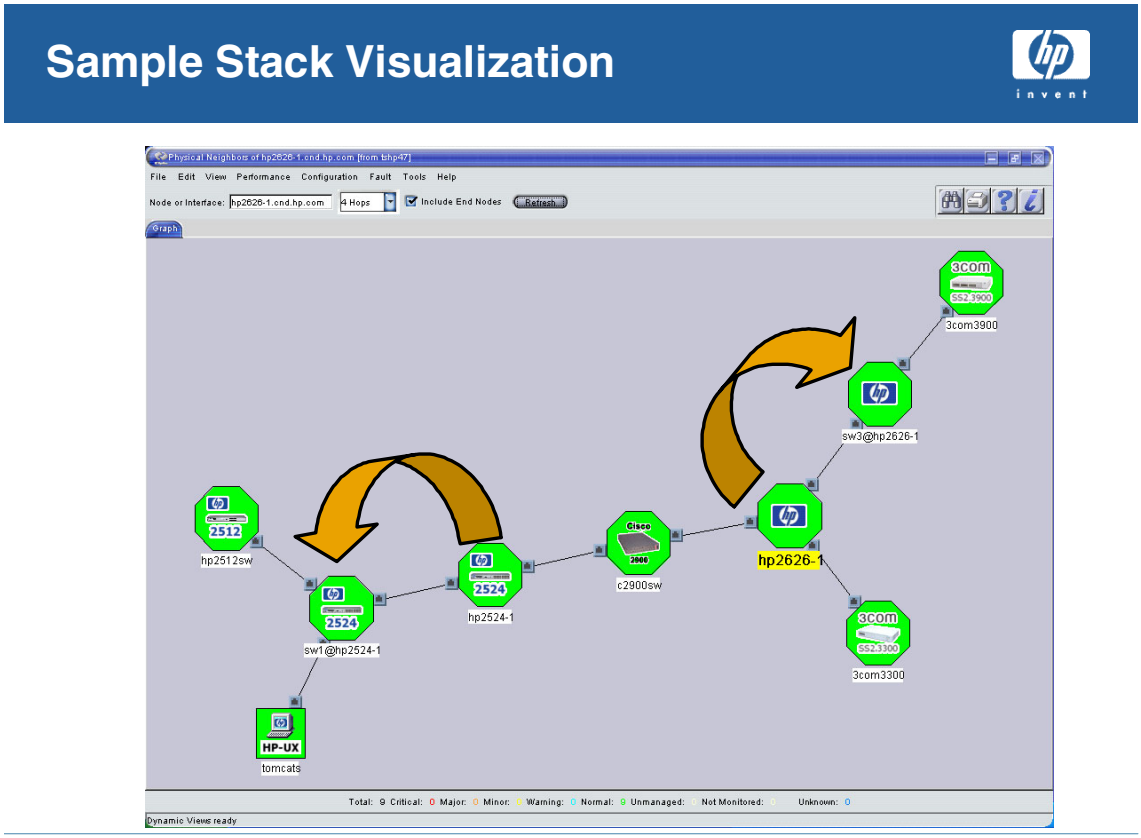
Stacked Switch Management

- Discovery creates an SNMP Proxy configuration entry for each slave device
  - identifies the commander’s node name as the proxy destination
  - the community string is a prefixed version of the commander’s community string
- Also updates SnmpNoLookupConf with the name of the slave device
- Since the SNMP configuration entries for the slave devices have the proxy set to the commander, the SNMP query actually goes to the commander.

©2004 Hewlett-Packard Development Company, L.P.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Sample Stack Visualization

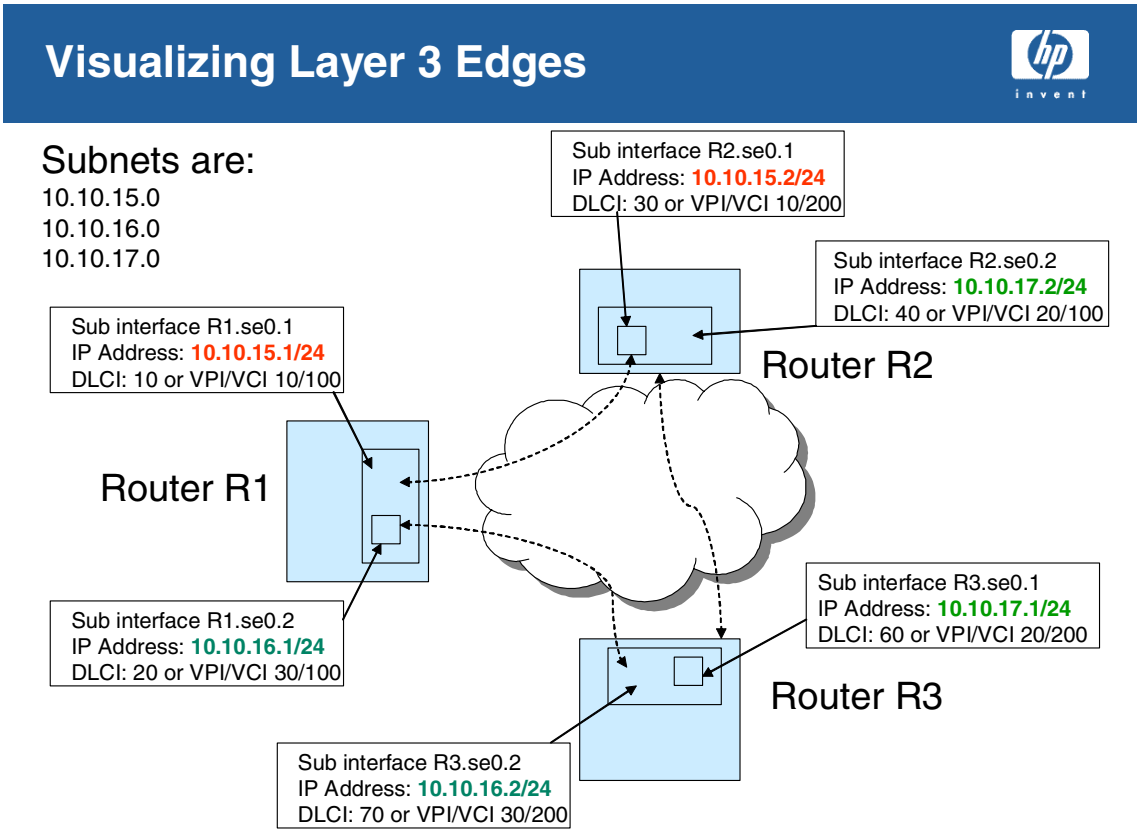


This display shot shows two different stacks. Each stack has one commander that is directly connected to one member which does not have an IP address. End nodes and other switches are connected to stack members. Other non-stack and non-Procurve switches areconnected to commanders.

Double clicking any stack member node displays the Node Details Page for that node, similar to the support for Node Details page for any other, non-stack switches.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Visualizing Layer 3 Edges



At the LAN/WAN boundary, it is common to have routers with ATM or Frame Relay interfaces that connect to a service provider VPN. These interfaces are often configured to be in an IP subnet with a 31, 30 or 29 bit subnet mask, with only two nodes existing in the subnet.

NNM AE 7.01 focused on layer-2 discovery.

Customers would like to see NNM AE connect these two interfaces together physically. This would allow for correlation of related events for the interface pairs and also for accurate neighbor and path view creation.

It could be argued that the layer-3 connection between two routers should not be modeled as a physical connection. From an architectural purity standpoint, this is probably correct. The benefits of connectivity analysis for polling and correlation outweigh this aspect.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

NNM 7.0 Handling of Layer 3 Connectivity

NNM 7.0 Handling of Layer 3 Connectivity

- Layer 2 centric (physical connectivity)
- Weak on Layer 3 connectivity.
- Typically occur in connections between routers (or other devices) with WAN interfaces
  - Routers with mainly ATM or FrameRelay interfaces, connecting to VPNs
  - T1/T3 Circuits and SONET etc.

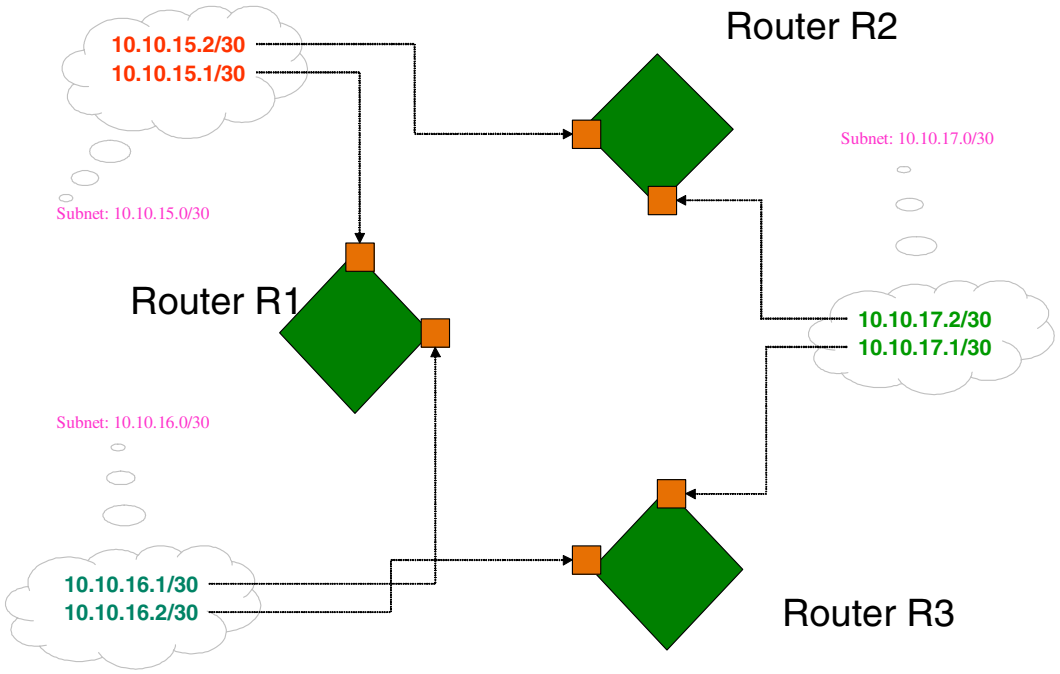
- For Layer 3 devices (e.g. routers)
  - Vendor proprietary protocols used for connectivity analysis
  - E.g. CDP for Cisco, EDP for Extreme.
  - No Vendor protocol means No connectivity!
  - Result in edge interfaces left unconnected … wrongly!
  - Inhibits effective Root Cause Analysis

©2004 Hewlett-Packard Development Company, L.P.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Visualization Without Connectivity Information

Current Topology Visualization (7.01) when Connectivity Info NOT Available



©2004 Hewlett-Packard Development Company, L.P.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

NNM 7.5 Addresses Layer 3 Edges

NNM 7.5 Addresses Layer 3 Edges

- Edge connectivity only
  - Layer 3 Core connectivity NOT addressed!
- Enhance connectivity accuracy
- Enable better Root Cause Analysis for polling and correlation.

- Also applicable to other generic WAN interfaces.
- 2-node Subnet - usually 30-bit subnet Mask
  - Utilize Subnet + Subnet Mask to derive Point-to-Point edge connectivity
  - Configurable Subnet Mask supported

- There are two scenario types:
  - Point to point
  - Point to multi-point
  - NNM 7.5 addresses ONLY Point to Point scenario!

U5089S ©2004 Hewlett-Packard Development Company, L.P. 24

NNM analyzes the IP interfaces on routers. When exactly two router interfaces exist in the same subnet, as identified by IP address and subnet mask, and both interfaces are not already connected to some other device via the existing layer-2 connectivity analysis, the interfaces will be connected.

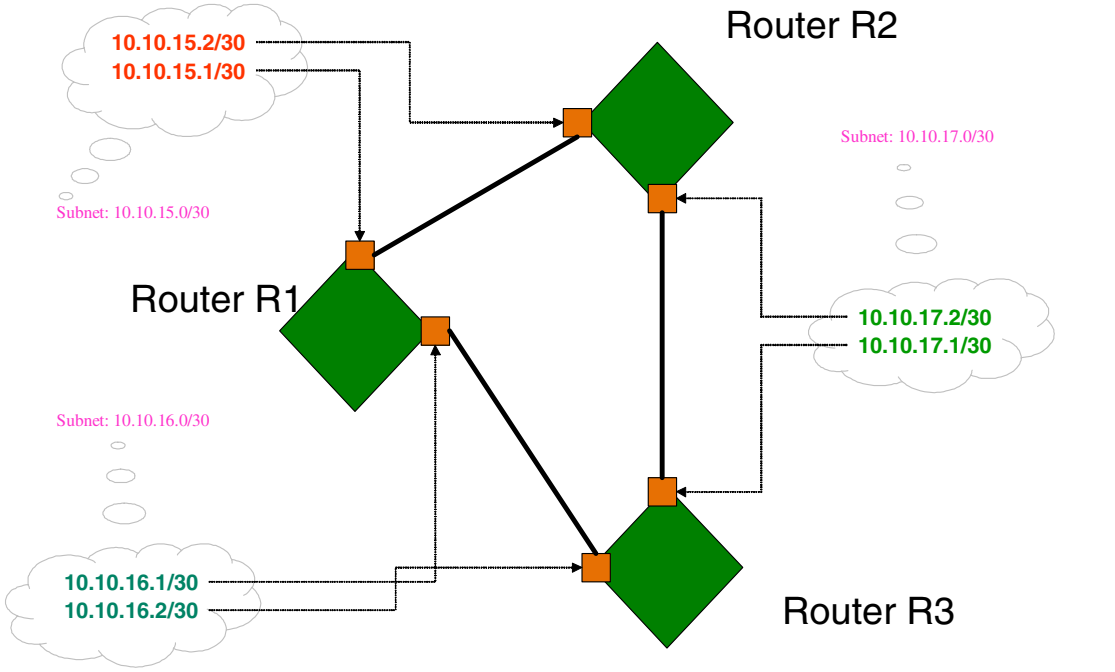
Connectivity information sources:

- Fdb Tables (Switches)
- CDP (Cisco + Procurve Devices)
- EDP (Extreme Devices)
- ILMI (ATM MIB supported devices)
- FDP (Foundry devices) – new!

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Visualization When Connectivity Info is Available

Accurate Topology Visualization when Connectivity Info is Available



©2004 Hewlett-Packard Development Company, L.P.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Configuring Layer 3 Edge Discovery

Configuration Details

•Provide out-of-the-box value with the most commonly used configuration setting as default – yet configurable!

•Configuration through $OV_CONF/nnmet/EdgeL3Conn.cfg

- **`enableConnectivity=1`**. 1 = enable (default), 0 = disable
- **`minAllowedBitmask=29`**. [0-31] (Default = 29)

©2004 Hewlett-Packard Development Company, L.P.

Layer 3 edge visualization is enabled by default. To disable it, create the following file.

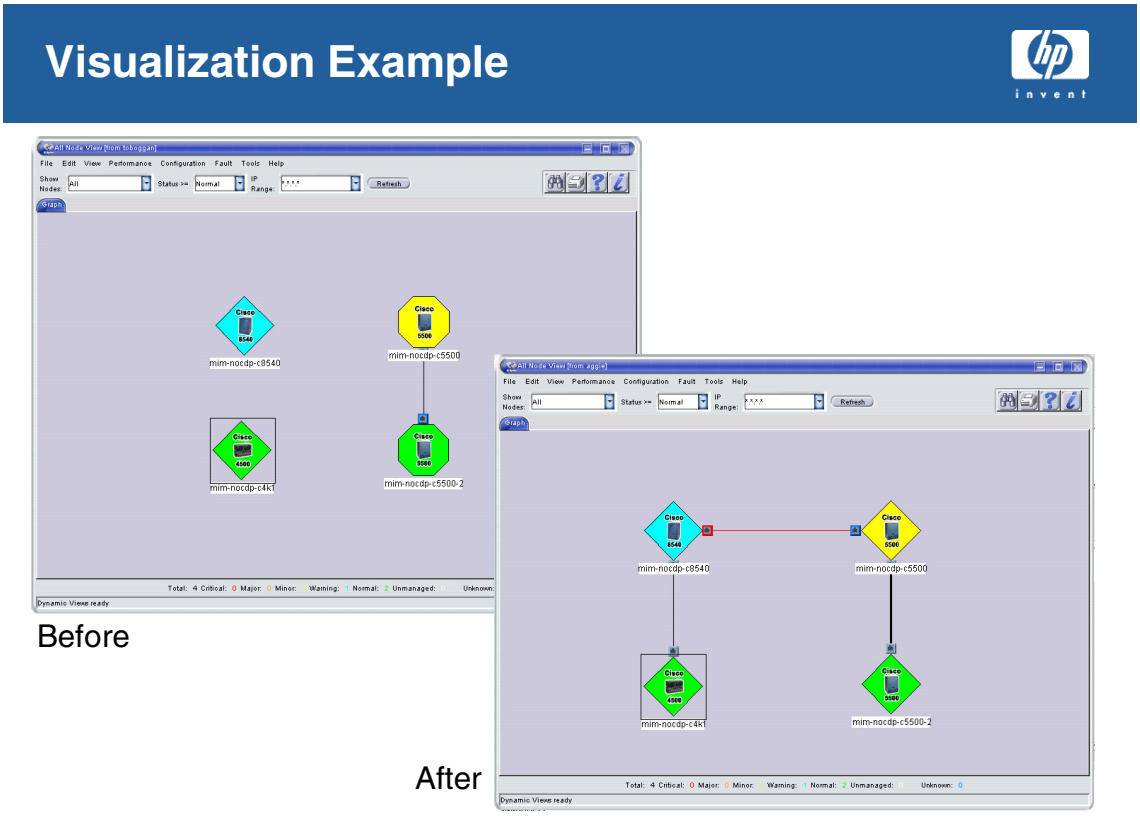
You can disable this capability or change the minimum allowable subnet bitmasks to identify participating routers. Edit `$OV_CONF/nnmet/EdgeL3Conn.cfg` and modify the `m_enableConnectivity` and `m_minAllowedBitmask` variables. The changes take effect as part of the next discovery (full or incremental) cycle.

- enableConnectivity. 1 = enable (default), 0 = disable
- minAllowedBitmask. [0-31] (Default = 29)

Note: Although configuration is carried through the configuration file, by default, the file does not exist.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Visualization Example



©2004 Hewlett-Packard Development Company, L.P.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Duplicate IP Address Support

NNM AE Extended Topology manages Overlapping Address Domains where two separate environments use overlapping IP address ranges, but which need to be monitored from a single manager.

Allowing netmon to Encounter Duplicate IP Addresses

In current networks `netmon` can expect to encounter the same IP address in use by multiple interfaces and/or systems. When this occurs, `netmon`

- prevents duplicate IP addresses from making it to `ovtopmd`'s topology
- prevents duplicate IP addresses from being picked as management addresses.

Whenever `netmon` finds that a newly discovered or added IP address is a duplicate, `netmon`:

1. drops the new as well as existing (old) interfaces which have the same IP Address.
2. if this address was earlier set as the management address, picks a new management address for that node/interface.
3. stores all the dropped IP addresses in the `netmon.noDiscover` file to prevent the same Anycast IP address being re-discovered. (`netmon` creates the file if it does not already exist.)

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

No external configuration is required.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Anycast Address Support

AnyCast Address Support

- An address that exists on multiple nodes, any one of which can respond to communications.
- A node sends a request to an anycast address and the “closest” node responds.
- IPv6 introduced the concept
  - In IPv6, anycast has its own reserved address space.
  - NNM 7.5 does not support IPv6 anycast
- IPv4 does not have a reserved anycast address space. An anycast address:
  - exists on multiple routers which broadcast the anycast address into the routing protocols.
  - is usually a software loopback address (other than 127.0.0.1).
  - usually has a network mask of 255.255.255.255.
- Monitored via SNMP query on the interface (not ICMP)

U5089S ©2004 Hewlett-Packard Development Company, L.P. 29

IPv6 introduced the concept of an “anycast” address; an address that exists on multiple nodes, any one of which can respond to communications with that IPv6 address. In IPv6, this concept is directly in the addressing architecture, with its own reserved address space. The theory is that a node can simply send a request to an anycast address and the “closest” node will respond.

In IPv4, this concept is used by convention, and is gaining popularity within the multicast community in particular as a method of locating rendezvous points. Since IPv4 does not have a reserved anycast address space, in practical terms you can not look at an IPv4 address and distinguish it from a regular unicast address. However, for an anycast address to work in the IPv4 space, it generally has to follow these conventions:

- The anycast address will exist on multiple nodes, usually routers since the node must be able to broadcast the anycast address into the routing protocols.
- The anycast address will be a software loopback address (other than 127.0.0.1).
- The anycast address will have a network mask of 255.255.255.255.

Thus, multiple routers will all advertise the same IP address into the routing protocols. The routing protocols take care of routing packets to the “closest” router serving the address based on the routing metrics.

In NNM 7.5, anycast addresses are only discovered and monitored in IPv4 environments. Since anycast address information is not available via SNMP MIB queries, discovery determines if

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

incoming addresses are qualified as anycast addresses using the following heuristic:

For any two or more interfaces with identical IPv4 addresses with the following characteristics:

- The interface is a software loop-back interface with an address other than 127.0.0.1.
- The matching address is in the same OAD
- The address has a subnet mask of 255.255.255.255
- Does not meet the characteristics for IP addresses as a backup/primary pair.

then categorize each address in the group as ANYCAST.

By default, anycast addresses are not monitored via ping, as it is generally impossible to tell if any given anycast node's address is responding to a particular ping. You can configure APA monitoring of AnycastAddresses in `paConfig.xml`. By default, ICMP polling is disabled and these addresses are only monitored via their interface via SNMP.

If an anycast address is monitored, then all IP address objects should generally reflect the same state. NNM concludes that all interfaces are in trouble if no node is responding to the anycast address. The interfaces containing that address are monitored by SNMP and become Minor if one of their addresses is down. If you disable SNMP polling as well, the interface becomes Critical because the failure of the address propagates directly to the interface with no way to verify it.

Anycast address should NOT generate any duplicate address warnings.

If an anycast address is enabled (by changing `paConfig.xml`) for ICMP polling, it will be treated like other IP addresses, with ping state set based on whether any response is received to the ICMP request, with no distinction of which node responded.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Backup Address Support

- Two interfaces have the same IP address.
- Must be on different devices
- One interface is administratively down (backup)
- IPv4 only

 ©2004 Hewlett-Packard Development Company, L.P.

It is common to have an interface configured as a “backup” to another interface, with only one interface enabled.

In this scenario, one of the interfaces will be in an administrative down state. An administratively down interface typically takes user action to bring to an active state, though sometimes this can be an automatic action when an attempt is made to communicate with the address.

You must configure `netmon` to ignore such addresses via the `netmon.noDiscover` configuration file.

Extended Topology can discover and manage duplicate IP addresses configured on different routers (in the same subnet), where one of the interfaces is administratively down. NNM discovers both the interfaces, and if possible creates a relationship of primary-backup between them.

NOTE NNM does not support for detecting duplicate IP addresses on the same node.

NOTE This feature covers IPv4 only.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Discovering Backup Addresses

Since backup/primary address information is not available via SNMP MIB queries, discovery determines whether incoming addresses are backup duplicate IPv4 addresses using the following heuristic:

For any two and only two interfaces with identical IPv4 addresses in the same subnet, in the same OAD,

- Both interfaces must be operational (accessible for SNMP) for the following administrative determination. Otherwise, a primary-backup relationship cannot be determined.
- If one of the interfaces is administratively up and the other interface is administratively down
  - — The address that is administratively up is qualified as the primary of a backup duplicate IP address pair
  - — The address that is administratively down is qualified as the backup of a backup duplicate IP address pair.

Backup interfaces do not generate extraneous warnings about duplicate IP addresses.

Monitoring Backup Addresses

Extended Topology allows duplicate IP addresses in the same OAD, and sets the “isDuplicated” attribute when detected. Additionally, “isBackup” indicates a backup interface. Filtering is allowed on either or both attributes.

When the interface becomes active, it will generate an Interface Up event. This is not correlated with the primary interface failure event in NNM 7.5.

The IP Address of the primary/backup interfaces is monitored based on the settings. Assuming the IP address is monitored, the ping state will be updated based on whether the IP address is responding. This may mean that an IP address on an administratively or operationally down interface may be marked as responding if the secondary or backup link is activated.

NOTE: In NNM 7.5, we are assuming that the two interfaces have the same IP address. There are other backup/failover scenarios where interfaces are activated to provide alternate routes without having the same IP address. This would be more typical with a dialup ISDN link as a backup to a higher bandwidth WAN link.

For backup interfaces that do not have a duplicate IP address, it is recommended that the configuration be adjusted such that pingEnable is set to false for those interfaces and/or IP addresses, so that pinging the address does not inadvertently activate the address.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Other Duplicate IP Address Support

Duplicate External Addresses

Duplicate external addresses exist when a service provider has multiple external connections to customers that have overlapping addresses. Technically, this is the realm of the Overlapping Address Domain support. However, the distinction here is that the routers primarily exist in the service provider's domain, and it is only the external interfaces that have overlapping addresses. The service provider may not be monitoring beyond the edge routers. In this case, the interfaces are all active, but will not be ping-able or routable.

ICMP polling is disabled.

Duplicate WAN Link Addresses

Under older WAN architectures, some customers configured their serial WAN links to all have the same IP addresses. For all intents and purposes, these can be treated the same as the external address case: they routable via ICMP, they are administratively up, etc.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Misconfigured Duplicate Addresses

Finally, there is the use case where there are two nodes that unintentionally have the same IP address configured. This is typically an end-node issue, but is not specifically restricted by type of node. Under this scenario, multiple nodes have operationally up interfaces that have the same IP address configured, AND that IP address is expected to respond to communication via ping or other methods.

NOTE: End-nodes with a single IP address are difficult to distinguish as both attempt to respond to the same address. These scenarios are typically handled by the nodes themselves reporting error conditions.

Requirements for Misconfigured Duplicate Addresses:

- Multiple nodes may have the same IP address in the same management domain.
- netmon will warn the user when multiple active monitored interfaces are detected.

How Extended Topology Handles Other Duplicate Address Scenarios

For the other duplicate IP address scenarios, the behavior is very similar to the duplicate backup IP address scenario.

For any known unreachable addresses, you should create an Extended Topology filter that describes these addresses. A matching class, “UnreachableAddresses”, can be created in the `paConfig.xml`, and the “pingEnable” setting should be “false”.

For other duplicate addresses, `netmon` will continue to generate warnings about the duplicates unless the address is put in the `netmon.noDiscover` file.

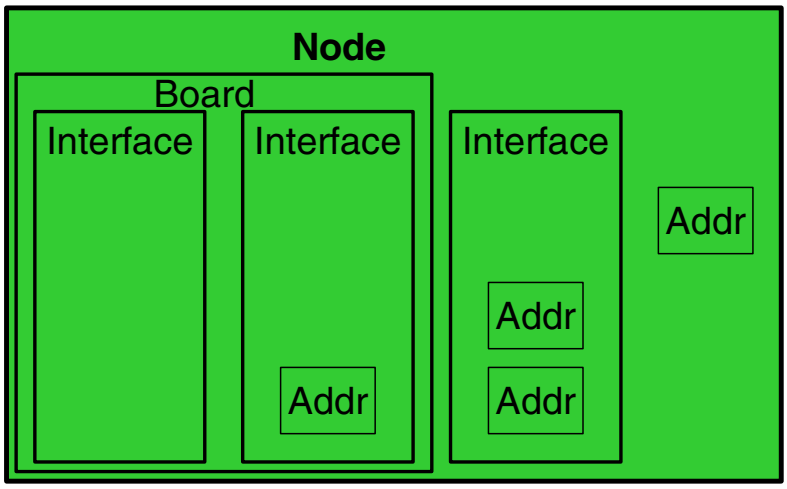
NOTE: When APA is your primary poller, `netmon` does not generate these warnings.

If polling is enabled on an address that exists on multiple nodes, the IP address state will be maintained based on whether ANY node is responding to ping on that IP address.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Simple Extended Topology Object Model

Simple Extended Topology Model



- Node may or may not have interfaces.
- Interfaces may or may not contain addresses.
- Interfaces and addresses may or may not be polled.
- Interface may be a virtual aggregated interface.

 ©2004 Hewlett-Packard Development Company, L.P.

Extended Topology models multiple levels of containment. This allows it to capture the variety of configurations that are possible on devices.

The smallest unit is the address. An address may be associated with an interface, a board, an aggregated (virtual) interface, or the node itself. Address health comes from a ping.

An interface may contain zero, one, or multiple addresses. The interface may be contained on a board or by the node. The health of an interface is determined by an SNMP query to the node using the MIB II variable IFOperStatus in the IFTable.

A board may contain zero or more interfaces and/or addresses. The board is contained by the node.

Interface and address status are tracked separately. It is possible for an address on an interface not to respond to a ping due to a routing error. However, when you query the node using SNMP (through a different management address) the operational status of the interface is up. Similarly, you could be able to ping an address, but the SNMP query to the management address could fail.

The Polling Engine has a large quantity of mixed requests in its queue -- pings for addresses, SNMP queries for interfaces, SNMP queries for boards, SNMP queries for aggregated ports, varied by system type, IFAdminStatus, connectedness, and current configuration entries. These queries happen in a random order. The Status Analyzer must be prepared to handle failures which arrive to it in any order.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

For any failure, the Status Analyzer drives the search for root cause up through the containers to the largest element which fails health verification. For example, if an address does not answer a ping, the Status Analyzer verifies the health of the interface. (These verifications do not pass through the Polling Engine.) If the interface is IFOperUp, the board and/or aggregated port health is verified, if there is one. If they are healthy, the node health is verified by an SNMP query to its management address followed by a ping to the management address.

Note that the Status Analyzer may have already confirmed that a container is down from a previous poll and the drive upward can leap forward to the current step in that previously initiated analysis.

As the search drives upward, an element fails the health verification. That is determined to be the **root cause** of the failure, and an alarm is generated for that element only. At that point, all **child components** are deemed to be unreachable. Their status in the Extended Topology database is set to Unknown (appears as dark blue), and a correlated event is generated for each that the component is unreachable. The **parent containers** receive a propagated status (silently) in the Extended Topology database of Minor and appear as yellow. An interface may propagate its status to multiple parents, for example a board and an aggregated port.

If it is determined that the entire node is down, APA begins Neighbor Analysis to determine whether this node's failure is the root cause or another node is at fault.

Note that this drive upward differs significantly from `netmon`'s approach of querying each interface and providing a stepwise node degradation. In APA's case, once the root cause is determined, the status is immediately set to Critical.

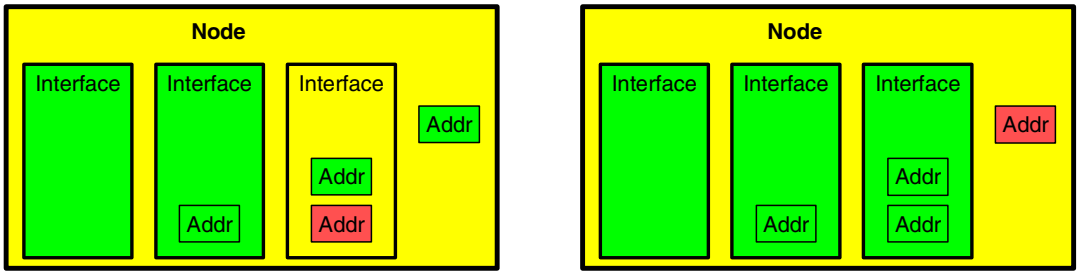
When status is forwarded to `ovtopmd`, only the root cause failure is forwarded (addresses are forwarded as interfaces). Secondary failures are not forwarded. The `ovw` display system propagates the status from the failed component to its containers according to the `ovw` rules.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

1: Address Down, Interface Up, Node Up

1: Address Down

- Polling Engine in APA observes ping failure.
- Sends to Status Analyzer in APA
- Determines interface (or node) is not down
- Ping_NotResponding status in Extended Topology database
- OV_APA_Address_Down alarm
- Pairwise utilized when address comes back up



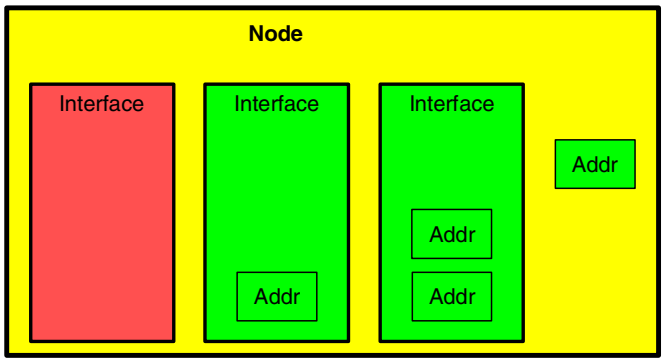
©2004 Hewlett-Packard Development Company, L.P.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

2: Interface Down, Node Up

2: Interface with No Address Down

- Polling Engine has nothing to ping.
- Polling Engine observes ifOperStatus is Down
- Sends to Status Analyzer
- Determines node is not down
- Critical status in Extended Topology database
- OV_APA_Interface_Down alarm
- Pairwise utilized when interface comes back up



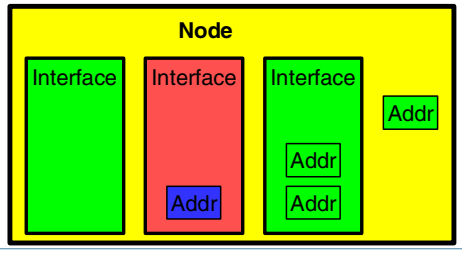
©2004 Hewlett-Packard Development Company, L.P.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

3: Address Down, Interface Down, Node Up

3: Interface Down

- Polling Engine observes ping failure and ifOperStatus is Down
- Sends to Status Analyzer
- Determines node is not down
- Critical interface status in Extended Topology database
- OV_APA_Interface_Down alarm
- Ping_Unreachable status for address
- Address unreachable alarm correlated with interface down alarm
- Pairwise utilized when interface comes back up
- Pairwise not needed for secondary failure (unreachable) events.

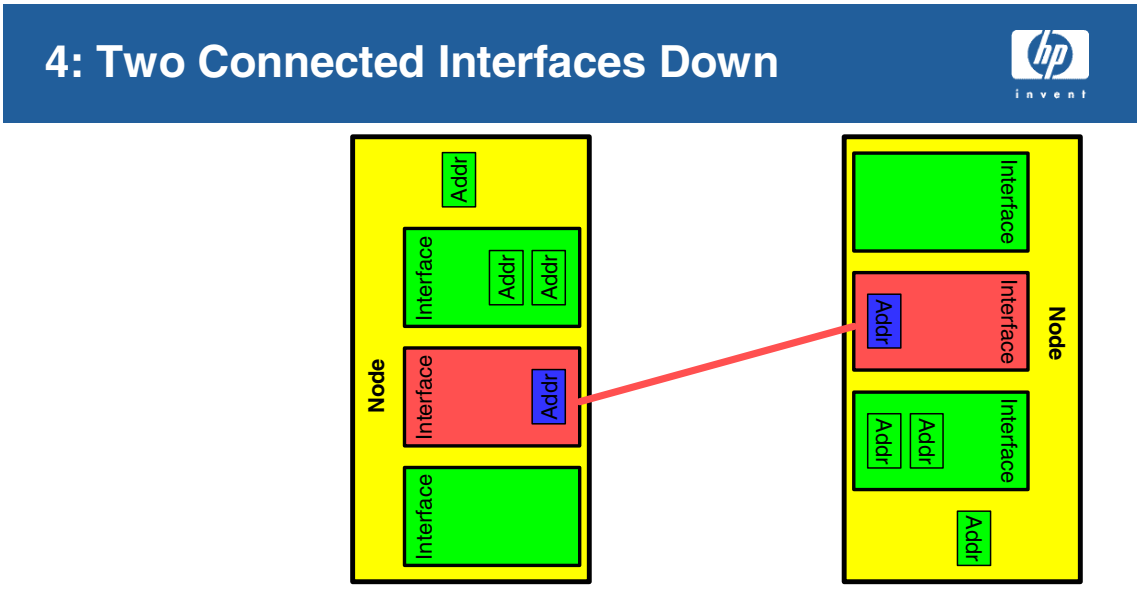


U5089S ©2004 Hewlett-Packard Development Company, L.P. 35

When you read the Alarm Browser, the Down alarm always indicates the root cause. Unreachable events all indicate secondary failures.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

4: Two Connected Interfaces Down



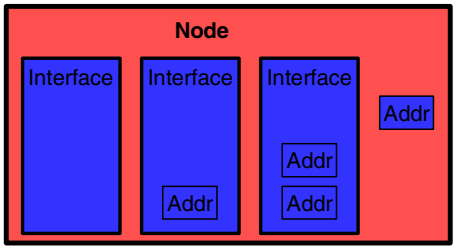
Since APA can still contact the node, no Neighbor Analysis is triggered.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

5: Node Down

5: Node Down

- Polling Engine observes ping failure or no SNMP response
- Sends to Status Analyzer
- Determines node is down
  - SNMP then ping to management address
- Critical node status in database
- Interface status is Unknown
- Address status is ping_unreachable
- Begin Neighbor Analysis
  - No alarms generated until Neighbor Analysis completes



U5089S ©2004 Hewlett-Packard Development Company, L.P. 37

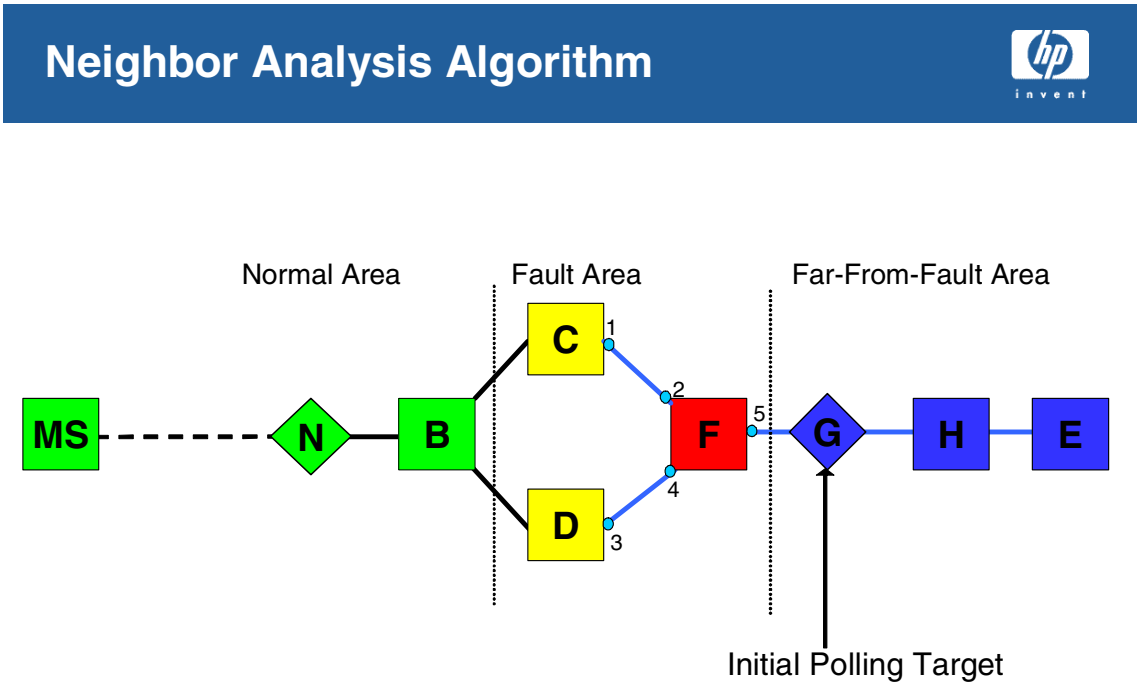
Once APA determines that the entire node is down, it begins a Neighbor Analysis to determine if this node is unreachable due to an upstream failure.

If the SNMP query fails but the follow-on ping succeeds, an SNMP_Not_Responding alarm is issued. So, for example, if you have an incorrect community string (after a correct discovery somehow), the node turns red, but the alarm in the Alarm Browser is SNMP_Not_Responding, not Node_Down.

It is possible that the management address is unreachable due to a routing error, but other interfaces are in fact up. You can get a false positive on Node_Down. It is important to set your management address to the one most reachable from the management station, not one further downstream.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Neighbor Analysis Algorithm

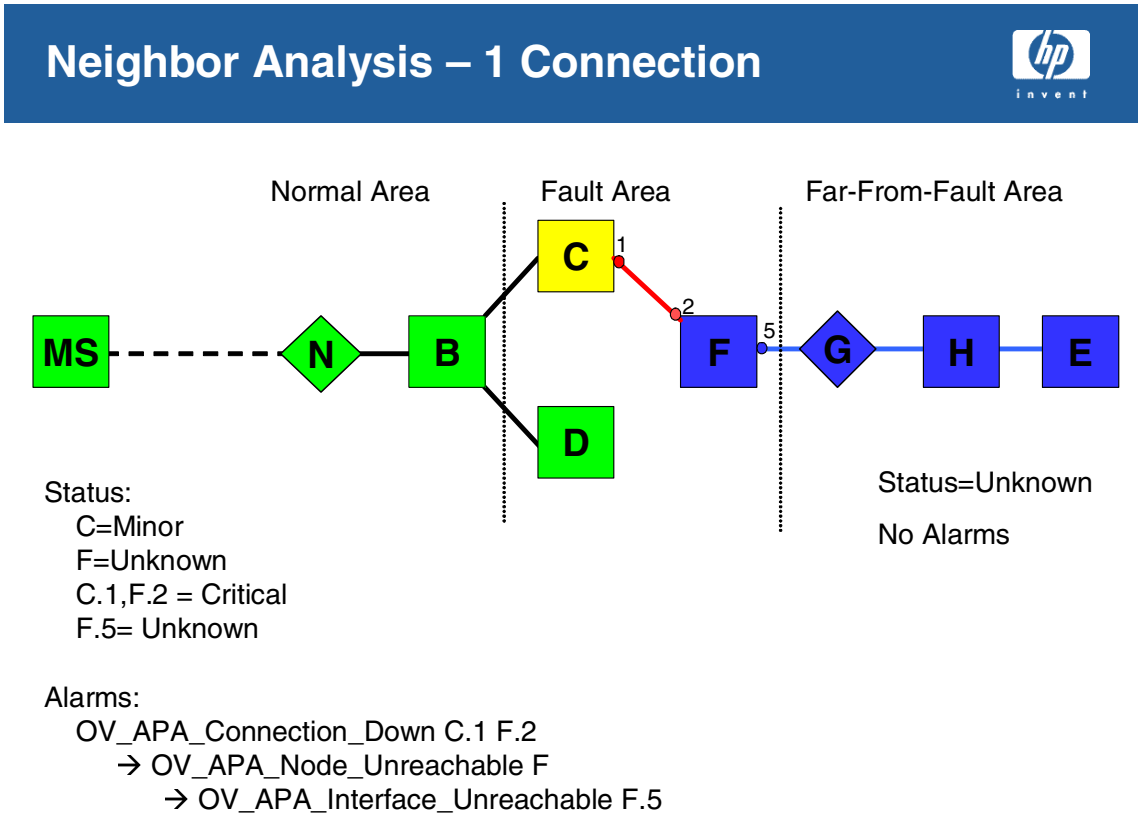


Neighbor Analysis begins by determining the area in which the fault occurs. APA examines the Extended Topology database to determine all the nodes connected to this node. For each of those, the node health is verified. If all of the neighbors are Node_Down, APA is looking beyond the fault area, surrounded by secondary failures. APA examines the neighbors of the node nearest the management station, working backward until at least one node is up.

The Fault Area is defined as the area where at least one node is Node_Up and one node is Node_Down.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Neighbor Analysis: One Connection



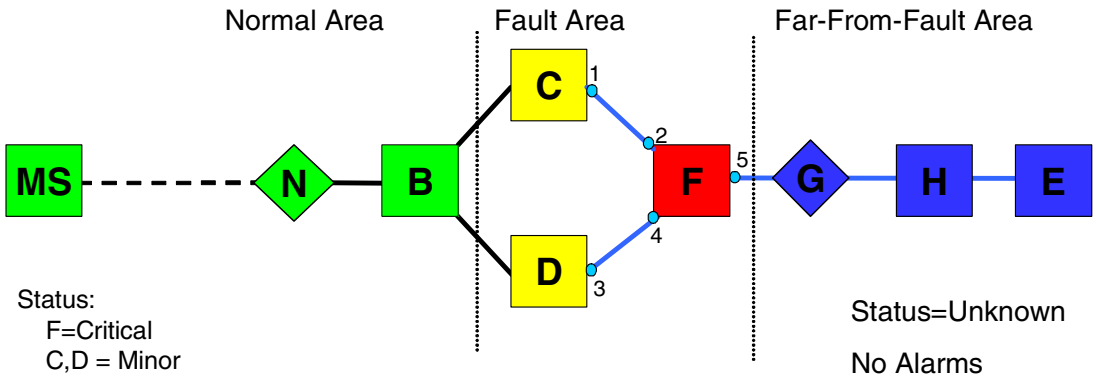
When the nearest node which is Node_Down has only one neighbor, APA queries that neighbor to determine the IFOperStatus of the interface connected to the down node. If the interface is down, APA marks the connection as the root cause, and sets the downstream node to Unknown.

is it possible for the IFOperStatus to be up? Is this really the check that's done? Or does APA just assume its the connection that's down?

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Neighbor Analysis: Two Connections

Neighbor Analysis – Multiple Connections



Status:
- F=Critical
- C,D = Minor
- F.2, F.4, F.5 = Unknown
- C.1, D.3 = Unknown

Alarms:
- OV_APA_Node_Down F
  - → OV_APA_Connection_Unreachable C.1 F.2
  - → OV_APA_Connection_Unreachable D.3 F.4
  - → OV_APA_Interface_Unreachable F.5
    - → OV_APA_Address_Unreachable

Status=Unknown

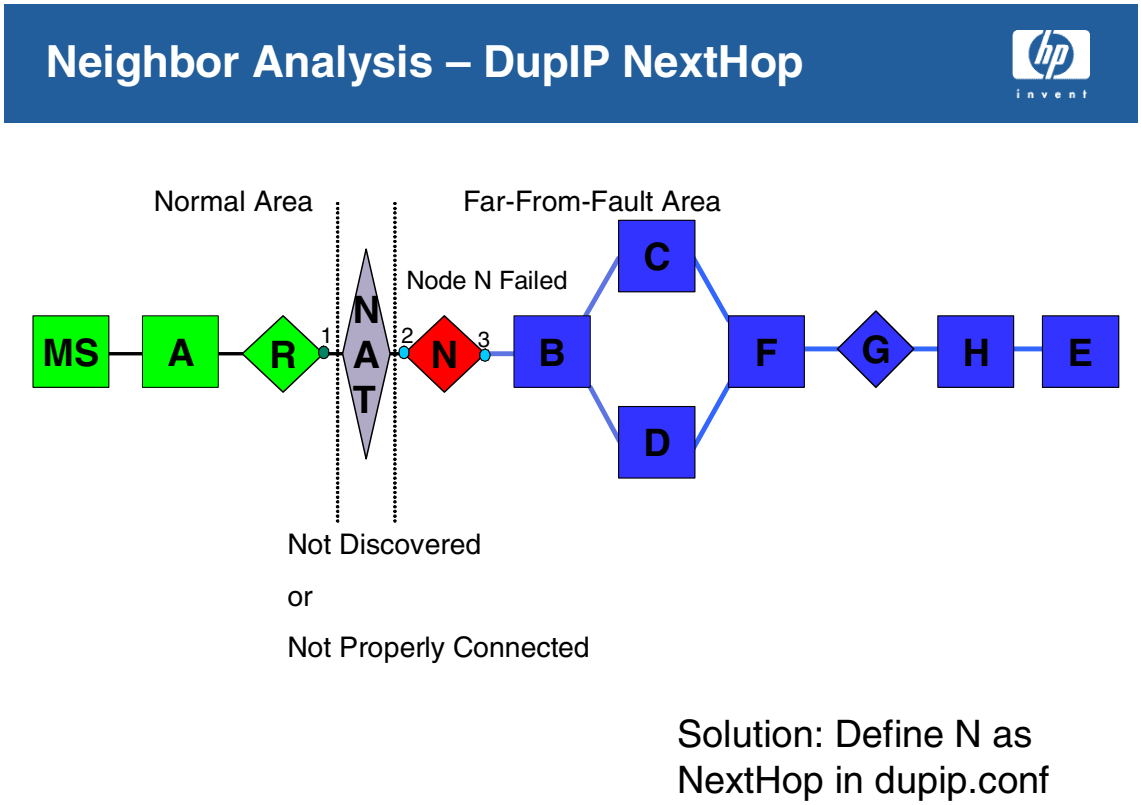
No Alarms

©2004 Hewlett-Packard Development Company, L.P.

When a down node has more than one neighbor, APA queries to see whether either neighbor has a working connection to the node in question. If neither neighbor can access the node, the node is marked as the root cause and the connections become secondary failures.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Neighbor Analysis: OAD NextHop



OAD environments present a special concern to NNM because the NAT device tries to be invisible. In this case, if the NAT device fails or the inbound interface on the router fails, it has no neighbors to query. The inbound interface on the router is seen as unconnected, and therefore unpolled. Because the management address cannot respond the node could be marked as the root cause. To avoid this, designate the first router(s) in the domain with the NextHop keyword in the `dupip.conf` file.

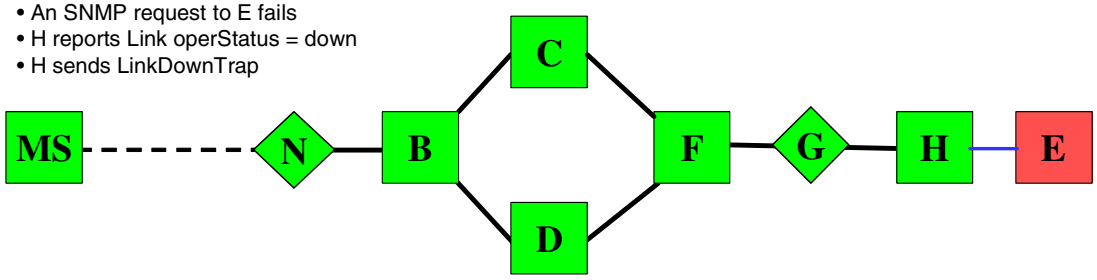
Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Neighbor Analysis: End Node Down

Neighbor Analysis – End Node Down

- End Nodes and Access ports are handled differently.
- Behavior is Configurable.

Symptoms:
- Ping E Fails
- An SNMP request to E fails
- H reports Link operStatus = down
- H sends LinkDownTrap



OV_APA_Node_Down E
→ OV_APA_Connection_Unreachable H-E

U5089S ©2004 Hewlett-Packard Development Company, L.P. 42

Turning off desktop systems at night or having servers fail should not create unnecessary troubleshooting for switch operators. Rather than have the Neighbor Analysis return a Connection_Down (because the node in question has only one neighbor), APA contains special code for end nodes that returns a Node_Down.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

Lab Exercises

Cisco Board Discovery

Assumptions:

- `$OV_CONTRIB/NNM7labs/initialsetup.ovpl` has been executed.

Directions

1. Change working directory to `$OV_CONTRIB/NNM7labs/Lab_deploy`:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_deploy
   ```

2. When the instructor has given you the go-ahead, execute the setup script:

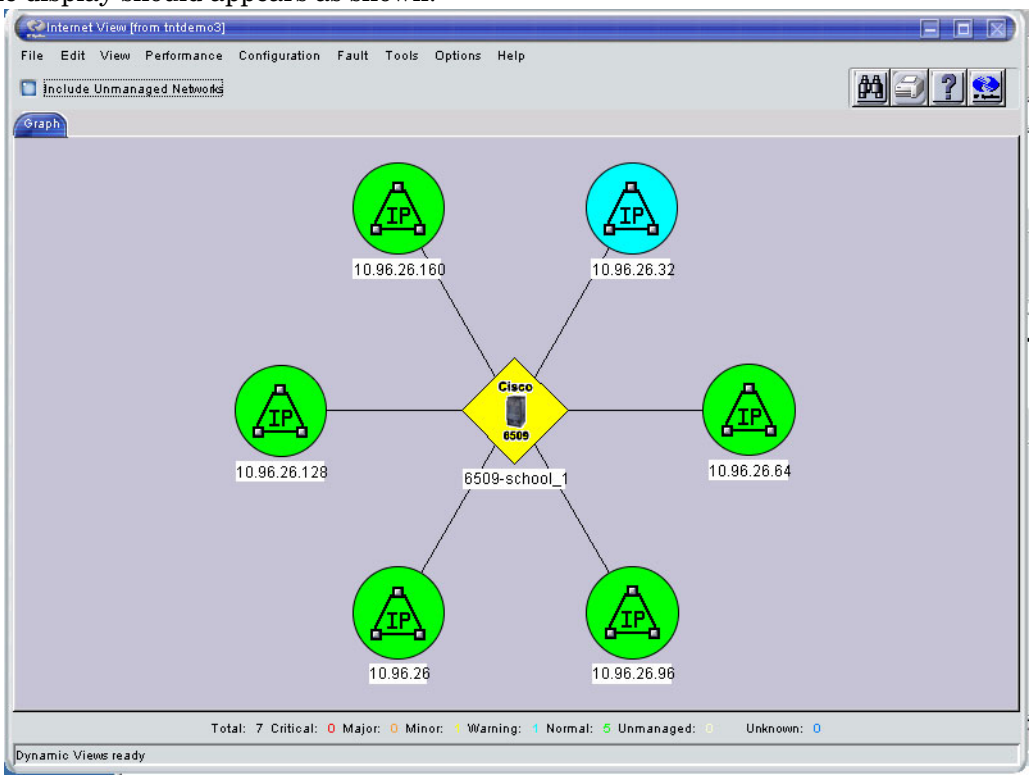
   ```
   setupLab_deploy.ovpl
   ```

3. When this completes, open an Internet View from Home Base.
4. Verify proper and complete discovery of the simulated network. Locate any symbol that may

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

be unmanaged or unknown. If unmanaged, select it, and then use the `ovw` menu `Edit:Manage`, or the dynamic view menu `File:Topology:Manage`. For any node which shows as a blank square, select the node and `Fault:Network Connectivity:Poll Node`.

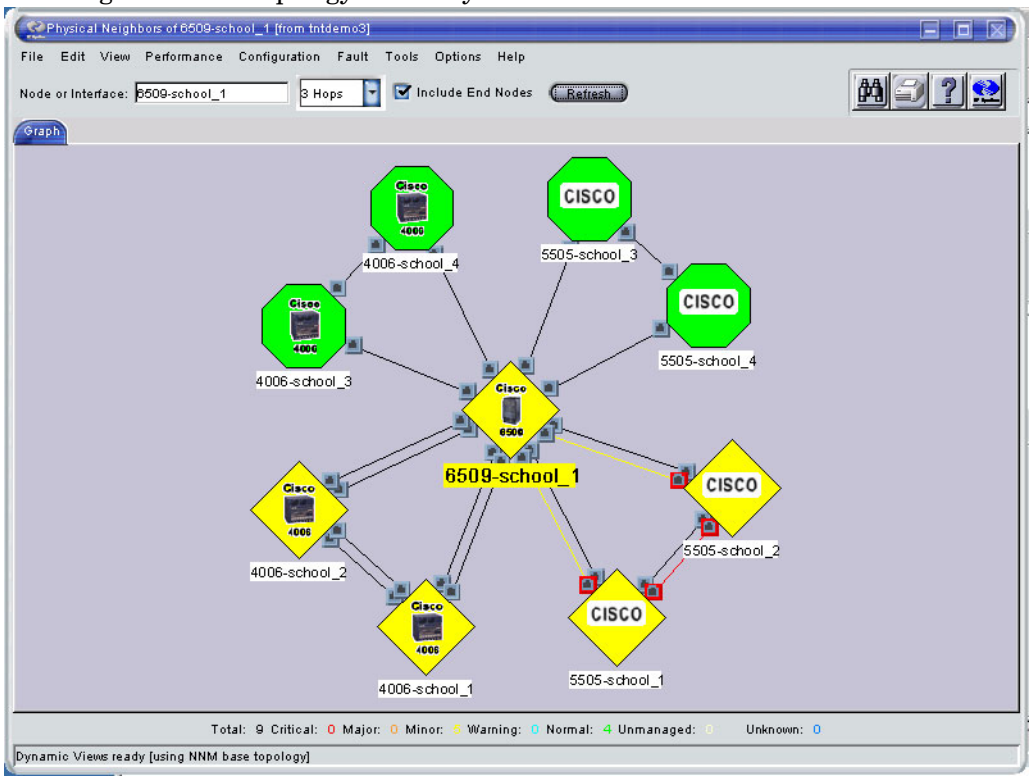
5. The display should appears as shown.



6. Execute a Neighbor view of the device 6509-school_1 with 3 hops and include end nodes, prior

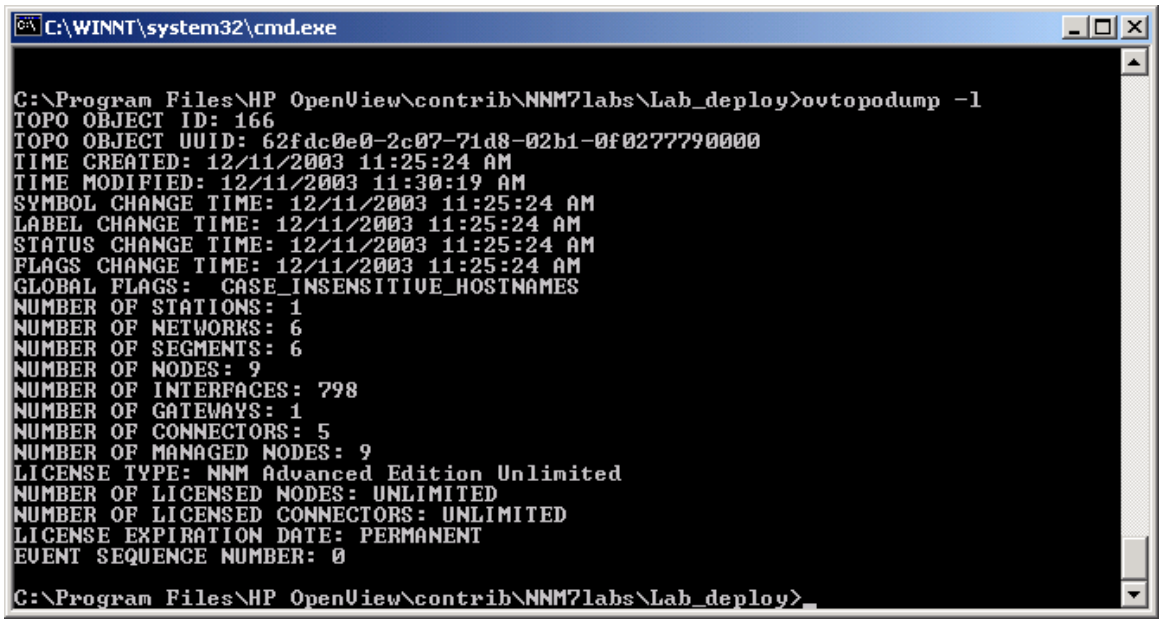
Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

to executing Extended Topology discovery.



7. Confirm correct topology has been discovered:

```
ovtopodump -l
```



8. When you have confirmed that all devices are managed and appear as shown in the above figures, commence the Extended Topology discovery from Home Base.

NOTE You will run several iterations of Extended Topology discovery throughout these labs. Each execution of the discovery will give you an opportunity to use the support tools to view progress of the discovery processes and agents. These

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

tools will be especially important in the labs that simulate discovery problems.

9. From the NNM Home Base, bring up a Node View. Select “All” for the Show Nodes field and “Normal” for the Status >= field, then hit the Refresh button. Note the status of the nodes and links in the Node View.
10. Right click on the 6509-school_1 node, and select Details.
11. In the Node Details window, click on the Boards link in the top frame.
12. Note the board detail information in this section of the Node Details window. Note the status of the board with interface ID 205.
13. Close the Node Details window.
14. When the instructor gives you the go-ahead, right click on the 6509-school_1 node. Select “APA Status Poll” from the resulting pop-up menu. Wait for the APA status poll to complete in the resulting APA Demand Poll window.
15. Repeat step 9to bring up a Node View of your entire network. Note the new status of the nodes and links in the Node View.
16. Right click on the 6509-school_1 node, and select Details.
17. In the Node Details window, click on the Boards link in the top frame.

Use of this training material is restricted in accordance with "TERMS OF USE AND LEGAL RESTRICTIONS" at the beginning of this document.

H IPv6 in Extended Topology

Module Objectives

Configuring Extended Topology Discovery of IPv6

Version C.00
U5089S Appendix H Slides

After completing this module, you will be able to:

- Describe and configure IPv6 addressing.
- Configure Extended Topology IPv6 discovery.
- Explain interactions between IPv4 and IPv6 in Extended Topology.
- Display and interpret IPv6 views.
- List requirements for IPv6 in Extended Topology and on managed devices.
- Configure IPv6 polling.

IPv6 Background

- Used in wireless environments
- Used when shortage of IPv4 addresses
- MIBs currently supported by Hitachi, NEC, and Juniper
  - Cisco working on implementing RFC 2465

©2004 Hewlett-Packard Development Company, L.P.

The 4 billion available IPv4 addresses are being consumed at a very rapid rate. Projections are that all IPv4 address space will be exhausted between 2005 – 2011. Class A and B addresses are rarely available, and some parts of the world (notably Japan) are being seriously impacted by the shortage.

Running out of Internet addresses forces users to pursue various “patching” strategies in order to keep using IPv4.

IPv4 routing tables are experiencing explosive growth. Variable-length headers and fragmentation en route further reduce router performance.

Security is not sufficient for current business needs.

IPv6, short for “Internet Protocol Version 6", is a set of Internet Protocol specifications designed by Internet Engineering Task Force (IETF).

IPv6 is the next step in the evolution of the dominant IPv4 protocol used throughout internet and corporate networks today.

IPv6 is not backward compatible with IPv4, but IPv6 will co-exist with IPv4 on IP networks as IPv6 usage grows.

IPv6 adoption is expected to occur gradually, with early adoption by customers who need its advanced features or who need to utilize its expanded address space.

IPv6 Features

Comparison of IPv6 to IPv4

Feature	IPv6	IPv4 + “Patches”
Address Space	Almost unlimited (128 bits)	4 billion (32 bits)
Mobile IP	Embedded Scales to global mobility	Can serve a limited number of terminals
Security	Standard method Deployable for global internet access VPN embedded	Several alternatives, all have scale limitations due to limited address space
Automatic configuration of network	Integral to IPv6 stds. Auto addressing of end-nodes Easier for customer to move to another ISP	No comprehensive standard solution Manually intensive
Performance	Hierarchical routing = smaller internet routing tables Fixed header=fast routing	Not designed for current, widespread use Flow control limitations

©2004 Hewlett-Packard Development Company, L.P.

IPv6 Address Types

3 Types of IPv6 Address

- Unicast address is an identifier for a single interface.
  - Extended Topology manages IPv6 devices based on their unicast address(es).
  - The rest of this course discusses only IPv6 unicast addresses.
- An anycast address is an identifier for a set of interfaces, usually on different nodes.
  - An IPv6 packet sent to an anycast address is delivered to one of the interfaces identified by that address.
- A multicast address is an identifier for a set of interfaces, usually on different nodes.
  - An IPv6 packet sent to a multicast address is delivered to all interfaces identified by that address.
- A single, physical interface can have multiple addresses.

U5089S ©2004 Hewlett-Packard Development Company, L.P. 4

An anycast packet might, for example, be sent out to a timeserver. The “nearest” known timeserver will receive the packet and respond with the current time.

A multicast packet might, for example, be sent out to all interfaces in a particular subnet to notify them of an important event that will soon affect them.

IPv6 Addresses

IPv6 Addresses

•IPv4 uses 32-bit addresses, with each 8-bit part converted from binary to decimal.

- For example, the IPv4 address:

  00001010 10001110 10000000 01000010

  is represented as 10.142.128.66

•IPv6 uses 128-bit addresses, divided into 8 16-bit parts. Each 16-bit part is represented as 4 hexadecimal numbers.

- The IPv6 address:
- 0011 1111 1111 1110 1000 0000 1111 0000
- 0000 0000 0000 0010 0000 0000 0000 0000
- 0000 0000 0000 0000 0000 0000 0001 0000
- 0000 0000 0000 0000 0000 0000 0000 0001
- is represented as 3FFE:80F0:0002:0000:0000:0010:0000:0001

©2004 Hewlett-Packard Development Company, L.P.

Abbreviating IPv6 Addresses

Abbreviating IPv6 Addresses

- This address can be further abbreviated, by eliminating leading zeroes, to 3FFE:80F0:2:0:0:10:0:1
- A double colon (“::”) indicates that the values that it replaces are all zeros.
- :: can be used (only once per address) to further abbreviate this address to 3FFE:80F0:2::10:0:1

©2004 Hewlett-Packard Development Company, L.P.

Note that 3FFE:80F0:2::10::1 is NOT a valid IPv6 address, since the :: cannot be used more than once.

IPv6 Address Scoping

IPv6 Address Scoping

- Link-local scoped addresses
  - Used only between neighboring nodes attached to the same link
  - Prefix 1111 1110 10 in the first 10 bits
  - Almost all interfaces will have a link-local address.
- Site-local scoped devices
  - Used only within an isolated internet
  - Prefix 1111 1110 11 in the first 10 bits
- Global-scoped addresses
  - Used to reference any address, regardless of the network in which it resides.
  - Globally unique
  - Prefix 001 in the first 3 bits

©2004 Hewlett-Packard Development Company, L.P.

Note that global-scoped unicast addresses for IPv6 must always begin with 2 or 3.

Examples:

Link-local scoped addresses:

```
FE80::1
FEB9:8C5::9
```

Site-local scoped addresses:

```
FEC2:202D::27
FEFA::3
```

IPv6 Prefixes

IPv6 Prefixes

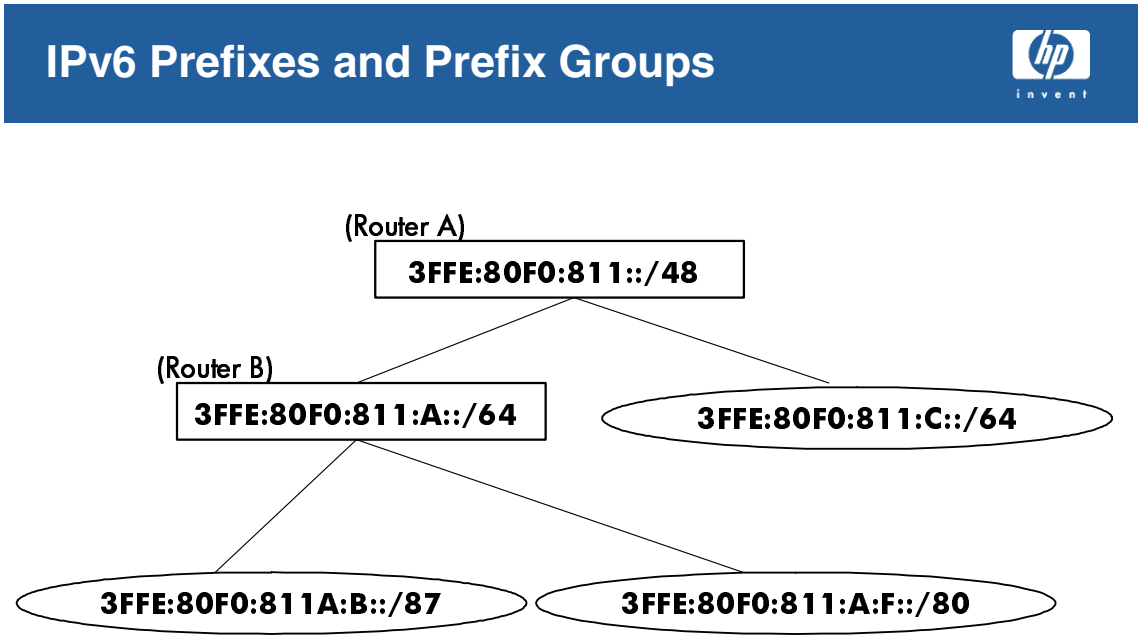
- IPv6 addresses are often grouped into a hierarchy of networks and subnets by the use of prefixes.
  - For example: 3FFE:80F0:2::/48 is a network, defined by the first 48 bits of the address.
  - All subnets and global-scoped addresses within this network begin with the same 48 bits.
- This network could be further subnetted, for example, into: 3FFE:80F0:2:AC::/64, 3FFE:80F0:2:1B15/64, 3FFE:80F0:2:C5F:A::/80, etc.
- Remember that the prefix is based on bits, not hex. If a prefix falls in the middle of a hex value, compute the prefix based on the bit value.

©2004 Hewlett-Packard Development Company, L.P.

An example of the bit versus hex prefix group computation: Consider the prefix 3FFE:80E0::/27. The address 3FFE:80F1:: would be included in this network, because the first 27 bits define the network. The first 27 bits for both the prefix and the address are:

```
0011 1111 1111 1110 1000 0000 111
```

IPv6 Prefix Example



Hierarchical prefixes make it easier and faster for routers to forward IPv6 messages.

Routers at each level “advertise” their address coverage UP the hierarchy to other routers. In this example, Router B would advertise its address coverage up to Router A. Router A would advertise its address coverage to other routers (not shown) that it is connected to.

Router A forwards the messages that have addresses in Router B’s range to Router B. Router A also handles message traffic to a group of nodes in the subnet: 3FFE:80F0:811:C::/64. IPv6 networks and subnets are often referred to as “prefix groups” in Extended Topology. There could be multiple prefix groups on a single, physical link.

Router B handles message traffic for two subnets (prefix groups), 3FFE:80F0:811A:B::/87 and 3FFE:80F0:811:A:F::/80.

IP Transition Strategies

IPv4 -> v6 Transition Strategies

•Some of the strategies designed to ease the transition from IPv4 to IPv6, are:

- "Dual stack": A router or host is configured to handle both IPv4 and IPv6. Several manufacturers now offer routers and hosts that support this.
- "Tunneling": Repackaging and forwarding an IPv6 packet over IPv4 link(s) to an IPv6 destination. Tunnels can be created manually or automatically. Dual-stack routers often support tunneling, converting v6 to v4, and (later) v4 back to v6.
- "IPv4-Compatible IPv6 addresses": Setting the first 96 bits of an IPv6 address to 0, and the last 32 bits to the IPv4 address. This works only between dual-stack nodes capable of automatic tunneling.

©2004 Hewlett-Packard Development Company, L.P.

There are two types of tunnels (v6 transported on v4):

- One is called "6over4", and requires the user to manually configure the tunnel.
- The other one is called "6to4", where the router automatically configures the tunnel, based on specific addresses.

IPv6 Resources

Additional IPv6 Resources

•IPv6 Books:

- Loshin, IPv6 Clearly Explained, Morgan Kaufmann Publishers, 1999, ISBN 0-12-455838-0
- Wegner, IP Addressing and Subnetting, Including IPv6, 1999, Syngress Publishing, ISBN 1-928994-01-6
- Miller, Implementing IPv6: Supporting the Next Generation Internet Protocols, 1999, Wiley, John & Sons, Incorporated, ISBN: 0-76-454589-2

•IPv6 RFCs:

- www.ietf.org/html.charters/ipv6-charter.html

•IPv6 Classes:

- Native6 (www.native6group.com; Seattle, Washington)

©2004 Hewlett-Packard Development Company, L.P.

Extended Topology and IPv6

Extended Topology and IPv6

- Basic network management for an IPv6 network

- Can monitor both IPv4 and IPv6 protocols from one management station
- Separate IPv6 views; not map dual stack devices in IPv4 views

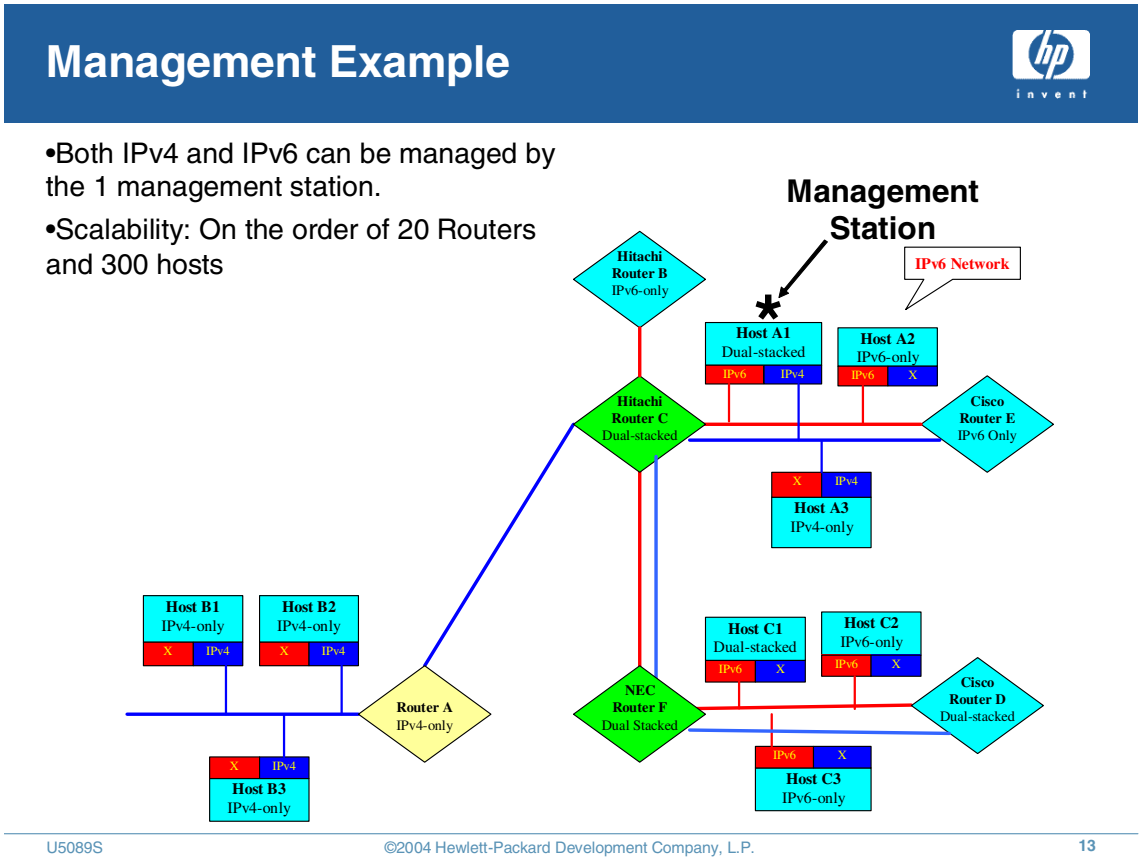
- Discovery of IPv6 devices
- Status monitoring via ICMPv6 “ping”

- Status change events integrated into NNM event subsystem
- Visualization of IPv6 layer 3 connectivity
- Visualization of IPv6 dynamic status changes

©2004 Hewlett-Packard Development Company, L.P.

You must purchase the Advanced Routing SPI to manage IPv6 networks.

IPv6 Management Example

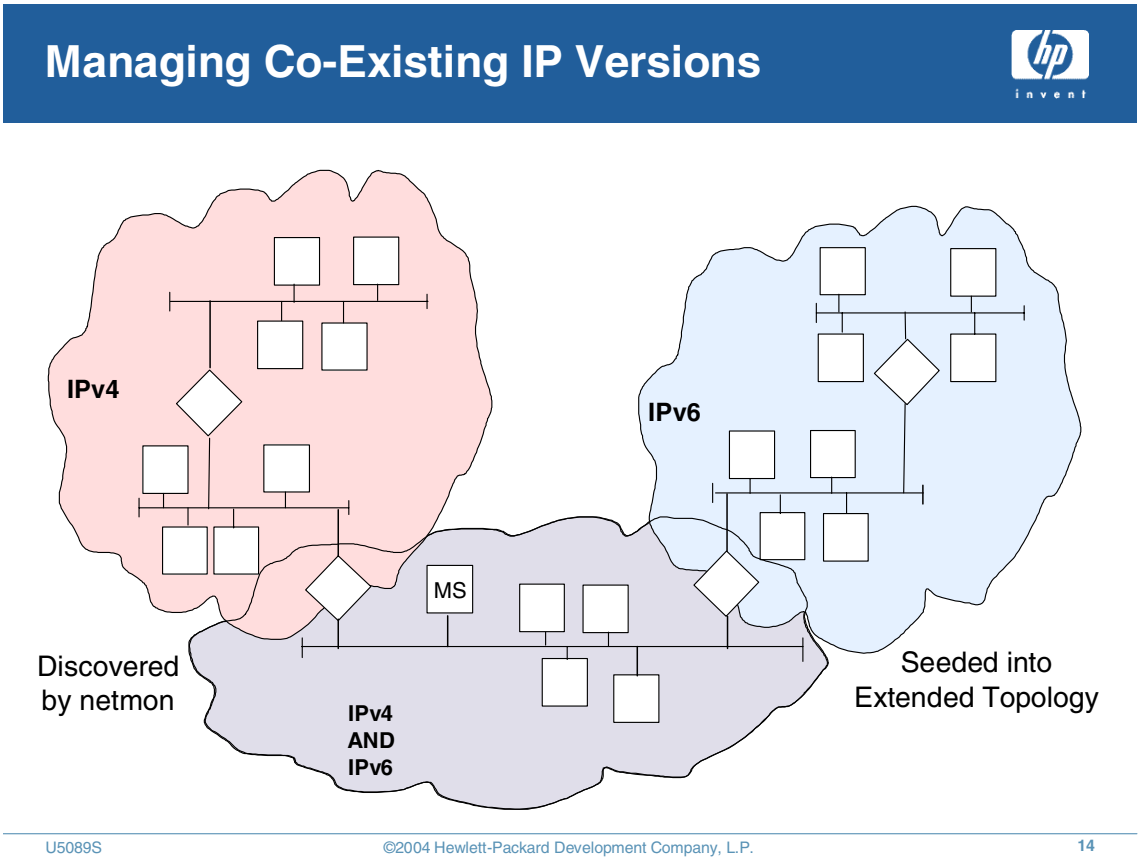


This is a simplified example of a real-world network with both IPv4 and IPv6 devices—NOT the way it would be shown in the Extended Topology GUI.

Router B is not fully managed because it is not running dual stack, even though it is a Hitachi router.

Routers D and E are not fully managed because they are Cisco routers and Cisco routers do not currently support the IPv6 General MIB. Routers D and E would each be labeled as an “Unknown Router” in Extended Topology.

Managing Coexistence with Extended Topology



IPv4 and IPv6 configuration in NNM and Extended Topology is independent.

To manage a node using IPv4 management, use the regular NNM configuration mechanisms (discovery filter, `netmon` seed file, etc.) and displays (`ovw`, dynamic views, network presenter).

To manage a node using IPv6 management, use IPv6-specific configuration files in Extended Topology (IPv6Seed.conf, etc.) and Extended Topology IPv6 dynamic views.

If a node is dual stacked, you are not required to configure it for both IPv4 and IPv6 management. You can configure it for:

- IPv4 management only
- IPv6 management only
- both IPv4 and IPv6 management
- neither

HP recommends that dual-stack nodes be configured for both types of management. To do this, configure both standard NNM configuration and Extended Topology IPv6.

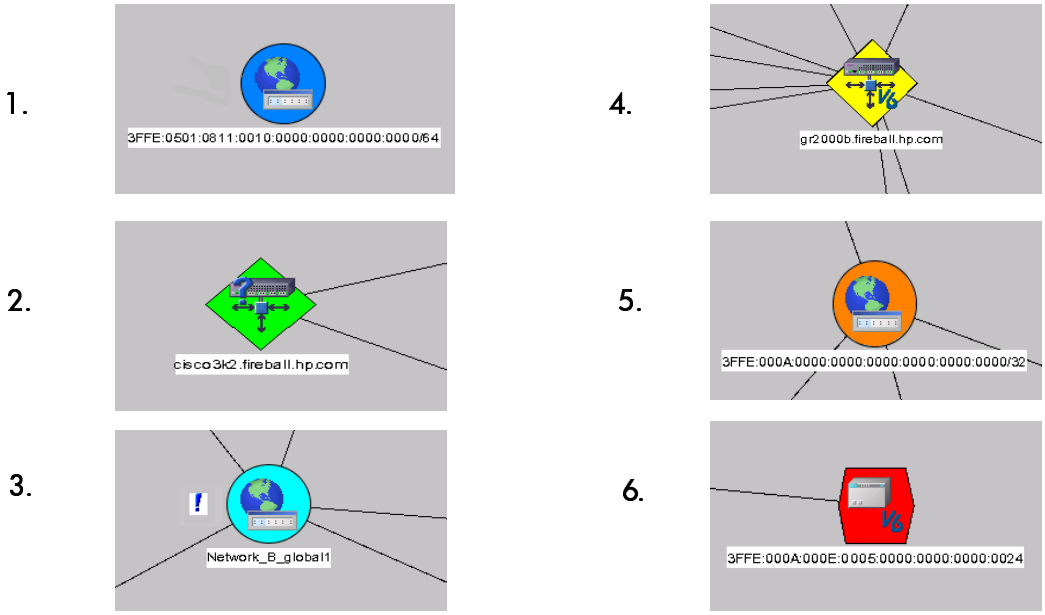
If you like to manage disjoint sets of IPv4 and IPv6 nodes or overlapping (partial or complete overlap) networks, you can configure NNM and Extended Topology to handle all these scenarios.

Status is independently computed (for IPv4 and IPv6). If a node is dual-stacked, the status shown

in the IPv6 views corresponds to IPv6 aspect of it only.

IPv6 Views

IPv6 Views: Symbols and Status Colors



 ©2004 Hewlett-Packard Development Company, L.P.

The symbols and status colors shown above illustrate the symbols and colors available in Extended Topology views. What can we tell about these devices?

1. The symbol indicates that this is a prefix group and its color indicates that its status is UNKNOWN. The (64-bit) numerical prefix is used as its label.

2. The symbol indicates that this is a router and its color indicates that its status is NORMAL. Its label was obtained from the seed file or from DNS. It is connected to 2 prefix groups. The "?" indicates that we can't be sure that this is an IPv6 router. (This router does not support the IPv6 General MIB, so the information shown may be incomplete.)

3. The symbol indicates that this is a prefix group and its color indicates that its status is WARNING. Its numerical prefix label has been replaced by a prefix name, which was added to the prefix name file. The "!" indicates that the status of this device has recently changed. It is connected to a total of 5 IPv6 devices (routers or end nodes).

4. The symbol indicates that this is an IPv6 router and its color indicates that its status is MINOR. Its label was obtained from the seed file or from DNS. It is connected to 9 prefix groups.

5. The symbol indicates that this is a prefix group and its color indicates that its status is MAJOR. Its (32-bit) numerical prefix is its label. It is connected to a total of 4 IPv6 devices (routers or end nodes).

6. The symbol indicates that this is an end node and its color indicates that its status is

CRITICAL. Its IPv6 address is its label.

Each interface in a router may have multiple addresses belonging to multiple prefix groups. For example:

Number of nodes	interfaces	addresses	prefix groups	lines shown
1	1	3	3	3
1	1	3	1	1

IPv6 Status Propagation

IPv6 Status Compounding Rules

- Possible Address Status Values
  - Normal = UP
  - Critical = DOWN (no Ping response)
  - Unknown (can't ping)
- Possible Interface Status Values (computed from Addresses)
  - Unknown (all Addresses have Unknown Status)
  - Critical (at least one Address is Down)
  - Normal (all addresses are Up or Unknown)
- Interface Status Values are compounded for Node Status.
- Address Status Values are compounded for Prefix-Group Status.

©2004 Hewlett-Packard Development Company, L.P.

Address Status -->	**Interface Status -->**	**Node Status**
Site-local Address Status	Site-local Interface Status	Site-local Node Status
Global Address Status	Global Interface Status	Global Node Status
Site-local + Global Address Status	Overall Interface Status	Overall Node Status

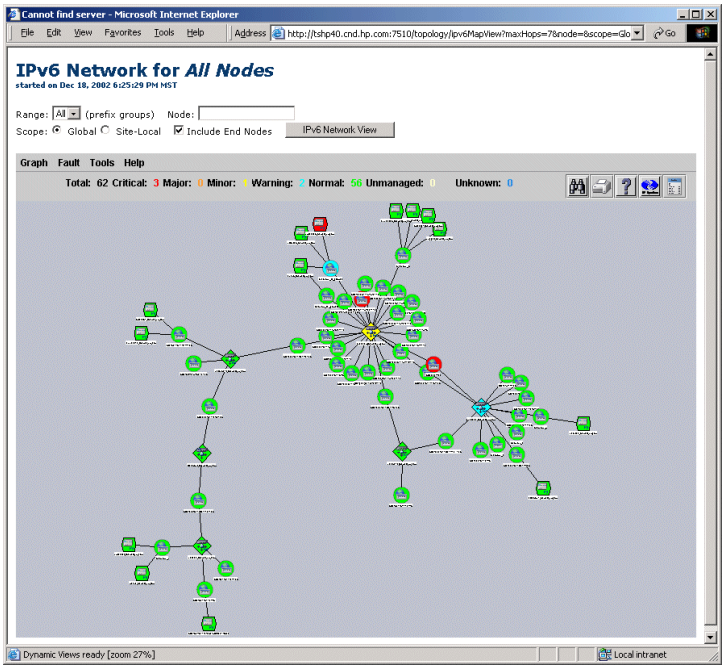
The status compounding rules shown apply to routers, end-nodes, and prefix groups. For interfaces (only), a single CRITICAL address status results in a CRITICAL overall status, regardless of the status of the other addresses.

Conditions on contributing objects	**Compound Status**
No objects exist	Unknown
All objects are Unknown	Unknown
All are Up (except ones that are Unknown)	Normal

Conditions on contributing objects	Compound Status
One is down and all others are up (except ones that are Unknown)	Warning
More than one is up and more than one is down (except the ones that are Unknown)	Minor/Marginal
One is up and all others are down (except the ones that are Unknown)	Major
All are down (except the ones that are Unknown)	Critical

IPv6 Network View

High-level IPv6 Network View



©2004 Hewlett-Packard Development Company, L.P.

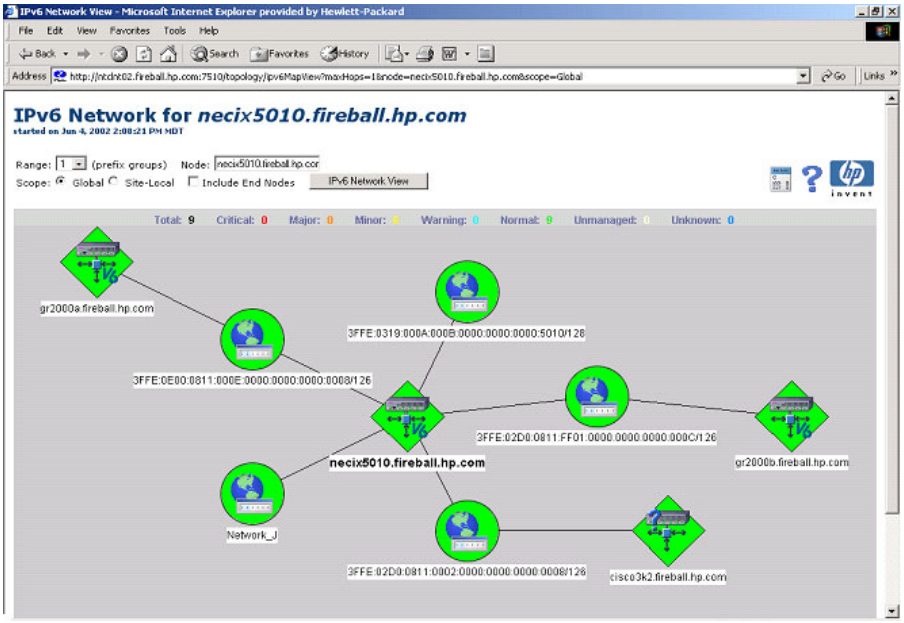
The IPv6 Network view shows you a graphical representation of an IPv6 network, including routers and end nodes that support IPv6. Lines connect IPv6 nodes to the prefix group or groups they logically connect to. A prefix group is similar to a subnet in IPv4. A node may belong to more than one prefix group. For example, a node could be configured with more than one IPv6 address, resulting in an IPv6 Network view showing multiple logical connection lines: one line for each IPv6 address to prefix group logical connection.

Status changes are dynamically displayed. New nodes are not displayed until the next discovery cycle and a manual refresh of the view.

Device names are shown (if available from DNS or from the seed file).

Focused IPv6 Network View

Focused IPv6 Network View



 ©2004 Hewlett-Packard Development Company, L.P.

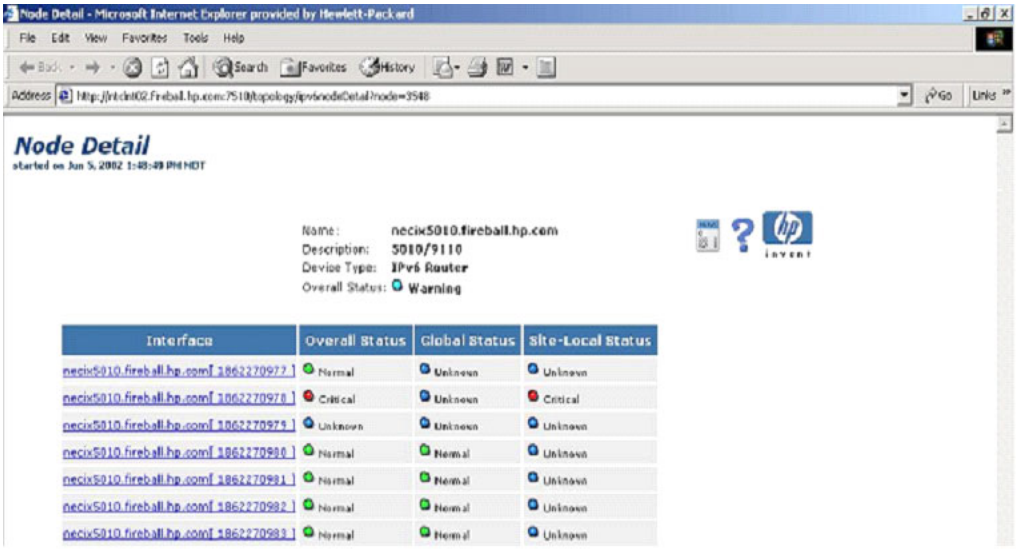
This sample view shows a more focused view of a portion of an IPv6 network. To see this view, we changed the Range to 1, because we wanted to focus on all routers and prefix groups that are closely connected to the router necix5010.fireball.hp.com. (A “Range” of 1 means “show me everything within 1 router of the specified node”.) We also had to enter the name of the node that we wanted to use as the starting point. Then we clicked on the IPv6 Network View button to see the modified view.

We can see that this router has IPv6 addresses belonging to 5 prefix groups. The checkbox for Include End Nodes was not selected, so no end nodes are shown.

To see a more detailed (tabular) view of the interface information for the router necix5010.fireball.hp.com, we can double-click on this router.

IPv6 Node View

IPv6 Tabular Node View



 ©2004 Hewlett-Packard Development Company, L.P.

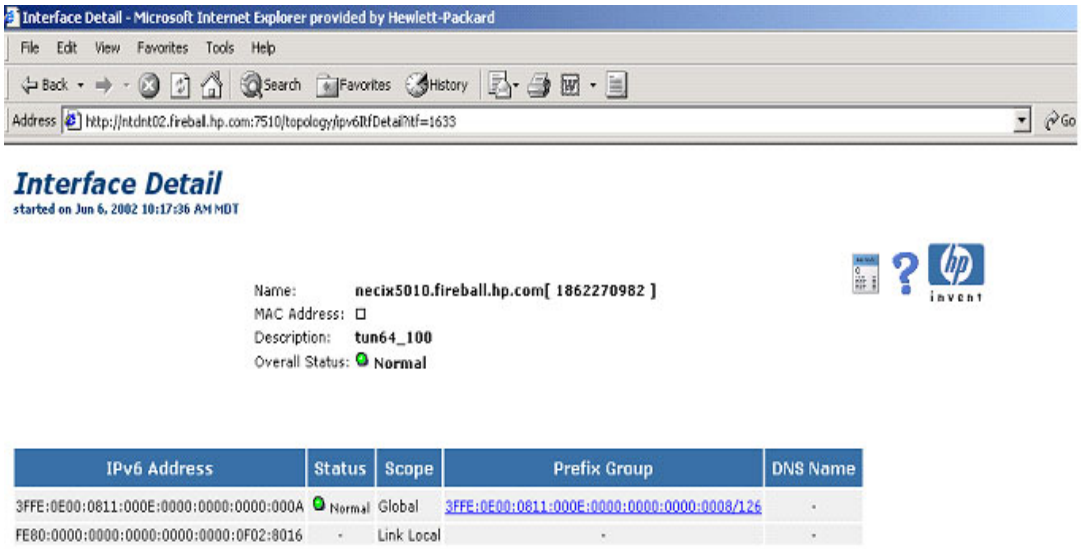
This is the Tabular View obtained by double-clicking on the router symbol for necix5010.fireball.hp.com. The table shows the list of interfaces for this node.

We can also see that the overall status for this node is WARNING, which is determined by compounding the status of the various interfaces. Both the Global status and the Site-local status values are used for compounding into the Overall status.

Tabular views are static. The “Status” column displays the status when the last ping was done. A manual refresh must be done to see new ping updates.

IPv6 Interface View

IPv6 Tabular Interface View



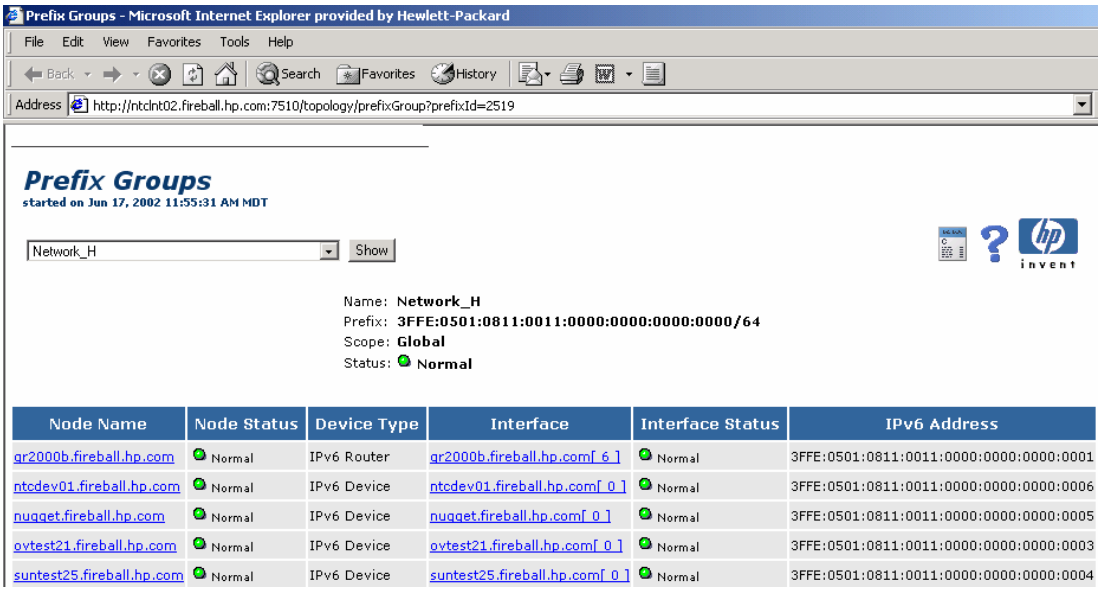
 ©2004 Hewlett-Packard Development Company, L.P.

To see this view, we clicked on one of the interfaces in the previous table (in the Interface column).

The table shows the list of addresses for this interface. This example has 2 addresses: 1 global scope and 1 link-local scope. The status of link-local scoped addresses are not monitored or displayed. The overall status for this interface is NORMAL because all addresses (other than link-local) have a status of NORMAL. If any address for an interface has a status of CRITICAL, then the overall status for the interface is shown as CRITICAL.

IPv6 Prefix Group View

IPv6 Tabular Prefix Group View



 ©2004 Hewlett-Packard Development Company, L.P.

To see this view, we clicked on one of the prefix group symbols on the GUI. The table shows each IPv6 address belonging to this prefix group. Each address in the prefix group, plus its node and interface, etc. are also shown.

Note that the Global address status for each of the nodes in this group is NORMAL—so the status for the prefix group is also NORMAL.

IPv6 System Requirements

IPv6 Extended Topology System Requirements

•The management station must

- Be running “dual stack” (IPv6 and IPv4), since some IPv4 capabilities are needed.
- Be checked to ensure that specific kernel parameters and patches are installed. (Please see the NNM Extended Topology release notes for further details.)

©2004 Hewlett-Packard Development Company, L.P.

IPv6 Router Requirements

Router Requirements

•This product utilizes the IPv6 General MIB (RFC 2465).

- Managed routers must support this MIB.
- Managed routers must also run dual stack for full management capabilities.

•Routers that do not support the General MIB and dual stack are only partially managed. Their connectivity information may not be shown accurately or completely.

•End-nodes may run either IPv6 alone or dual stack.

•End nodes need not support this MIB.

©2004 Hewlett-Packard Development Company, L.P.

Routers that currently meet these requirements are:

- Hitachi:

  GR2000, with software version 06-02 and later
- NEC:

  Bluefire 700 series with software version R03.06 or later

  Bluefire ix1000 _____

  Bluefire ix5000 series with software version 5.3.08 or later)

  Bluefire ix5210 series with software version 2.0 or later.

  Juniper: with kernel 6.2R1.40 or later

If a router is running single-stack (IPv6), instead of dual stack, Extended Topology will:

1. Display it in the GUI as an end node, rather than as a router
2. Show whatever connectivity information is available from other sources (seed file, DNS, or neighbors)
3. Show UP/DOWN (green/red) status obtained via ping
4. Status and events would be based upon whatever is discovered and may not be complete.

If a router is running dual stack, but does not support the General MIB, Extended Topology will:

1. Show whatever connectivity information is available from other sources (seed file, DNS, or neighbors)
2. Still be able to ping it, receive events from it, and show status color information (depending upon information available)
3. Display it in the GUI as an “unknown router” (with a “?” on the router symbol)

Extended Topology can communicate with this router (over IPv4) using SNMP and can figure out that this device is a router—but Extended Topology cannot obtain the more detailed router information that is provided via the IPv6 General MIB.

IPv6 Management Limitations

Limitations: A Partial List of Features Not Supported

- Routers that do not support the General MIB or that are unable to run dual stack
- Layer 2 management
- Other IPv6 features (security, mobility, multicast and anycast management, full tunnel management)
- On-demand polling, distributed management (DIM), high availability, data collection & trending

©2004 Hewlett-Packard Development Company, L.P.

IPv6 Discovery

IPv6 Discovery

- Initial discovery is defined by the seed file.
- The Discovery agent uses IPv6 DNS to obtain hostname and IPv4/IPv6 address associations.
- Periodic discovery of IPv6 nodes and connectivity; By default, discovery runs once every 24 hours.
- Stored as separate objects in Extended Topology database

©2004 Hewlett-Packard Development Company, L.P.

Make sure that your DNS is configured correctly for both name?address lookups and reverse (address?name) lookups.

Configuring IPv6

Configuring IPv6

- You MUST configure:
  - Router addresses in the IPv6 seed file
  - Community strings for each IPv4 address (using NNM)

- Configuring the following items is OPTIONAL:
  - Router connectivity information (in the seed file)
  - Re-discovery interval
  - Prefix names
  - Polling frequencies

U5089S ©2004 Hewlett-Packard Development Company, L.P. 26

Before starting your IPv6 discovery, you need add some information to three files. The following four files control your IPv6 discovery and status polling:

- `IPv6Seed.conf`
- `IPv6Prefix.conf`
- `IPv6Polling.conf`
- `IPv6.conf`

These files are located in `$OV_CONF/nnmet` on UNIX and `%OV_CONF%\nnmet` on Windows.

You configure these files to adjust discovery and polling to meet your needs. Each file contains configuration examples and instructions showing how to add information.

Configuring IPv6 Seed File

Modifying the IPv6 Seed File (1)

- Discover existence and Layer 3 connectivity of IPv6 networks
  - Turn on or off in **`setupExtTopo.ovpl`**
  - Toggle whether to stop with seeded nodes
- Add all of the IPv6 routers in your network to the IPv6 seed file, located at: **`$OV_CONF/nnmet/IPv6Seed.conf`**
- The file format (for each line) is:

  <IP version> <IPv4/IPv6 address> <hostname>
  - separated by space(s) or tab(s)
- For example:

  6 10.2.115.12 bigrouter1
- For best results, use the IPv4 loopback address for each router that will be managed in your environment (one entry per line).

©2004 Hewlett-Packard Development Company, L.P.

Use the following procedure to configure your IPv6 discovery:

1. Modify the `IPv6Seed.conf` file.

   Extended Topology uses the `IPv6Seed.conf` file to seed your IPv6 discovery. For best results, enter the addresses of all of the routers and end nodes you want to discover into this file. To enter these addresses, follow the instructions included within the `IPv6Seed.conf` file.

   The SNMP MIB information needed to manage the router is obtained using IPv4 and the IPv4 address. The loopback address is the most reliable choice.

   In both IPv4 and IPv6, the term “loopback address” is used to define a virtual interface address for a network device. This virtual address is more reliable and robust than a normal interface address, because it can be used to connect to the device over different routes. That is why we recommend use of the IPv4 loopback address in the seed file.

   (The term “loopback address” is also used to refer to a fixed address, which is used for communication testing on the protocol stack on a SINGLE DEVICE, FROM WITHIN THAT DEVICE. In that context, the IPv4 “loopback address” is always 127.0.0.1, and the IPv6 “loopback address” is always ::1. Nothing is sent out over the network. This is NOT the type of loopback address that we are recommending for the seed file.)

   You MUST enter “6” as the first parameter. This indicates that you are providing information for an IPv6 network.

The second parameter can be either an IPv4 address or an IPv6 address. If you provide an IPv6 address as the second parameter, Extended Topology uses DNS to try to get the corresponding IPv4 address. If there is no corresponding IPv4 address, the node is shown as an end node. Extended Topology relies on IPv4 for SNMP communications.

You may provide a hostname as the third parameter. Extended Topology uses DNS to get a corresponding IPv4 address for this router but the result may not be optimal, because a router hostname maps to multiple addresses in DNS.

2. If you are installing Extended Topology for the first time, you need to execute the `setupExtTopo.ovpl` script and follow the instructions. Once you complete this step, you just started your first discovery.
3. If you executed the `setupExtTopo.ovpl` script prior to configuring your IPv6 discovery, you need to execute, as root, the `etrestart.ovpl` script. This starts a new Extended Topology discovery.

NOTE IPv6 discovery relies heavily on both forward and reverse name resolution. You can achieve name resolution by using either DNS or hosts files. Make sure all of your IPv6 addresses resolve properly before proceeding.

Execute the `/opt/OV/support/NM/getIPv6NameByAddr IPv6Addr` (located in *`install_dir`*`\support\NM` on Windows) command to check reverse name resolution.

See “Troubleshooting IPv6 Discovery” in the Extended Topology manual for more information about diagnosing IPv6 discovery problems.

Configuring IPv6 Discovery

Modifying the IPv6 Seed File (2)

•(Optional) You may also use the seed file to provide router connectivity information. This is a way to provide additional information to the discovery agent and is useful as a default value.

•For example:

6 3FFE:501:811:12::2/64 mainrouter5

©2004 Hewlett-Packard Development Company, L.P.

Extending Discovery

Another option is available for extending discovery: you can enter information on end-nodes or IPv6-only routers in the seed file. If you add such information, please do so at the BEGINNING of the seed file and use their IPv6 addresses (for end-nodes) or IPv6 address and prefix length (for routers, as shown on the above slide) for the second parameter. If you are having trouble with certain end-nodes and routers not being discovered, this will solve the problem.

Limiting Discovery

Discovery starts with nodes specified in this file and proceeds recursively with discovered nodes. Extended Topology reads routing tables from routers and can discover devices beyond the nodes specified in the `IPv6Seed.conf` file. If you want to limit IPv6 discovery to the nodes you add to this file, edit the `IPv6.conf` file and follow the instructions contained in the file.

Consistent Hostnames

Consistent IPv6 Hostname Usage

•It is strongly recommended that you use the same hostname for all of the IPv4 and IPv6 addresses on a particular router or end node. (If you use different hostnames for the same device, the device may be displayed in the GUI as two separate symbols.)

• You will also need to use NNM to configure the community string for each router.

- Select **`Options:SNMP Configuration`**
- Configure SNMP community strings for the IPv4 addresses

©2004 Hewlett-Packard Development Company, L.P.

Properly Displaying Routers not Supporting IPv6 MIBs

If you attempt to discover a router that doesn't support the IPv6 MIBs, the router can show up as two different nodes. To try to show the router properly, take the following steps:

1. Ensure that all IPv6 interface addresses are registered in DNS with the same name.
2. Ensure that at least one IPv4 interface of the router is also registered in DNS under the same name.
3. Ensure that you have configured the SNMP community strings in NNM for the IPv4 interface of the router.
4. Ensure that all IPv6 addresses and prefix lengths for the router are listed in the `IPv6Seed.conf` file without specifying an optional name.
5. Identify all end nodes located beyond the router that you wish to manage and enter their addresses in the `IPv6Seed.conf` file since these end nodes won't get discovered automatically. Do not enter the link-local addresses of these nodes.

Properly Displaying IPv6 Nodes with Multiple Addresses

If you configure a node with than one IPv6 address you should list all of the IPv6 addresses for this end node in the `IPv6Seed.conf` file. Please take the following steps:

1. Make sure that all IPv6 addresses for this node are registered in DNS with the same name.
2. Make sure that all IPv6 addresses for this node are listed in the seed file without specifying an optional name.

IPv6 Status Monitoring

IPv6 Status Monitoring

- Status is continuously updated from polling (ICMPv6 pings).
  - By a separate polling agent
- Status changes show up dynamically on the GUI.
- Address-down events are displayed in the NNM Alarm Browsers. A contextual GUI can be launched from the NNM Alarm Browser.
  - Generates NNM events when status changes

©2004 Hewlett-Packard Development Company, L.P.

Configuring IPv6 Polling

Configuring IPv6 Polling

- The Polling frequency can optionally be configured separately for different devices and different parts of the network.
- Edit **`$OV_CONF/nnmet/IPv6Polling.conf`**.
- Each line corresponds to one or more IPv6 hosts.
- Each line must contain 3 parameters:
  - address/hostname/prefix,
  - frequency of pings (in minutes), and
  - ping timeout (in seconds).
  - Use <space> or <tab> between parameters.
  - Examples:

```
myMachine.myDivision.myCompany.com 60 1
3ffe:2d0:811:ff01::13      30             3
3ffe:80f0:823:b::/64       1440    2
*.myDivision.myCompany.com 20  1
```

 ©2004 Hewlett-Packard Development Company, L.P.

Modify the `IPv6Polling.conf` file if you want to change the polling frequency of the nodes. "*" can be used as a wildcard in hostnames and it matches zero or more characters. If an address passes more than one filter, the first one is used.

The command `reloadIPv6PollConf.ovpl` reloads/resets the polling parameters. This command can be run at any time and takes effect immediately.

The default polling interval and time-out values ("DEFAULT 5 1"; at the end of the polling file) can be modified, but NOT eliminated.

IPv6 ping is sent to each address, not per interface. Extended Topology NEVER pings "link-local scoped" addresses.

NOTE Extended Topology IPv4 discovery can be configured to monitor the number of changes reported by netmon and trigger an Extended Topology IPv4 discovery when a threshold is reached. This does not apply to IPv6. Periodic discovery configuration is crucial to IPv6 accuracy.

Examples

Example 1: Ping all of the IPv6 addresses that resolve to:

myMachine.myDivision.myCompany.com every 60 minutes with a 1 second time-out.

Example 2: Ping the IPv6 address 3ffe:2d0:811:ff01::13 every 30 minutes with a 3 second time-out.

Example 3: Ping all of the IPv6 addresses which have the prefix of 3ffe:80f0:823:b::/64 every 24 hours (1440 minutes) with a 2 second time-out.

Example 4: Ping all IPv6 addresses whose name ends with: myDivision.myCompany.com every 20 minutes with a 1 second time-out.

Configuring Prefix Names

Configuring Prefix Names

- Configuration of prefix names is optional.
- Stop all NNM and Extended Topology processes (**ovstop**) before you modify this file.
- The prefix name file is located at **$OV_CONF/nnmet/IPv6Prefix.conf**.
- Each line in the file must contain 2 parameters:

  <prefix> <prefix name>

  with each parameter separated by space(s) or tab(s)
- For example:

```
3FFE:501:811:D::/64          MyTestNet_1
FEC0:C:210::/48     ProductionNetwork2
```

- Run **ovstart** to restart all processes.

©2004 Hewlett-Packard Development Company, L.P.

IPv6 prefix groups are similar to subnets in IPv4. The `IPv6Prefix.conf` file is used to give a user-friendly name to your prefix groups once they are discovered.

When a prefix is discovered, the name in this file is used to identify that prefix in the GUI. Editing this file allows you to replace numerical prefixes with meaningful names.

IPv6 Logfiles

IPv6 Logfiles

- The logfile for the discovery agent is located at:
  /var/opt/OV/log/ovet_daIPv6.log

- The logfile for the polling agent is located at:
  /var/opt/OV/log/monitorIPv6Agent.log

- The logfile for the stitchers is located at:
  /var/opt/OV/log/ovet_disco.log

- See the product release notes and manual for further information.

©2004 Hewlett-Packard Development Company, L.P.

Changes to the polling agent can be made by modifying the file: `$OV_LRF/monitorIPv6Agent.lrf` (located in `%OV_LRF%` on Windows), and then restarting the polling agent. One of the options available in this file is to configure the number of polling agent retries, if the initial ping fails. This feature utilizes the "`-pr`" parameter, followed by the number of retries (for example, "`-pr 1`" for 1 retry). There are additional comments and examples about this option in the file.

Lab Exercises

Lab Exercises

•Review questions

U5089S ©2004 Hewlett-Packard Development Company, L.P. 34

1. What are three reasons for choosing IPv6 addressing?

2. What steps are involved in preparing to manage IPv6 networks?

3. How does Extended Topology handle routers that run only IPv6?

4. What should you place in the IPv6 seed file?

5. How do you control the frequency of status polling for IPv6 nodes?

6. What is the IPv6Prefix.conf file used for?

I Lab Solutions

Chapter 1, Introduction

1. Describe the steps and interfaces involved in management by exception.

   Answer:

   In management by exception, when an error occurs in the network, an alarm is generated and placed in the Alarm Browser. By selecting the alarm and selecting Actions:Views, you can launch a dynamic view targeted at that type of failure alarm. In the dynamic view, you can access additional information about the device(s) in question as well as the troubleshooting tools to resolve the problem.

2. Name four ways to reduce the number of messages in the Alarm Browser.

   Answer:

   Set the event to Log Only.

   De-duplicate the event.

   Remove it with an ECS correlation.

   Remove it with a Composer correlator.

3. How does Event Correlation assist NNM in addition to keeping the Alarm Browser uncluttered?

   Answer:

   Event correlation can feed new events into `pmd` so that other processes have better information for network management. It can also reduce the number of events that other processes must handle.

4. Name three protocols that Extended Topology manages. What are the product requirements?

   Answer:

   Some of the protocols that Extended Topology manages include OSPF, HSRP, and IPv6.

   These protocols require the purchase of the NNM Advanced Routing SPI.

Lab Exercises

Objective: The purpose of this lab is to build your ability to launch a user interface using either the traditional Graphical User Interface (GUI), or the Web-based User Interface.

NOTE: All of the executables mentioned in the lab solutions are in the OpenView binary directory, unless stated otherwise. The OpenView binary directory is:

`\Program Files\HP OpenView\bin` (on Windows)
`/opt/OV/bin` (on UNIX)

1. From Home Base, start the Internet view.

   Answer:

   Start your web browser and type the URL `http://`*`hostname`*`:7510`. On Windows, you can also click your desktop icon or select `Start->Programs->HP OpenView->Network Node Manager`

Home Base.

Select Internet View from the drop-down list and click [Launch View].

2. Go to the Alarm Browser from Home Base and launch a Neighbor View.

   Answer:

   a. From Home Base, click on the Alarm Browser tab.

   b. Click the button for All Alarms.

   c. Select the license notification from your computer.

   d. Select Actions:Views->Neighbor.

   e. Click [ ] Include End Nodes and [Refresh].

3. Review the current lab environment.

Chapter 2, Discovering Connectivity with Extended Topology

Review Questions

1. Describe the differences between `netmon` discovery and Extended Topology discovery.

   Answer:

   `netmon` discovery finds nodes on the network whereas Extended Topology revisits those found nodes to gather additional information.

   `netmon` discovery is based primarily on standard (MIB II) data whereas Extended Topology gathers proprietary MIB data.

   `netmon` discovery runs on a continuous basis and provides information for display as it arrives. Extended Topology discovery runs only at specified times and only provides display data updates when discovery is completed.

2. What processes are involved in Extended Topology discovery?

   Answer:

   `ovet_bridge` passes information from the topology database (`ovtopmd`) to the Extended Topology processes.

   `ovet_disco` takes that list of nodes and schedules agents, helpers, and stitchers to collect the data and interpret it.

3. Which file lists the nodes from `netmon` discovery that are passed to Extended Topology discovery?

   Answer:

   The file `hosts.nnm` lists the nodes which will be included in Extended Topology discovery.

Chapter 3, Enabling Extended Topology

This lab will introduce you to some of the features in the Extended Topology product. In this lab you will set up Extended Topology, load a static database and explore some of the different views that are part of the Extended Topology capabilities.

Enabling Extended Topology

1. Check the current list of processes.

   Answer:

   ```
   #$OV_BIN/ovstatus -c
   ```

2. While logged in as root, enable Extended Topology.

   Answer:

   ```
   #$OV_BIN/setupExtTopo.ovpl
   ```

   The script prompts for a few answers. Answer yes to everything.

   For the Extended Topology administration user and password, use `ov`. You will need these when you access Extended Topology configuration from Home Base.

   Run `ovstatus -c`.

3. Start the Extended Topology Discovery Status monitoring interface.

   Answer:

   Start Home Base and click the Discovery Status tab.

4. Monitor discovery status.

   Answer:

   Watch Home Base and monitor the text under the progress bar, or repeatedly run `$OV_BIN/ovstatus -v ovet_disco`. When the message says “Awaiting next discovery cycle,” it is complete.

5. Check the current list of processes.

   Answer:

   ```
   $OV_BIN/ovstatus -c
   ```

   You will see that there are a number of new processes that all start with an `ovet` name. If these appear in the list (RUNNING or NOT_RUNNING), you have Extended Topology enabled.

6. Check for new information about the topology.

   Answer:

   You can look at various views, but most classroom installations do not have connector devices supported by Extended Topology. `Tools:Topology Summary` (or [`Show Topology Summary`] on the Home Base Discovery Status tab) shows that Extended Topology is now enabled.

Review Questions

1. From the Home Base Discovery Status tab, select `[Extended Topology Configuration]`.
2. What options are available for controlling the frequency of discovery?

   Answer:

   Extended Topology Discovery can be set to:

   - Discover whenever Extended Topology is restarted
   - Discover on a recurring basis
   - Based on the number of changes reported by NNM
   - Based on a specified time and day
3. If Extended Topology has already been enabled, how would you go about launching another discovery process?

   Answer:

   Execute `$OV_BIN/etrestart.ovpl` or Select [`Initiate Full Discovery Now`] in Home Base.

Installing the Extended Topology Demo

To see additional Extended Topology features, install the Extended Topology demo:

The demo is not complete, but can be used to show all the new views plus launch these views directly from alarms in the browser. This demo should not be installed on a system which needs its NNM or Extended Topology environment to be restored after installing the demo.

HP-UX Installation

1. Due to various types of caching, it is best to eliminate browser temporary files and java caches after demo installation, prior to bringing up the home base. Run through the demo once to cache as much as possible before bringing in the customer. Have the home base already up.
   a. rm -rf ~/.netscape/cache
   b. rm –rf ~/.jpi_cache
2. `cp DemoNNM75.tar.gz /tmp`
3. `gunzip /tmp/DemoNNM70.tar.gz`
4. Make a directory to hold the demo files. You must use precisely this directory name.
   a. `tar -xvf /tmp/DemoNNM75.tar`
5. Execute `/opt/OV/contrib/DemoNNM75/bin/setupDemo.ovpl` to run the install script.
   a. Verify all requested information.
   b. Say yes to all protocols and services questions.

To show views, choose connector devices from ovw or dynamic views and launch views. You can also choose events in the alarm browser and launch views directly. There are events for HSRP and OSPF in the All Alarms Browser that can be used to launch those views.

Windows Installation

1. Due to various types of caching, it is best to eliminate browser temporary files and java caches after demo installation, prior to bringing up the home base. Run through the demo once to cache as much as possible before bringing in the customer. Have the home base already up.
   a. Run the Control Panel (Start->[Settings]->[Control Panel])
   b. Double-click “Internet Options”
   c. Select [General] Tab
   d. In the “Temporary Internet Files” section, Select [Delete Files…] button
   e. Select [OK], then [OK] again to get back to the control panel
   f. Double-click “Java Plug-In”
   g. Select [Cache] Tab
   h. Select [Clear] button, respond [Yes] when asked to clear
   i. Close Java Plug-in window and the Control Panel
2. Locate the `DemoNNM75.exe` file.
3. Unzip the demo by double-clicking the executable. Accept the default target directory. Verify that it installs directly under the `\Program Files\HP OpenView\contrib\DemoNNM75` directory (does not create another Demo* directory under it).
4. Double-click `\Program Files\HP OpenView\contrib\DemoNNM75\bin\setupDemo.ovpl` to run the install script.
   a. Verify all requested information.
   b. Accept the default directory.
   c. In the WinZip self-extractor window, be sure to click [`Unzip`], then click [`Close`].
   d. Say yes to all protocols and services questions.
   e. Ignore the error about replacing an SNMP entry.

To show views, choose connector devices from ovw or dynamic views and launch views. You can also choose events in the alarm browser and launch views directly. There are events for HSRP and OSPF in the All Alarms Browser that can be used to launch those views.

Exploring Extended Topology Views

1. Exploring some of the Extended Topology Views
   a. Start Home Base.
   b. What Views are available to you?

      Answer:

      The following views are available:

      - Neighbor View
      - Node View
      - Station View

- Internet View
- Network View
- Path View
- VLAN View
- Problem Diagnosis View
- OSPF View
- HSRP View
- IPv6 Global Network View (if your system has IPv6 installed)
- Overlapping Address Domain View

c. Launch an Internet View. Select `Tools:Topology Summary`.

What information is present here?

Answer:

This summary presents the overall topology information known by NNM and Extended Topology. Note that one of the key differences is that Extended Topology provides information about your IPv6 environment. The difference in the number of nodes found comes from the demo updating the Extended Topology database with nodes that are not in the `ovtopmd` database.

d. Close the Topology Summary window and the Internet View.

2. From Home Base, select OSPF View.

a. How many areas are connected to Area 0.0.0.0?

Answer:

In the graph you can see seven additional areas.

b. Click on the All Areas tab. How many areas are defined here?

Answer:

Here you see eight areas: Area 0.0.0.0 plus the others.

c. Expand Area Name 0.0.0.0 in the table. How many area border routers are there?

Answer:

This area contains four area border routers.

d. Go back to the Graph tab and double-click Area 0.0.0.90. Compare this to the listing for the same area in the All Areas table.

Answer:

In the All Areas tab, click on Area Name 0.0.0.90. Each view shows four routers and the six interfaces used to connect them. The table also shows the OSPF link metrics.

3. From Home Base, start the VLAN View. How many devices participate in the WIRELESS VLAN (which is not in an Overlapping Address Domain)? What happens when you double-click on one of the devices?

Answer:

Three devices participate. Double-clicking shows the node details for the device.

4. Neighbor Views

a. From the VLAN view select (single-click) the 10.96.30.2 device in the WIRELESS VLAN.

b. From the same browser window, select `Tools:Views->Neighbor View`.

c. By default, how many hops are viewed and are end nodes displayed?

Answer:

The default is to display 2-hops and no end nodes.

d. Change the number of hops to 3 and check the box for Include End Nodes. Then click [`Refresh`].

Note the changes to the display.

5. Right-click in the background of the Neighbor View and select `Highlight VLAN->WIRELESS`. What happens?

Answer:

White boxes are added around the devices participating in the VLAN.

Remove the Demo Topology

To facilitate your experience in the rest of the course modules, please remove the demo topology and allow your system to rediscover the classroom.

1. From the demo directory, run the unsetup script.

- UNIX:

a. `cd $OV_CONTRIB/DemoNNM70/bin`

b. `unsetupDemo.ovpl`

c. Accept all defaults.

d. `cd $OV_TMP` and run the cleanup script that has been created there.

- Windows:

a. In Windows file explorer, browse to *`install_dir`*`\contrib\DemoNNM75\bin`.

b. Double-click `unsetupDemo.ovpl`.

c. Accept all defaults.

d. In Windows file explorer, browse to *`install_dir`*`\tmp` and run the cleanup script that has been created there.

Chapter 6, Controlling Extended Topology Discovery

Preparing for Extended Topology Labs

Setting Up Extended Topology MIMIC Labs

Objectives:

Successfully setup the student workstation in preparation for execution of the Extended Topology Labs.

At the completion of this module you will be able to:

- Successfully execute and complete the labs.

Student Workstation System Setup Procedure:

NOTE UNIX: This procedure assumes that everything is executed as User root in a ksh environment and that the `/opt/OV/bin/ov.envvars.sh` file has been sourced in each working window.

Windows: This procedure assumes that `%OV_BIN%\ov.envvars.bat` has been executed in a cmd window.

CAUTION The procedure described for the Student's workstation assumes that NNM has been installed AND the Extended Topology component has been setup using the `$OV_BIN/setupExtTopo.ovpl` script.

During the `setupExtTopo.ovpl` script, answer YES to all questions. When you are prompted for the Administrator password (for tomcat), enter a username of `ov` and a password of `ov`. Allow discovery to complete before continuing.

1. Install the lab files. They must go in exactly the right directory. Execute

   UNIX:

   ```
   gunzip NNM7labs.tar.gz

   tar –xvpf NNM7labs.tar
   ```

   Windows: Install the `NNM7labs.zip.exe` self extracting file. Ensure that the files extract into the `%OV_CONTRIB%\NNM7labs` directory.

2. Initialize the labs using the IP address of the MIMIC server that your instructor provides.

   ```
   cd $OV_CONTRIB/NNM7labs
   initialsetup.ovpl ip_addr_MIMICSERVER
   ```

3. You will want to be able to save images of discoveries for comparison with broken discoveries. Enable `File:Save` and `File:Load` from dynamic views.

   The following labs require you to save dynamic views for later comparison. These options are not enabled by default in the product. To enable these menu options, do the following:

   a. Close all browsers and java consoles.

b. Change to the directory `$OV_WWW_REG/dynamicViews/$LANG/`.

c. Save a copy of the current `dynamicViews.xml` file as #dynamicViews.xml.

d. Use an editor to edit the file `dynamicViews.xml`.

e. Search for the string "Save/Load: unsupported".

f. Delete both lines that have the string "Save/Load: unsupported". One starts with `<!--` and the other ends with `-->`. You must correctly delete both lines or the XML will have mismatched brackets.

g. Save the new `dynamicViews.xml` file.

h. Clear the browser and java caches:

- UNIX:

  ```
  rm -rf ~/.netscape/cache
  rm -rf ~/.jpi_cache
  ```

- Windows: Select Start->Settings->Control Panel.

  Double-click Internet Options.

  Under Temporary Internet Files, press the Delete Files button, click OK.

  In the control panel, double-click Java Plug-in.

  Under the Cache tab, press the Clear button.

  Close all windows.

You may have to disable proxying for your web browser as proxies also cache. See your instructor for information on this.

4. Your system is ready to begin the labs.

Troubleshooting Deployment

Lab D: Full Discovery, Setting the Stage

Objectives:

- Set the stage for the following labs: discover the correct topology.
- Gain additional experience using the support tools against additional agents not used before.

At the completion of this module you will be able to:

- Run several Extended Topology discovery process monitoring tools and understand their results.
- Execute the database dump tools and understand the results.

Assumptions:

- `$OV_CONTRIB/NNM7labs/initialsetup.ovpl` has been executed.

Directions

1. Change working directory to `$OV_CONTRIB/NNM7labs/Lab_deploy`:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_deploy
   ```

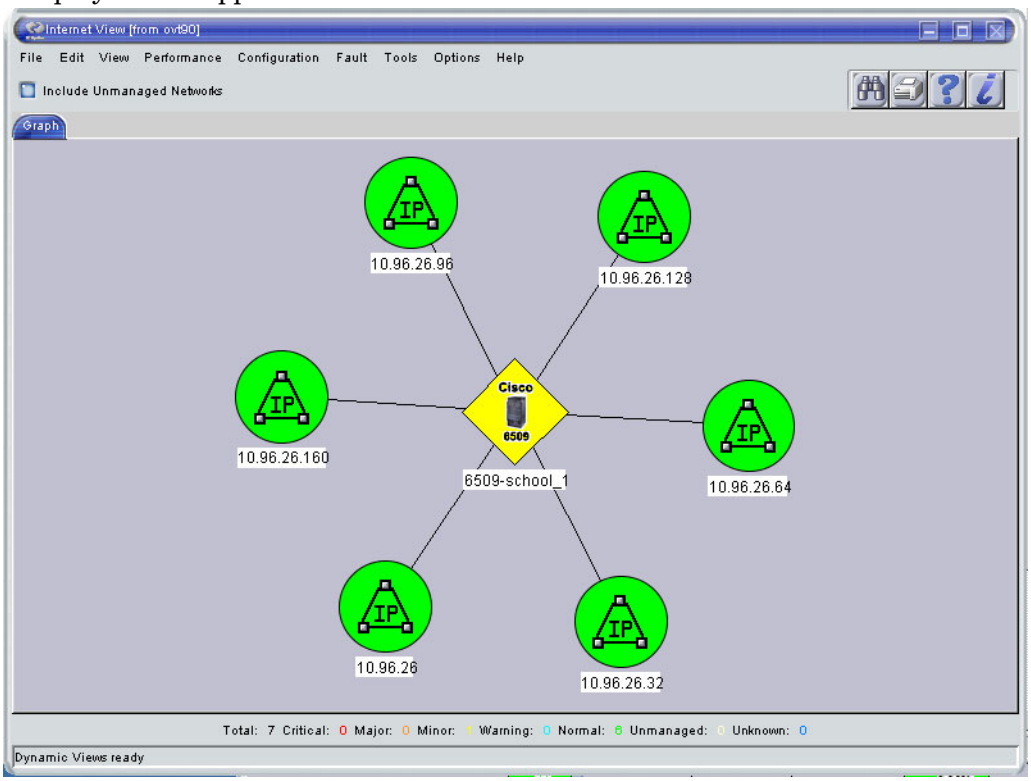
2. When the instructor has given you the go-ahead, execute the setup script:

   ```
   setupLab_deploy.ovpl
   ```

3. When this completes, open an Internet View from Home Base.

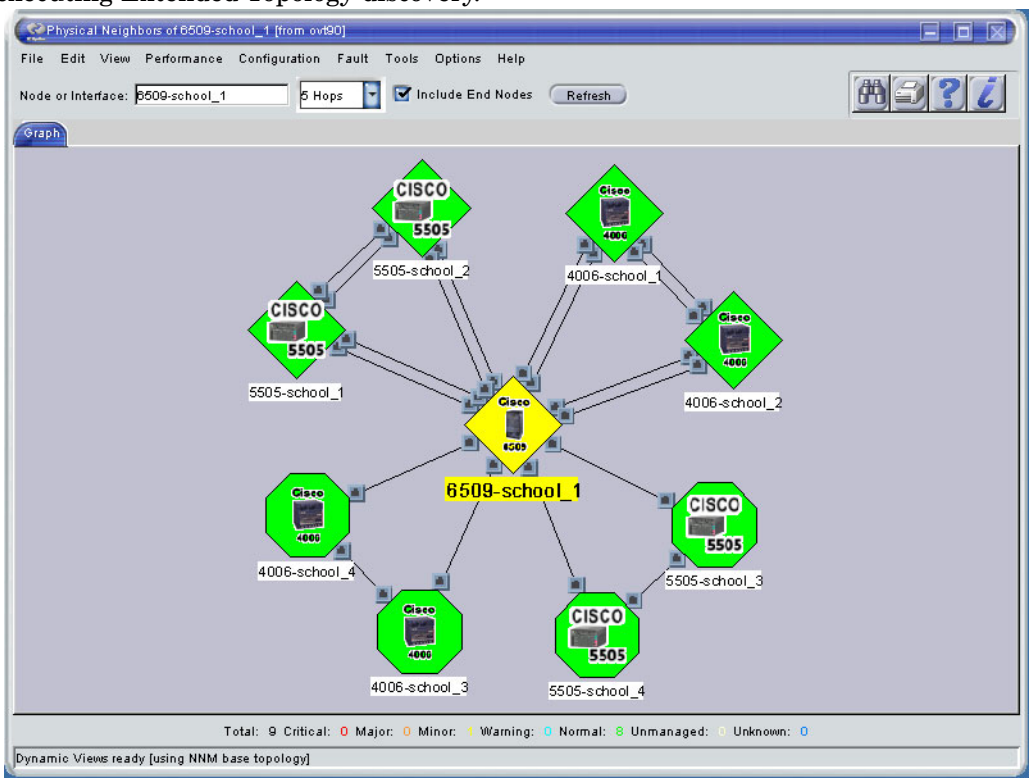
4. Verify proper and complete discovery of the simulated network. Locate any symbol that may be unmanaged or unknown. If unmanaged, select it, and then use the menu `Edit:Manage`. For any node which shows as a blank square, select the node and `Fault:Network Connectivity:Poll Node`.

5. The display should appear as shown.



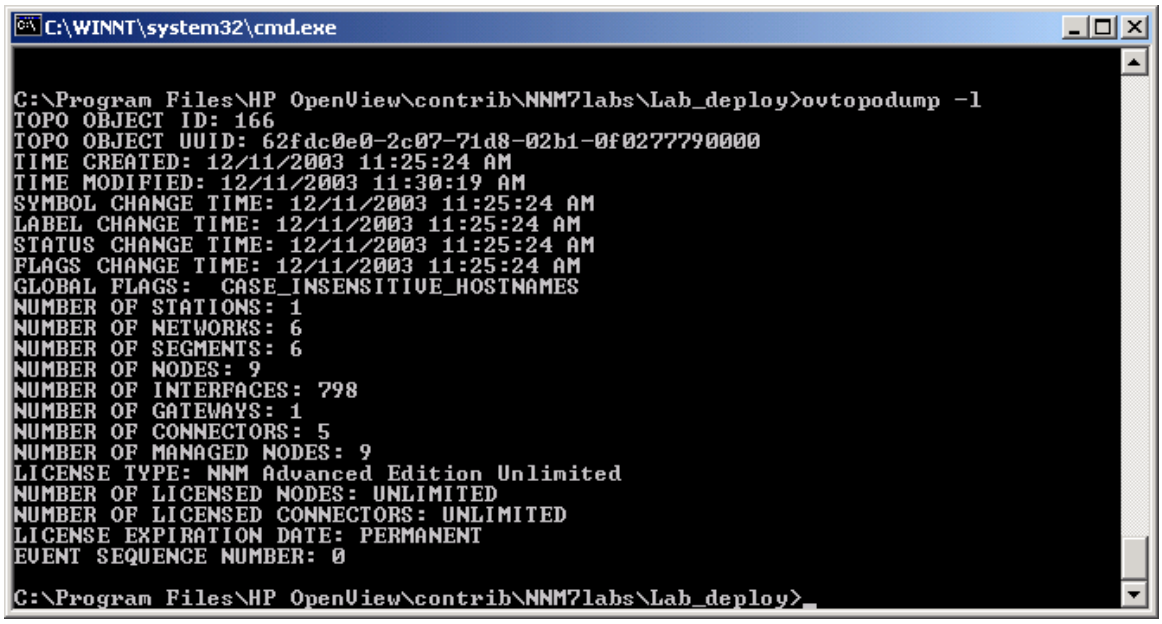
6. Execute a Neighbor view of the device 6509-school_1 with 3 hops and include end nodes, prior

to executing Extended Topology discovery.



7. Confirm correct topology has been discovered:

```
ovtopodump -l
```



8. When you have confirmed that all devices are managed and appear as shown in the above figures, commence the Extended Topology discovery from Home Base.

NOTE You will run several iterations of Extended Topology discovery throughout these labs. Each execution of the discovery will give you an opportunity to use the support tools to view progress of the discovery processes and agents. These

tools will be especially important in the labs that simulate discovery problems.

Be aware that the topology is quite small and the overall discovery process is brief. Hence the tools that dump status or progress and watch the log file(s) are only "active" a short period of time (during actual discovery). Run several discoveries in order to understand the full impact of these labs.

9. Once discovery has started (entered Phase 1 as indicated by `ovstatus -v ovet_disco`), in a separate window tail the `disco_log` file. Be aware that this file is renamed to `ovet_disco.log.old` when `ovet_disco` is restarted and when it completes.

   ```
   tail -f $OV_PRIV_LOG/ovet_disco.log
   ```

10. Execute:

    ```
    $OV_MAIN_PATH/support/NM/dumpDiscoStatus.ovpl
    ```

    to monitor the overall progress of discovery and to identify the state of each agent.

    m_State= 1 or 2 or 3 or 4. State 4 indicates completion.

    You are interested in m_State=3 (active) and 4 (completed).

    However, if you execute `dumpDiscoStatus.ovpl` too early, you will get stuck inside the query command as follows (the same occurs with dumpAgentProgress.ovpl):

    ```
    #> dumpDiscoStatus.ovpl

    ================================================Start

    ovet_oql ( Command Line OQL )

    Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.


    Executing query:

    select * from disco.status;

    ================================================END
    ```

    If you are stuck inside the query tool, you may get the prompt:

    ```
    |"tntdemo2:2.>"
    ```

    If so, press the Enter key and then type `quit;`.

    You will either see a listing of progress or exit the tool. If you exit, re-execute the tool.

A correct execution of the tool should result in a data dump similar to:

```
#> dumpDiscoStatus.ovpl
================================================Start
ovet_oql ( Command Line OQL )
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.

ovet_auth has authenticated your session.
Executing query:
select * from disco.status;
.
{
        m_DiscoveryMode=0;
        m_Phase=-1;
        m_BlackoutState=1;
        m_CycleCount=0;
}
( 1 record(s) : Transaction complete )
ovet_oql ( Command Line OQL )
```

```
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.

ovet_auth has authenticated your session.
Executing query:
select * from agents.status where m_State <> 0;
.......
{
        m_Name='CDP';
        m_State=4;
        m_LastRecordTime=1044731840;
        m_NumConnects=1;
}
{
        m_Name='Details';
        m_State=4;
        m_LastRecordTime=1044731839;
        m_NumConnects=1;
}
{
        m_Name='EDP';
        m_State=4;
        m_LastRecordTime=1044731843;
        m_NumConnects=1;
}
{
        m_Name='ExtremeSwitch';
        m_State=3;
        m_LastRecordTime=1044731965;
        m_NumConnects=1;
}
{
        m_Name='ILMI';
        m_State=4;
        m_LastRecordTime=1044731850;
        m_NumConnects=1;
}
{
        m_Name='InterfaceDetails';
        m_State=4;
        m_LastRecordTime=1044731859;
        m_NumConnects=1;
}
{
        m_Name='StandardSwitch';
        m_State=3;
        m_LastRecordTime=1044731964;
        m_NumConnects=1;
}
( 7 record(s) : Transaction complete )
#>
================================================END
```

While the m_State=3, the agent is not finished and the results are likely to be incomplete.

NOTE One of the more common problems with discovery is an agent appears to “hang” at m_State=3. By default, agents may take up to one hour before they time out. The training setup has reduced this to 10 minutes.

For a short period of time after the discovery has completed, you may be able to view the results of the agent when the m_State=4.

If you execute the `dumpDiscoStatus.ovpl` command and it returns the following output, you will not be able to view any agent details.

```
#> dumpDiscoStatus.ovpl
================================================Start
ovet_oql ( Command Line OQL )
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.

ovet_auth has authenticated your session.
Executing query:
select * from disco.status;
.
{
        m_DiscoveryMode=0;
        m_Phase=0;
        m_BlackoutState=0;
        m_CycleCount=0;
}
( 1 record(s) : Transaction complete )
ovet_oql ( Command Line OQL )
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.

ovet_auth has authenticated your session.
Executing query:
select * from agents.status where m_State <> 0;
.
( 0 record(s) : Transaction complete )
================================================End
```

While the m_State =3 (or 4) for an agent, you can run the `dumpAgentProgress.ovpl` tool to get agent details, for example:

```
#> dumpAgentProgress.ovpl ExtremeSwitch
================================================Start
Looking for agent "ExtremeSwitch" in Extended Topology database...

ovet_oql ( Command Line OQL )
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.

ovet_auth has authenticated your session.
Executing query:
select * from ExtremeSwitch.despatch;
...
{
        m_UniqueAddress='10.96.26.193';
        m_Name='black_diamond';
        m_ObjectId='1.3.6.1.4.1.1916.2.11';
}
{
        m_UniqueAddress='10.96.26.194';
        m_Name='3808-1';
        m_ObjectId='1.3.6.1.4.1.1916.2.17';
}
{
        m_UniqueAddress='10.96.26.226';
        m_Name='3808-2';
        m_ObjectId='1.3.6.1.4.1.1916.2.17';
}
( 3 record(s) : Transaction complete )
ovet_oql ( Command Line OQL )
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.

ovet_auth has authenticated your session.
Executing query:
select * from ExtremeSwitch.returns where m_LastRecord = 1;
.
( 0 record(s) : Transaction complete )
```

```
================================================END
```

The above results show that there are THREE entries to be processed, but ZERO have been processed.

The dump listing below shows the three devices (records) exist, and have completed processing.

```
#> dumpAgentProgress.ovpl ExtremeSwitch
================================================START
Looking for agent "ExtremeSwitch" in Extended Topology database...

ovet_oql ( Command Line OQL )
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.

ovet_auth has authenticated your session.
Executing query:
select * from ExtremeSwitch.despatch;
...
{
        m_UniqueAddress='10.96.26.193';
        m_Name='black_diamond';
        m_ObjectId='1.3.6.1.4.1.1916.2.11';
}
{
        m_UniqueAddress='10.96.26.194';
        m_Name='3808-1';
        m_ObjectId='1.3.6.1.4.1.1916.2.17';
}
{
        m_UniqueAddress='10.96.26.226';
        m_Name='3808-2';
        m_ObjectId='1.3.6.1.4.1.1916.2.17';
}
( 3 record(s) : Transaction complete )
ovet_oql ( Command Line OQL )
Copyright (c) 1990-2003 Hewlett-Packard Co.,  All Rights Reserved.

ovet_auth has authenticated your session.
Executing query:
select * from ExtremeSwitch.returns where m_LastRecord = 1;
...
{
        m_UniqueAddress='10.96.26.193';
        m_Name='black_diamond';
        m_UpdAgent='ExtremeSwitch';
        m_Capabilities=['isLanSwitch'];
        m_LastRecord=1;
}
{
        m_UniqueAddress='10.96.26.226';
        m_Name='3808-2';
        m_UpdAgent='ExtremeSwitch';
        m_Capabilities=['isLanSwitch'];
        m_LastRecord=1;
}
{
        m_UniqueAddress='10.96.26.194';
        m_Name='3808-1';
        m_UpdAgent='ExtremeSwitch';
        m_Capabilities=['isLanSwitch'];
        m_LastRecord=1;
}
( 3 record(s) : Transaction complete )
```

```
================================================END
```

This is ONE EXAMPLE of using the `dumpDiscoStatus.ovpl` and `dumpAgentProgress.ovpl` tools.

11. Once discovery completes, execute `$OV_BIN/ovet_topodump.ovpl {-? for usage)` to dump a summary and detailed listing of the results of Extended Topology discovery. For the purposes of our lab direct the output to a file such as:

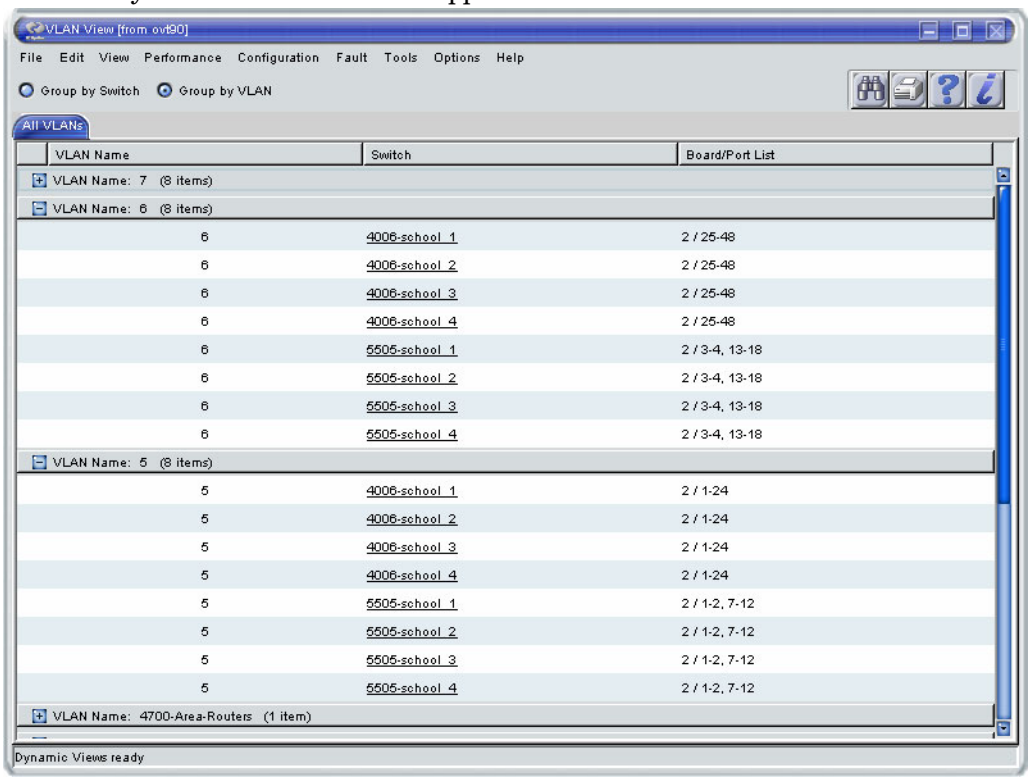
    ```
    ovet_topodump.ovpl -info > labdiscoFull-topo.txt
    ```

    Since you will execute additional discoveries under a variety of simulation scenarios ("broken discovery"), you will need to compare the results in subsequent labs.

12. Since the purpose of this lab is to discover the full topology in preparation for the following labs, use the VLAN view, Node View, and Neighbor Views as directed below to gain familiarity with this topology and the VLAN structure. Subsequent labs will simulate a variety of discovery issues, which will then require you to compare the "broken" discovery with the full discovery.

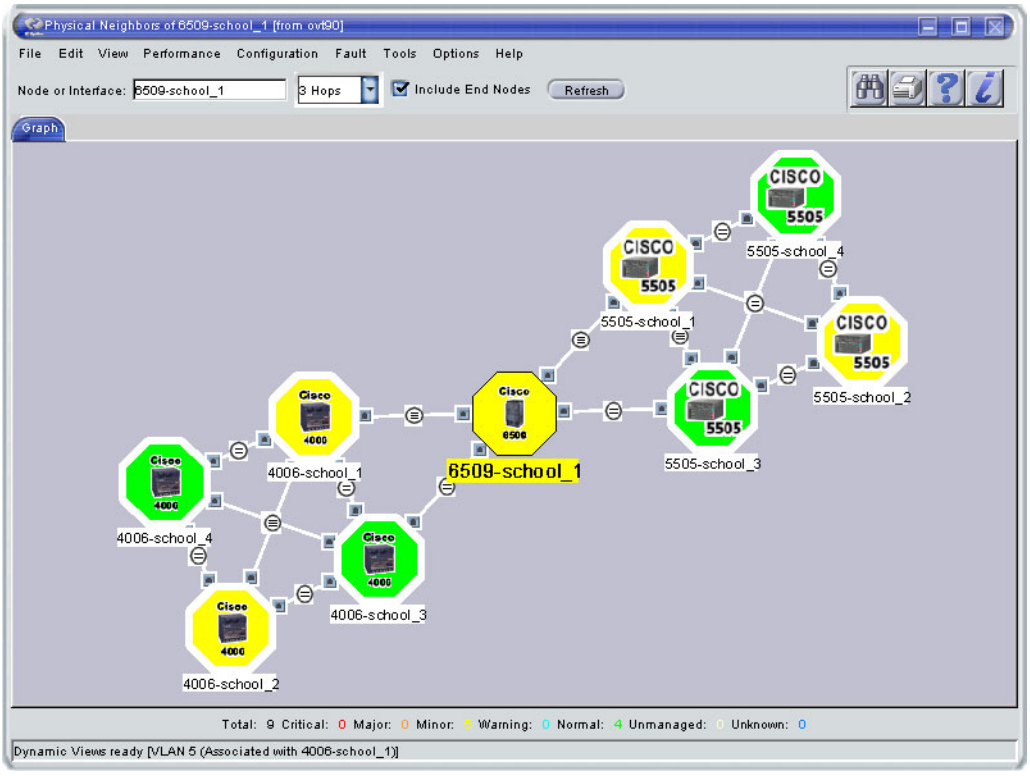
13. Once Extended Topology discovery completes, confirm correct and complete topology has been discovered, load the VLANs view from Home Base. Open the group for OAD Name: Non-overlapping default. You can get a better view by placing the mouse over the header VLAN Name, pressing the right mouse button, and selecting Ungroup.

    Confirm that your VLAN information appears as shown.



14. Now confirm the topology using the Neighbor View from 6509-school_1 using 5 hops and

Include End Nodes.



15. Within the neighbor view of 6509-school_1, use the Highlight VLAN feature and highlight VLAN 5 associated with 4006-school_1. Save the view using the `File:Save` option.

Once you have confirmed the topology discovered is similar to that documented above, inform the instructor you are ready to proceed with Lab E.

CAUTION Do not proceed to Lab E until instructed to do so. The MIMIC simulator must be configured for Lab E.

Lab E: Extended Topology SNMP Failure

CAUTION Successful execution of this lab depends on following the directions given by your instructor AT THE TIME those instructions are issued.

Do not proceed with this lab until the Instructor has said to continue. Changes to the MIMIC server are required.

Objectives:

- Understand the importance of SNMP access to network devices and the impact changes to SNMP access have on Extended Topology discovery and rediscovery.
- Gain additional experience using the support tools in an effort to understand Extended Topology discovery processes and agent behavior when snmp access and/or devices become unavailable.

At the completion of this module you will be able to:

- Properly diagnose when SNMP access failures (or device failures) adversely impact Extended Topology discovery.
- Run several Extended Topology discovery process monitoring tools and understand their results.
- Execute the database dump tools and understand the results.

Assumptions:

- Successful completion of Lab D.

Directions, Phase 1: Rediscovery

Using the base discovery you had for Lab D, wait for the instructor's go-ahead and rediscover the same environment.

1. Re-execute Extended Topology Discovery from Home Base or the command line.
2. Using similar procedures outlined in Lab D, use the tools to view the status/progress/output of the Extended Topology tools used before.
   a. Run several iterations of discovery,
   b. Run `dumpDiscoStatus.ovpl` several times during each discovery
   c. Run `dumpAgentProgress.ovpl` *{agentName}* for a variety of agents several times during a discovery
   d. `tail` the `ovet_disco.log` file
   e. Run `ovet_topodump.ovpl -nodeif`
3. When `ovstatus -v ovet_disco` reports discovery has STARTED (after the initial set of messages), run the tools in separate windows several times throughout discovery, for example:

   ```
   $OV_MAIN_PATH/support/NM/dumpDiscoStatus.ovpl

   $OV_MAIN_PATH/support/NM/dumpAgentProgress.ovpl details
   ```

   a. For example, in one terminal window run `dumpDIscoStatus.ovpl` to determine when the agent(s) are finished (reach m_State=4).
   b. In one or two other windows, execute different `dumpAgentProgress.ovpl` *{agentName}*

based upon the agents listed from the `dumpDiscoStatus.ovpl` command.

c. From the Topology Summary, select `Doesn't Respond to SNMP`. This shows which device(s) (if any) failed to respond to Extended Topology SNMP query and some suggested reasons for the failure.

4. Once discovery completes, execute `$OV_BIN/ovet_topodump.ovpl`.

   Prior to using the tool, `cd` to the directory you want to save the output into, for example:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_deploy/
   ```

   Determine what node(s) if any, is causing problems with discovery. Here are some hints:

   a. Determine which agent is causing the most delay in the discovery progress (`dumpDiscoStatus.ovpl`)

   b. Use the tools provided and pipe the output as shown in the EXAMPLE:

      `dumpAgentProgress.ovpl CDP| grep m_Unique | sort` (note, try different agents)

      Look for those IP address that do not have pairs. This indicates the agent has not completed the query to that device.

   c. Determine if any device failed an SNMP request. From the Topology Summary, select `Doesn't Respond to SNMP`.

5. Do a Node View, Show Nodes: All, Status >= Normal, IP Range: *.*.*.* What is different about this view?

   Answer:

   Nothing is different. It appears exactly as for Lab D.

6. Hold your mouse over 6509-school_1. What information is available in the mouseover?

   Answer:

   The same information as in Lab D. This information has been carried forward from the previous discovery. Unless you attempt a MIB query of 6509-school_1, you have no indication that something changed for this discovery cycle.

Directions, Phase 2: Initial Discovery

This time you will start with a clean database and observe the impact of a non-responsive device during initial discovery.

1. Change working directory to `$OV_CONTRIB/NNM7labs/Lab_deploy`:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_deploy
   ```

2. Re-setup the lab environment to start with a fresh database.

   ```
   setupLab_deploy.ovpl
   ```

3. Re-execute Extended Topology Discovery from Home Base or the command line.

4. Using similar procedures outlined in Lab D, use the tools to view the status/progress/output of the Extended Topology tools used before.

5. Once discovery completes, execute `$OV_BIN/ovet_topodump.ovpl`.

   Prior to using the tool, `cd` to the directory you want to save the output into, for example:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_deploy/
   ```

   Determine what node(s) if any, is causing problems with discovery. Here are some hints:

a. Determine which agent is causing the most delay in the discovery progress (`dumpDiscoStatus.ovpl`)

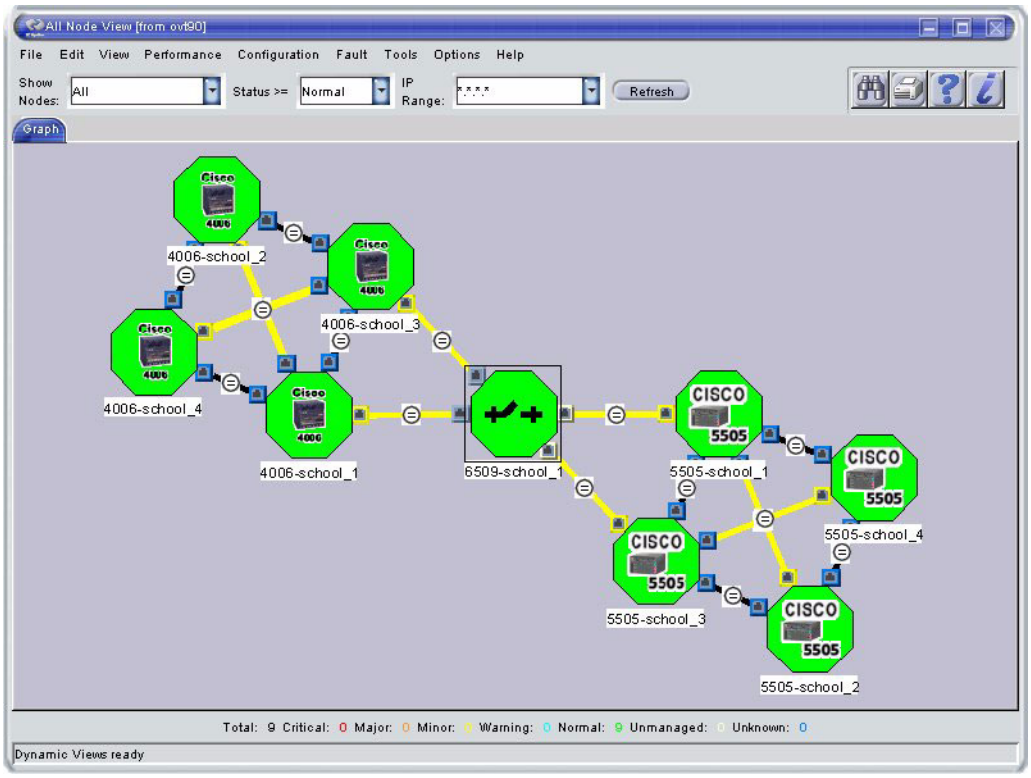
b. Use the tools provided and pipe the output as shown in the EXAMPLE:

`dumpAgentProgress.ovpl CDP| grep m_Unique | sort` (note, try different agents)

Look for those IP address that do not have pairs. This indicates the agent has not completed the query to that device.

c. Determine if any device failed an SNMP request. From the Topology Summary, select `Doesn't Respond to SNMP`.

6. Do a Node View, Show Nodes: All, Status >= Normal, IP Range: *.*.*.* What is different about this view?

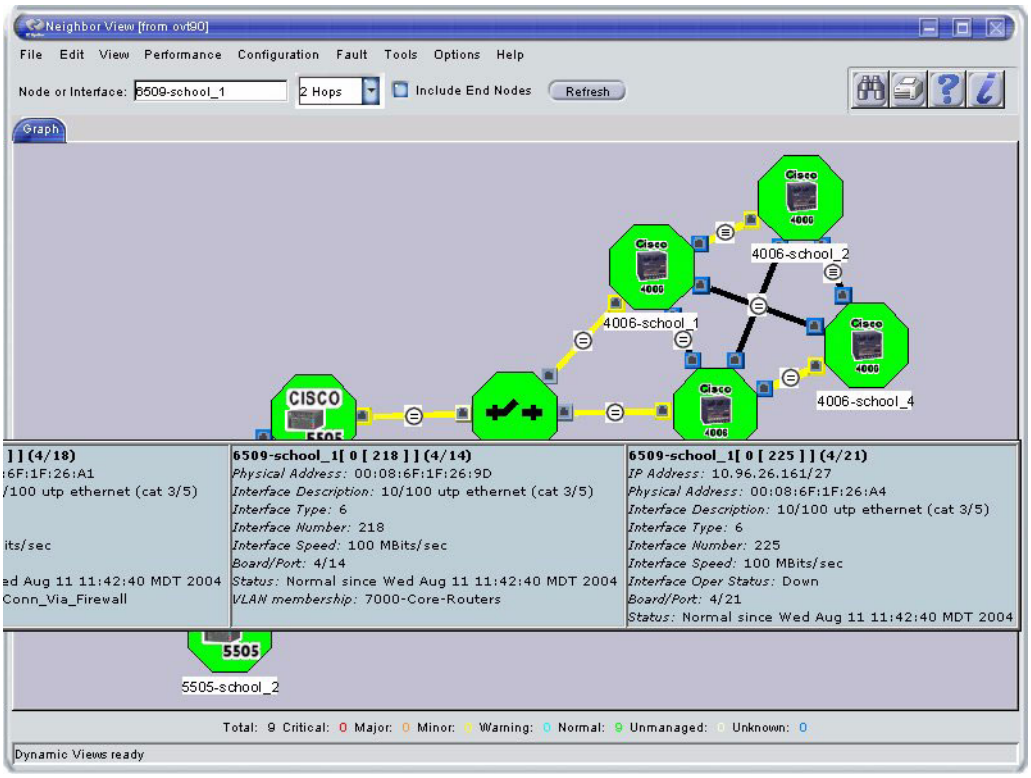


7. Hold your mouse over 6509-school_1 and its interfaces. What information is available?

Answer:

The mouseover indicates that the node was unresponsive during discovery. The additional

information is interpolated from the SNMP data gathered from the neighbors.



Lab E Extended Topology SNMP Failure Review Questions:

1. What is the impact to the Extended Topology discovery process when a device fails to respond to the initial Extended Topology SNMP access test?

   Hint: What were the differences between the Details agent and the other agents?

   Answer:

   a. The access test is attempted for all nodes that are passed to the discovery process by `ovet_bridge` (see the list in `$OV_DB/nnmet/hosts.nnm`).

   b. The details agent will attempt to query all those devices; however, those that fail are NOT passed (dispatched) to other agents, hence the other agent's lists do not include those devices that fail the access test.

2. What was the affect to the views (Neighbor, Node, VLAN, etc.) as a result of incomplete access to all the devices discovered and managed by NNM?

   Answer:

   Information was interpolated from neighboring queries and should be suspect.

3. How can you determine what devices fail to respond to an Extended Topology SNMP query?

   Answer:

   a. From the Topology Summary, select `Doesn't Respond to SNMP`. This shows which device(s) (if any) failed to respond to Extended Topology SNMP query and some suggested reasons for the failure.

   b. Examine the output of `dumpAgentProgress.ovpl Details` and determine which device fails to provide the complete set of information. Specifically which devices fail to provide

the `m_ObjectId` and the `m_Description` values?

a. from the Topology Summary, select `Doesn't Respond to SNMP`. This shows which device(s) (if any) failed to respond to Extended Topology SNMP query and some suggested reasons for the failure.

1. selecting `Doesn't Respond to SNMP`

a. from the Topology Summary, select `Doesn't Respond to SNMP`.

Lab G: Zone Discovery

CAUTION Do not proceed with this lab until the Instructor has said to continue.

Objectives:

- Understand correct behavior of Zone discovery with multi-zone devices.

At the completion of this module you will be able to:

- Implement correct Zone Discovery configurations.

Assumptions:

- Student has completed prior labs.

Directions

1. Execute the Extended Topology Configuration GUI from Home Base, Discovery Status tab.

WARNING **Do not change the configuration for anything in this browser window except as instructed below.**

2. Ensure that the Selection buttons are not checked for both "`Initiate a new discovery whenever Extended Topology is restarted`" and "`Enable recurring discover`".
3. Within the Discovery Zone Configuration box, position the cursor inside the text box besides "Members:"

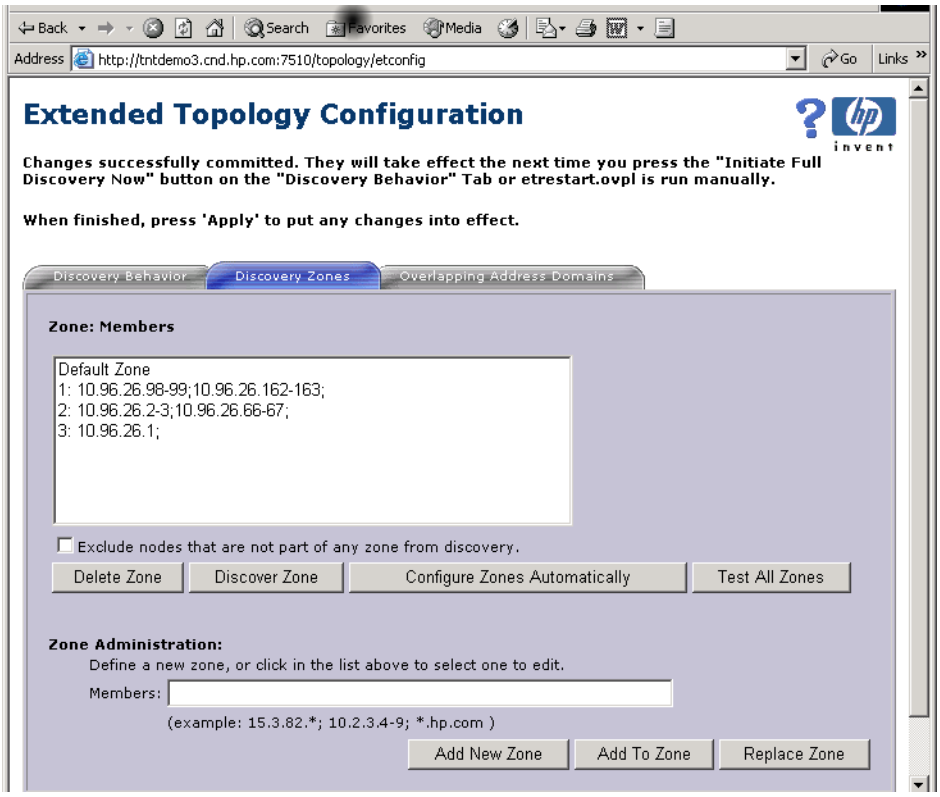
   Type in this box: `10.96.26.98-99;10.96.26.162-163` (Zone 1)
   then click [`Add New Zone`].

   Type in this box: `10.96.26.66-67;10.96.26.2-3;` (Zone 2)
   then click [`Add New Zone`].

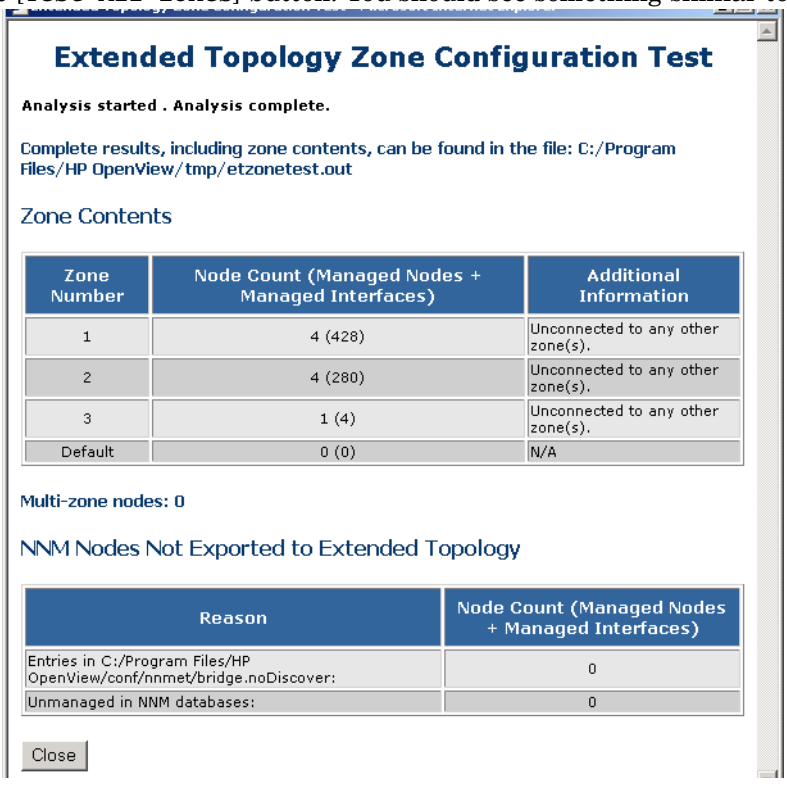
   Type in this box: `10.96.26.1;` (Zone 3)
   then click [`Add New Zone`].

   Click [`Apply`].

The browser window should be similar to this:



4. Now click the [`Test All Zones`] button. You should see something similar to:



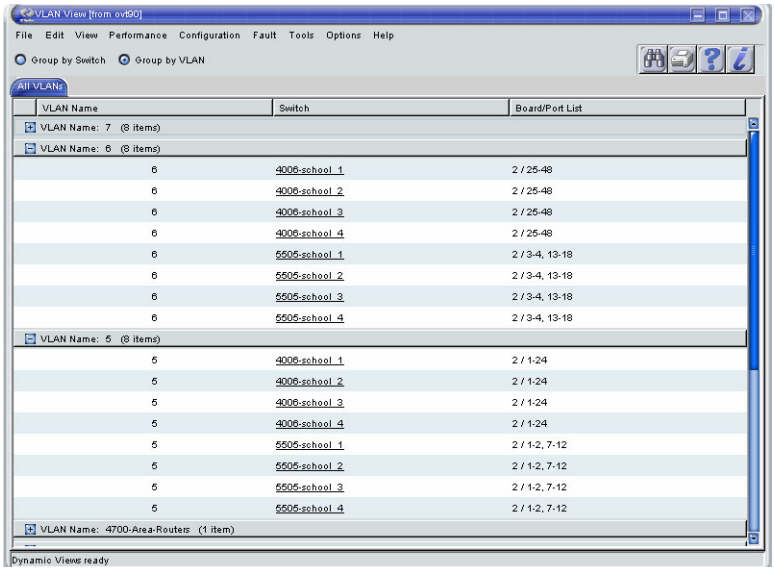
5. Click the "`Close`" button in the Zone Configuration Test window. Then click the "`Cancel`" button in the Extended Topology Configuration window.

NOTE If your system has discovered the local classroom network, you will see nodes in the "default" zone in addition to the nodes in the two zones we just configured and that are part of the Mimic Simulation Network. Recommendation: fix this before proceeding. Ensure New Node Discovery is OFF. Then re-run the `setupLab_deploy.ovpl` script.

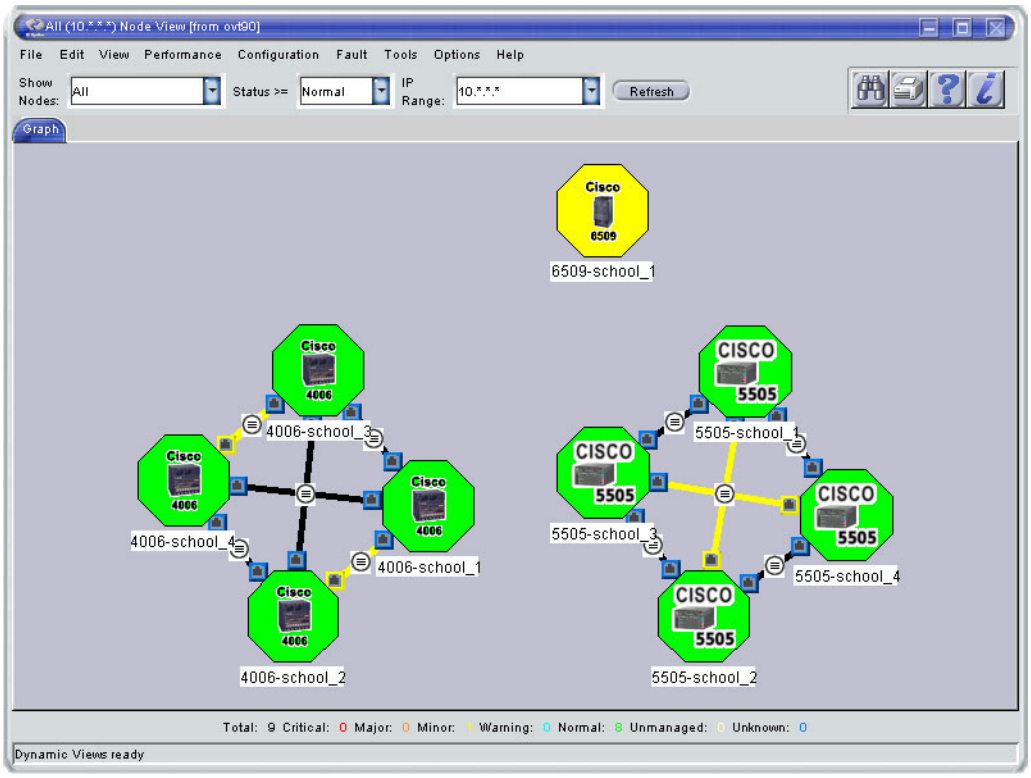
Repoll 6509-school_1 (`Fault:Network Connectivity->Poll Node`) or execute `$OV_CONTRIB/NNM7labs/Lab_deploy/bin/demandPoll3.ovpl`.

6. When the instructor has given you the go-ahead, initiate a full Extended Topology discovery.
7. Using similar procedures outline in previous labs use the tools to view the status/progress/output of the Extended Topology tools used before.
   a. Run `dumpDiscoStatus.ovpl` several times during each discovery
   b. Run `dumpAgentProgress.ovpl` *{agentName}* for a variety of agents several times during a discovery. Of particular interest will be the `CiscoSwitchSnmp` agent.
   c. `tail` the `ovet_disco.log` file
   d. At the completion of discovery run `ovet_topodump.ovpl`.
8. Confirm you have discovered the full topology similar to Lab D using the three zones. Execute a Node view as before.
9. Check the VLANs view to ensure full VLAN discovery WITHOUT overlap or duplicate contents.
10. Once discovery has completed, use the `ovet_topodump.ovpl` tool to isolate any difference.

    Hint: write the output to a file.
11. Use two windows side by side to compare Lab G and Lab D results as you did in earlier labs. Compare the Topology Summary page, and the results from the `ovet_topodump.ovpl`, etc.
12. Do a VLAN View. How does this differ from Lab E?



13. Do a Node View. How does it differ from Lab E?



Answer:

The VLANs are disconnected and 6509-school_1 is discovered.

14. Before proceeding to other labs, remove the zone configuration.

    a. Execute the Extended Topology Configuration GUI from Home Base, Discovery Status tab.

WARNING **Do not change the configuration for anything in this browser window EXCEPT as instructed below**

    b. Within the Discovery Zone Configuration box, and inside the `Zone: Members` list, select the first zone and then click [`Delete`].

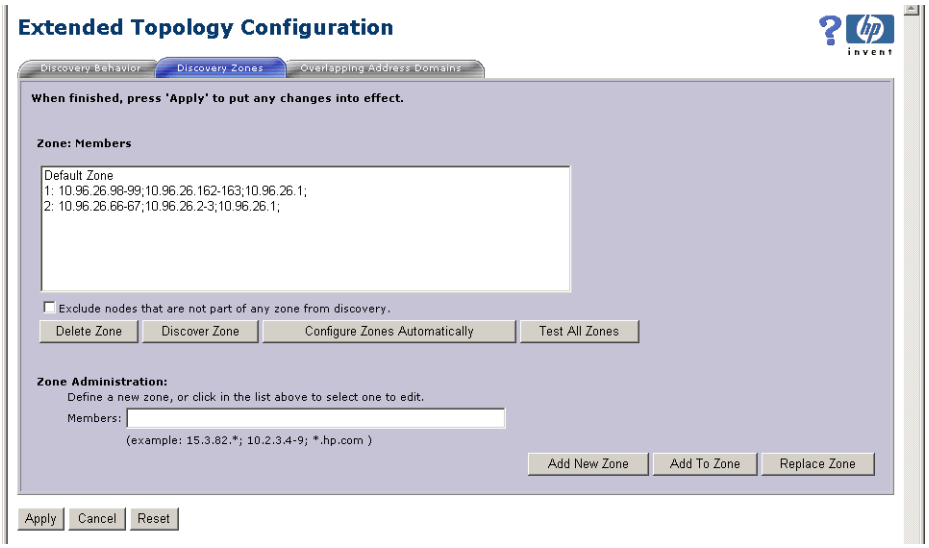
    c. Select the remaining zones and click [`Delete`].

    d. Click [`Apply`].

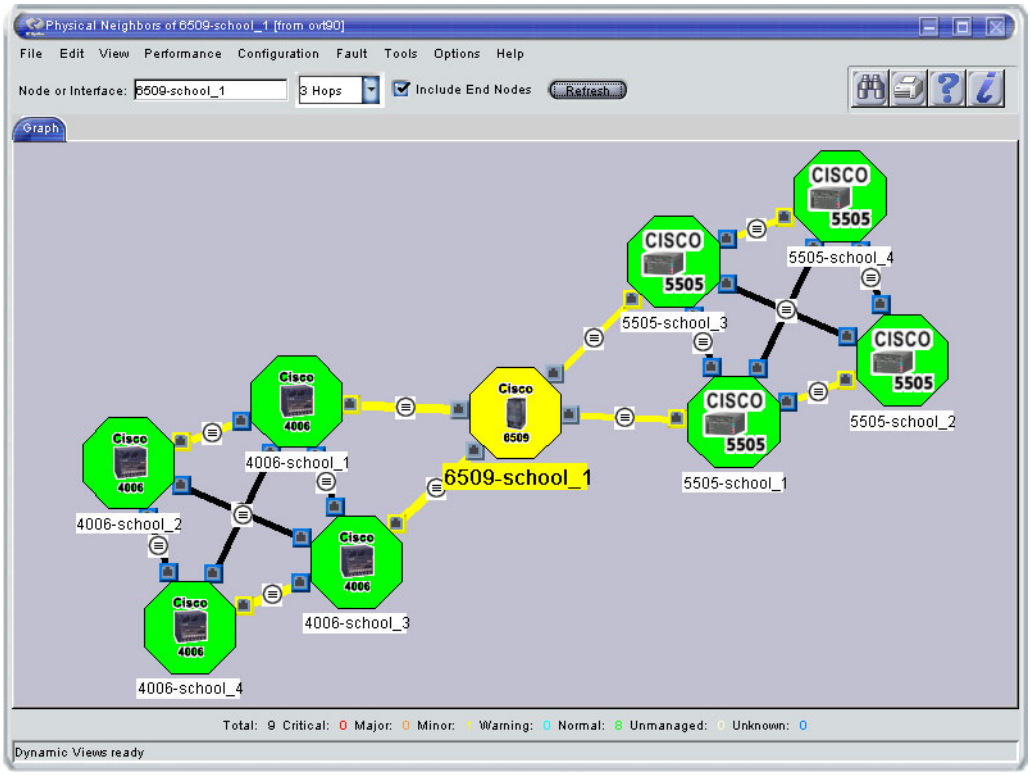
    e. Confirm removal by reloading the browser window.

15. Configure the zones according to the overlap suggested in the module. Delete Zone 3 and add 10.96.26.1 to Zone 1 AND Zone 2.

Answer:



If 10.96.26.1 is added to Zones 1 and 2, the zones will overlap and discovery and views show the original configuration (all nodes connected, VLANs combined). However discovery progress still proceeds one zone at a time.



Lab G Zone Discovery Review Questions:

1. Describe the major differences between the zone configuration and the affect each had on the final results for topology?

   Answer:

With the zones, nodes were discovered. However if no zones overlap, the relationship between zones is not discovered. Therefore, the Node View shows disconnected segments and the VLAN view shows split VLANs (two VLAN 5's with 4 nodes each rather than one VLAN 5 with 8 nodes).

Summary

1. Zone configuration requires insight and knowledge regarding the connectivity of the network.
2. Care must be taken in defining the specific devices that participate in a particular zone or zones.
3. The proper behavior for collecting and hence presenting information regarding Meshed, Switched VLANS eliminates duplication of data across zones.

Chapter 5, Scaling netmon Discovery and Polling

1. How could you use `xnmtopoconf` to unmanage all end nodes currently discovered?

   Answer:

   Write a filter that excludes all connector devices (not routers and not switches, etc). This will pass any end node. Use the command `xnmtopoconf -unmanage` *`MS_name filter_name`*.

2. How could you use `xnmtopoconf` to continuously unmanage all end nodes that get discovered?

   Answer:

   Place the `xnmtopoconf` command in a `cron` job or Scheduler list.

3. How could you use `ovautoifmgr` to continuously unmanage all end nodes that get discovered?

   Answer:

   Use the same filter definition and place it in the `ovautoifmgr.conf` file as the qualifying node filter. Leave out the other filters. Place the command `ovautoifmgr` in a cron job or Scheduler list.

4. Which tool would you use to unmanage all switch ports connected to end nodes? Why?

   Answer:

   You need to use `ovautoifmgr` for this so that you can specify the ports based on what they connect to. Specify switches in your qualifying node filter, all interfaces in the interface filter, and the end nodes filter used above in the end node filter position in your `ovautoifmgr.conf` file.

5. Run `resolveNames.ovpl` to see whether any systems in your current topology do not resolve.

   Answer:

   Execute the command `resolveNames.ovpl` and note the results. Run `resolveNames.ovpl -e` to scan the Binary Event Store. (***resolveNames.ovpl -e takes a very long time on Windows after the demo has been run and removed. It works fine on a clean Windows system. You may want to completely remove the NNM databases and restart this lab.)

   The tool is in the `/opt/OV/support` (UNIX) or *`install_dir`*`\support` (Windows) directory.

6. Review the list of names not currently looked up with the command `snmpnolookupconf -dumpCache`.

   Answer:

   Run the command and note the results. Nodes which are not resolvable (from `resolveNames.ovpl`) may not be excluded from lookup since `netmon` only places segment names, IPX names, and names derived from MAC addresses in the `snmpnolookup` cache.

Chapter 8, Active Problem Analyzer

Setting APA As Your Default Poller

1. Click on the Polling/Analysis Summary tab of NNM's Home Base. You'll note that all the numbers are 0 in this summary.
2. Change the default poller for your system from `netmon` to APA.

   Answer:

   ```
   ovet_apaConfig.ovpl -enable APAPolling
   ```

3. Look at the Polling/Analysis Summary again in NNM's Home Base. Within 5 minutes, you should start to see numbers change in this summary.

APA Demand Polling

CAUTION Successful execution of this lab depends on following the directions given by your instructor AT THE TIME those instructions are issued.

Directions

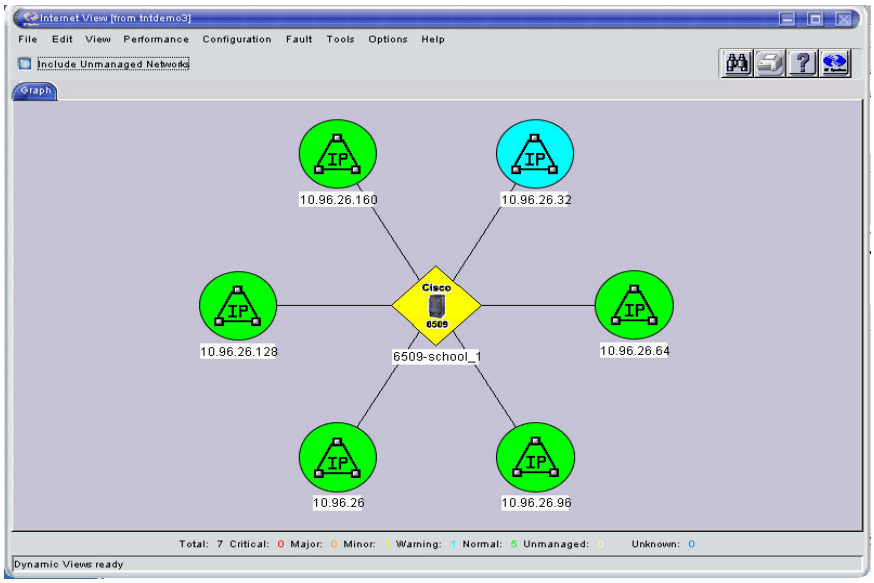
1. Change your working directory to `$OV_CONTRIB/NNM7labs/Lab_deploy`:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_deploy
   ```

2. When the instructor informs you to proceed, execute the setup script:

   ```
   setupLab_deploy.ovpl
   ```

3. When the setup script completes start Home Base. Launch the Internet View.
4. Expand the groups and locate any symbol that may be unmanaged or unknown. If unmanaged, select it, and use the menu `Edit:Manage`.
5. The map should appear as shown.



6. Commence the Extended Topology discovery. Use the Home Base Discovery tab or execute:

   ```
   etrestart.ovpl -verbose
   ```

7. From the NNM Home Base, launch a Node View. Select “All” for the Show Nodes field and “Normal” for the Status >= field, then click the Refresh button. Note the status of the nodes in the resulting view.
8. When the instructor gives you the go-ahead, right click on the 6509-school_1 node. Select “APA Status Poll” from the resulting pop-up menu. Wait for the APA status poll to complete in the resulting APA Demand Poll window.
9. Close the APA Demand Poll window and return to the Node View window.
10. Observe the status of the 6509-school_1 node.
11. When the instructor gives you the go-ahead, repeat the steps to obtain a new status for the 6509-school_1 node.

Chapter 9, Configuring Extended Topology Discovery of OSPF

Lab A: OSPF Full Discovery

Objectives:

Gain hands-on experience with OSPF discovery and views.

At the completion of this exercise you will be able to:

- Execute the OSPF discovery
- Properly configure OSPF discovery configuration according to defined limits
- Recognize and troubleshoot common OSPF discovery problems

CAUTION Successful execution of this lab depends on following the directions given by your instructor AT THE TIME those instructions are issued.

Assumption:

- Student has previously executed the `$OV_CONTRIB/NNM7labs/initialsetup.ovpl` script.

Directions

1. `cd $OV_CONTRIB/NNM7labs/Lab_ospf`
2. When the instructor informs you to proceed, execute the OSPF Lab setup script:

   `setupLab_ospf.ovpl`
3. When the setup script completes, start the Extended Topology discovery. From the Home Base Discovery tab, [`Initiate Full Discovery Now`] or execute:

   `etrestart.ovpl -verbose`

NOTE Although we restart Extended Topology processes, we do not use the Extended Topology level 2 views except for the OSPF view throughout the OSPF lab.

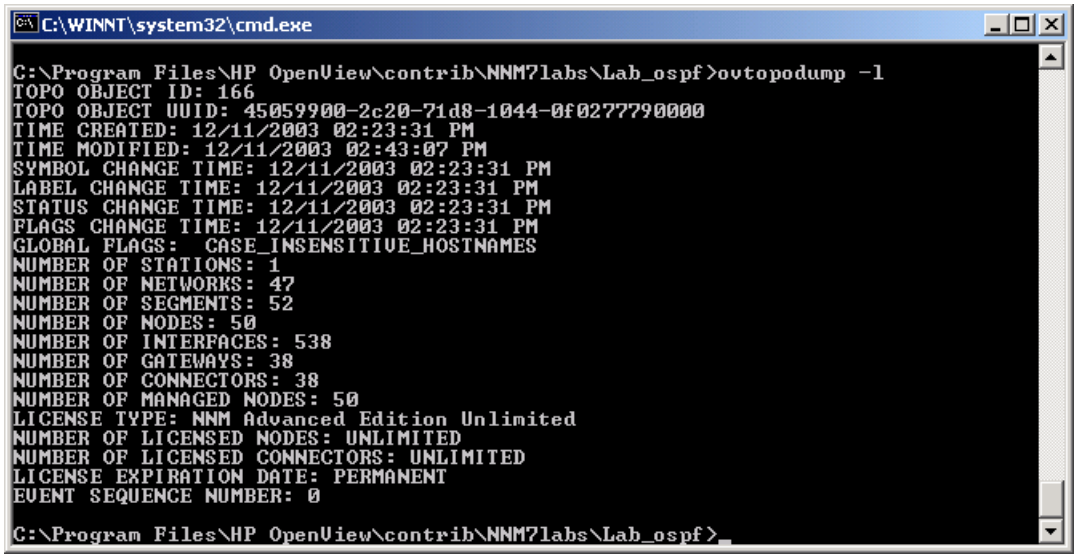
4. Confirm availability of NNM Extended Topology views. Load the `VLANs view` using Home Base.

NOTE You should see an empty VLAN table. If you get a message indicating: "`NNM Extended Topology is not currently available,`" you must either re-execute `etrestart.ovpl` or wait for it to complete.

5. Confirm the full OSPF topology has been properly loaded into NNM, execute:

   `ovtopodump -l`

   You should see the following information:



6. Edit the OSPF configuration file to confirm correct designation of the OSPF Seed router:

   `vi $OV_CONF/nnmet/Ospf.cfg`

   This file contains the OSPF-specific discovery configuration. Of particular importance is the SEED {IP-ADDRESS} designation (at least one is required) AND the INCLUDE and EXCLUDE statements.

   For LAB A (Full OSPF Discovery), ensure the `SEED IP_ADDRESS =10.96.27.32`.

   Next ensure that ALL the `INCLUDE` and `EXCLUDE` lines are commented out (using the "#" character). Save and exit the file editor.

   The `Ospf.cfg` file should look similar to:

```
=========================================================Start
#The next line will use the seed 10.96.27.32 for the start of the discovery.
SEED  10.96.27.32
#
# The next line will discover areas 0,1,2,3,4,7,9
# INCLUDE 0.0.0.0 0.0.0.1 0.0.0.2 0.0.0.3 0.0.0.4 0.0.0.7 0.0.0.9
#
# The next line will exclude areas 99 and 100 from the discovery.
# EXCLUDE 0.0.0.99 0.0.0.100
=========================================================END
```

7. Execute the OSPF discovery:

   `ospfdis.ovpl`

   You will see the following:

```
#> SEED is 10.96.27.32
INCLUDE list is
EXCLUDE list is
```

8. While the `ospfdis.ovpl` discovery process is still running, examine the contents of the OSPF data base folder:

```
#> ll $OV_DB/ospf
total 0
-rw-rw-r--   1 root       sys             0 Mar 18 09:49 ospfdis.lock
```

```
-rw-rw-r--   1 root       sys             0 Mar 18 09:49 ospfdis.tmp
```

The `ospfdis.lock` and `ospfdis.tmp` files indicate an active `ospfdis.ovpl` process. No final OSPF data is available for the OSPF view.

9. Confirm the discovery process completed correctly, showing 46 paths and 1 error, indicated by the output in the shell:

```
#> OSPF discovery results, 46 paths found, 1 errors found.
See $OV_PRIV_LOG/ospfdis.err for any errors reported.
```

NOTE You should have 50 paths found. Please inform the instructor if you do not! (You may have to re-run the `setupLab_ospf.ovpl` for Lab1, and/or re-run the `nmdemandpoll` tool and/or re-execute `$OV_BIN/etrestart.ovpl -verbose`.

10. View the contents of `$OV_PRIV_LOG/ospfdis.err` file. We will return to this file later.
11. Once the OSPF discovery process completes, view the contents of the OSPF database folder:

```
#> ll $OV_DB/ospf
total 14
-rw-rw-r--   1 root       sys          6572 Mar 18 09:51 ospfdis.data
```

12. The `ospfdis.data` file contains the OSPF the connectivity data/references database used to generate the OSPF view. View the contents of this file (`more`, `cat` ,etc.)
`more $OV_DB/ospf/ospfdis.data`

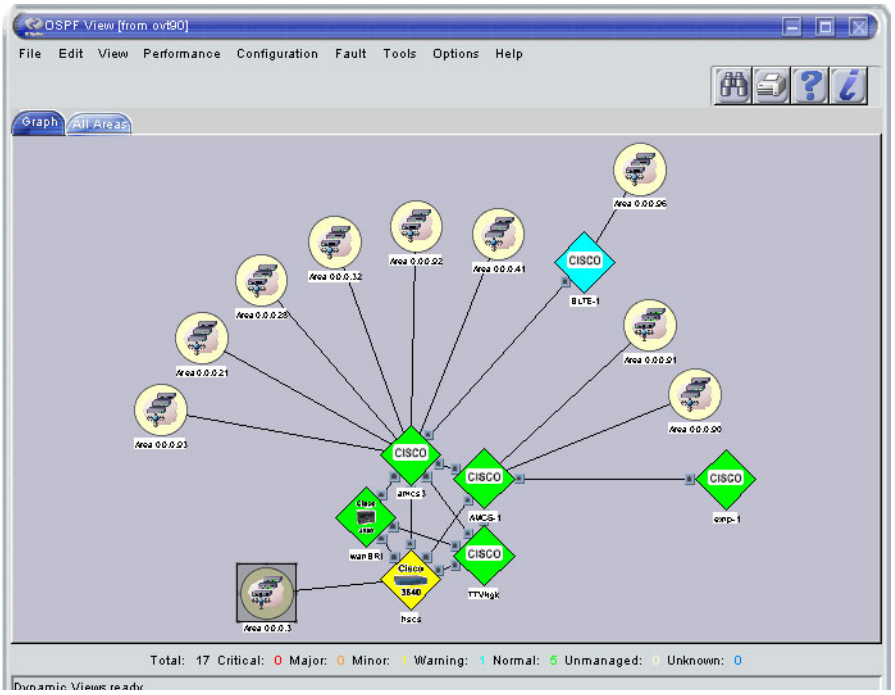
Contents of `$OV_DB/ospf/ospfdis.data`

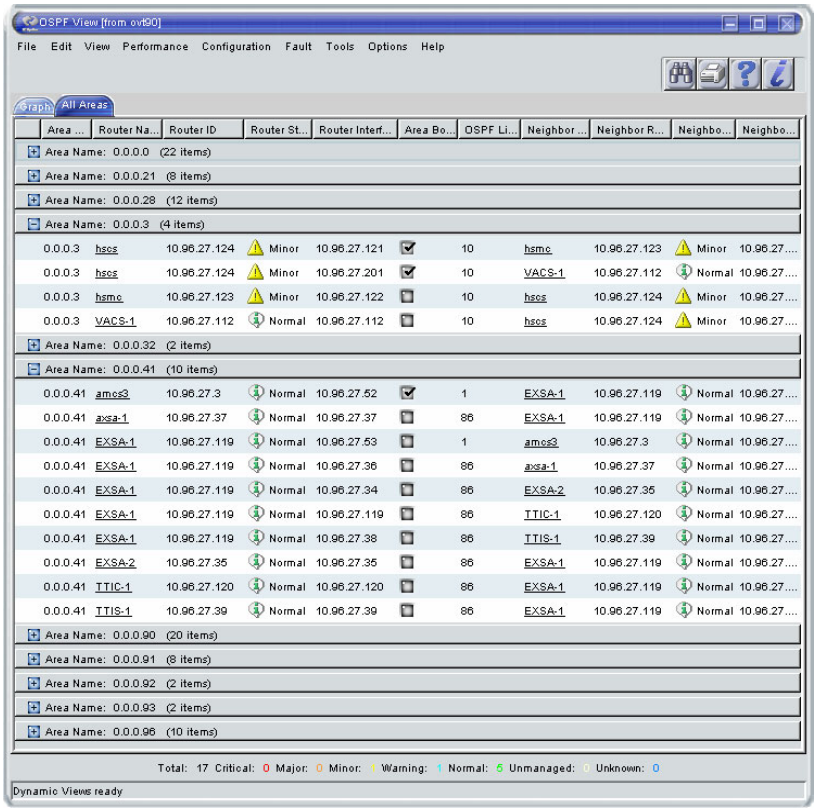
```
=====================================================Start
#File format
#nodeId sysname routerId abrvalue area ip nodeId sysname routerId abrvalue ip
#abrvalue is 0 for normal, 1 for area border router
#2 for autonomous border router, 3 for both 1 and 2
#
#area of seed 0.0.0.0
#seed is 10.96.27.32
#
524 AMCS-1 10.96.27.32 3 0.0.0.0 10.96.27.2 516 amcs3 10.96.27.3 3 10.96.27.3
524 AMCS-1 10.96.27.32 3 0.0.0.0 10.96.27.32 522 TTVNgk 10.96.27.24 0 10.96.27.24
…continues
=====================================================End
```

NOTE The `nodeId` reference is to the NNM object database (`ovobjprint -o nodeId`).

13. View the fully discovered OSPF topology using Home Base to launch the OSPF view.

You should see the backbone area 0.0.0.0 with 10 area symbols as shown:





14. Save this view. Select `File:Save`. Name the view `lab1afullOSPF.xml`.

15. Navigate the views and drill into each area. Note on paper the Area Names (such as 0.0.0.4, 0.0.94, etc). Take note of the Area Border routers. There are 4: AMCS-1, amcs3, BLTE-1, and hscs.

    Verify this.

Summary

Significant points about the lab thus far:

1. This is the full `OSPF` lab network that has been simulated from an actual customer network.
2. The `$OV_CONF/nnmet/Ospf.cfg` file was configured for the proper SEED IP-ADDRESS. YOU MUST IDENTIFY THE PROPER OSPF SEED ADDRESS FOR YOUR NETWORK.
3. We omitted any discovery restrictions by NOT using either the `INCLUDE` OR the `EXCLUDE` statement.
4. OSPF discovered data is contained in the text file `$OV_DB/ospf/ospfdis.data`.

Lab A: Full OSPF Discovery Questions:

1. What is the significance of the OSPF BACKBONE Designation?

   Answer:

   An OSPF backbone is responsible for distributing routing information between areas. It consists of all Area Border Routers, networks not wholly contained in any area, and their attached routers.

2. How is the Backbone Area Designated?

   Answer:

   It is designated as area 0.0.0.0.

3. Area 0 contains seven routers (AMCS-1, amcs3, BLTE-1, exnp-1, hscs, TTVNgk, and wanBRI). Area 96 contains 5 routers: BLTE-1, SRDO-1, SDFL-1, VATE-1 and SDBA-1.

   What router(s) does SDBA-1 know exists?

   Answer:

   BLTE-1, SRDO-1, SDFL-1, VATE-1 and SDBA-1

   What routers does BLTE-1 know exists?

   Answer:

   SRDO-1, SDFL-1, VATE-1, SDBA-1 and the Area Border Router: amcs3.

   What is the significance of “known routers” in this context?

   Answer:

   The ABRs separate the areas so that router updates contain only information needed for that area, in our example, the routers SRDO-1, SDFL-1, VATE-1 and SDBA-1 receive information from the ABR BLTE-1 that is required within Area 96 (need to know basis).

4. List the routers that are (hint, read the `ospfdis.data` file):

   a. Area Border Routers:

      Answer:

      AMCS-1, amcs3, hscs, BLTE-1

   b. Autonomous Border Routers:

      Answer:

      AXDA-1, amhh-1, AXWE-1, AXEP-1, amdl-a, axsa-1, TTIA-1, TTID-1, exau-1, TTIU-1,

   c. Both Autonomous and Area Border Routers:

Answer:

AMCS-1, amcs3, hscs, BLTE-1

You may proceed to the next lab at your own pace.

Lab B: OSPF Limited Discovery

Objectives:

Gain hands-on experience with OSPF Limited discovery and views.

At the completion of this module you will be able to:

- Execute a limited OSPF discovery using INCLUDE and EXCLUDE statements.

Assumption:

- Student has previously successfully completed the Full OSPF Discovery Lab.

Directions

1. Edit the `$OV_CONF/nnmet/Ospfg.cfg` file and use either `INCLUDE` or `EXCLUDE` statement to limit the OSPF discovery to areas *0, 3, 90, 91, 92 and 96*:

   ```
   vi $OV_CONF/nnmet/Ospf.cfg
   ```

   For LAB B (Limited OSPF Discovery), ensure the `SEED IP_ADDRESS =10.96.27.32` and that you use ONE of the `INCLUDE` OR `EXCLUDE` statements. Save and exit the file editor.

   The `Ospf.cfg` file should look similar to:

   ```
   ====================================================Start
   #The next line will use the seed 10.96.27.32 for the start of the discovery.
   SEED  10.96.27.32
   #
   #The next line will discover areas 0,3,90,91,92,96
   INCLUDE 0.0.0.3 0.0.0.90 0.0.0.91 0.0.0.92 0.0.0.96
   #
   # The next line will exclude areas 99 and 100 from the discovery.
   # EXCLUDE 0.0.0.99 0.0.0.100
   ====================================================END
   ```

2. Save the current `ospfdis.data` file:

   ```
   cd $OV_DB/ospf
   ```

   Copy the current `ospfdis.data` file to `ospfdis.data.lab1`.

3. Execute the limited OSPF discovery:

   ```
   ospfdis.ovpl
   ```

   You will see the following:

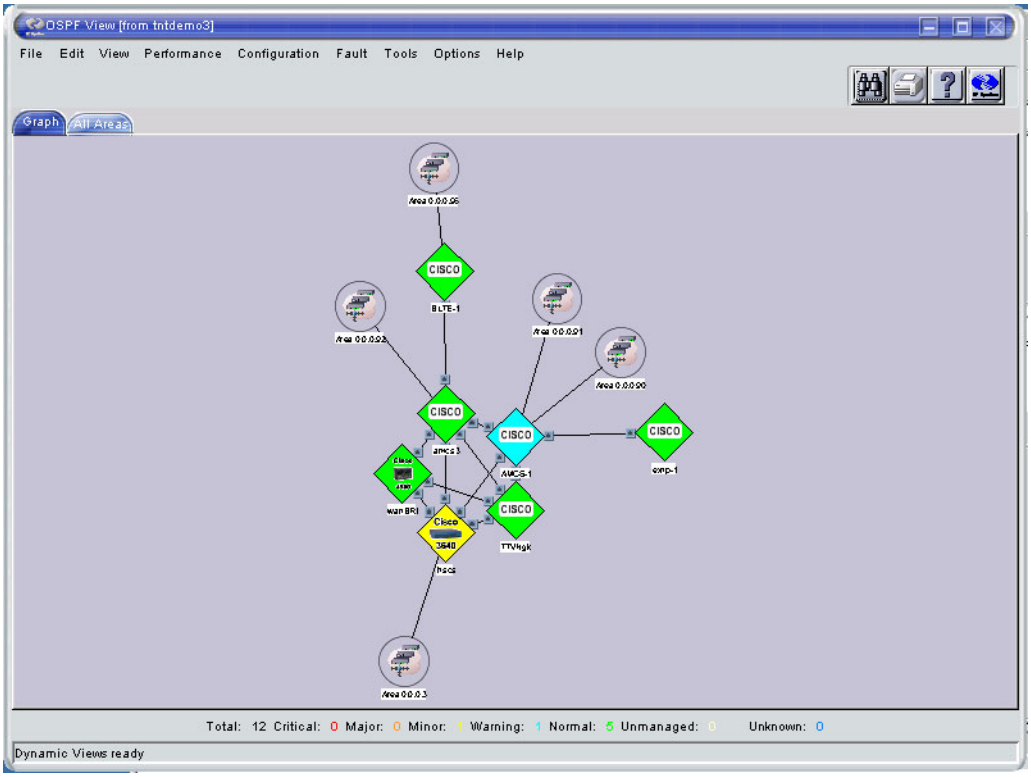
   ```
   #> ospfdis.ovpl
   SEED is 10.96.27.32
   INCLUDE list is 0.0.0.3 0.0.0.90 0.0.0.91 0.0.0.92 0.0.0.96
   # EXCLUDE list is
   OSPF discovery results, 33 paths found, 1 errors found.
   See /var/opt/OV/log/ospfdis.err for any errors reported.
   ```

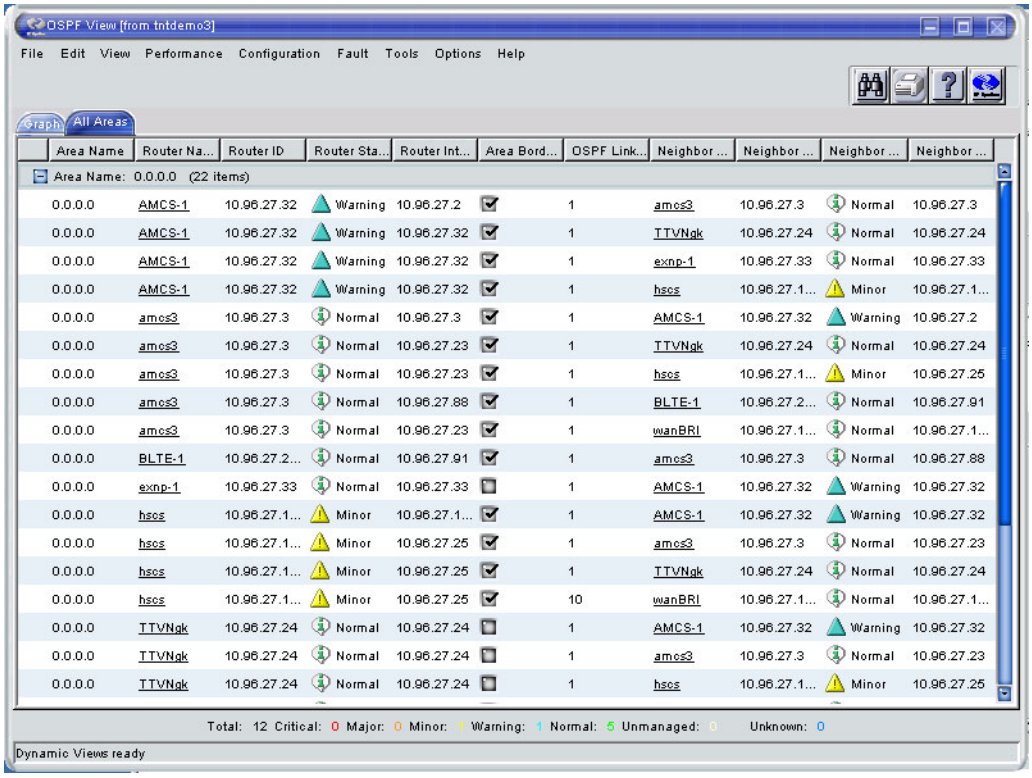
4. List the contents of the OSPF database directory:
   `ll $OV_DB/ospf`
   <=># ll $OV_DB/ospf
   total 24
   -rw-rw-r-- 1 root sys 4262 Mar 18 14:20 ospfdis.data
   -rw-rw-r-- 1 root sys 6572 Mar 18 09:51 ospfdis.old

   The most recent discovery is `ospfdis.data`, the one just prior to the current is the

`ospfdis.old`

5. Compare the results between the 1st and 2nd (3rd etc) OSPF discovery files in `$OV_DB/ospf`. BE CERTAIN TO COPY THE `ospfdis.data` files each time you run the OSPF discovery. (You will need these copies to compare the results) and label them Lab-1Aospfull-1, Lab-1Bospflimited1, etc…).
6. Run at least two separate OSPF discoveries using an `INCLUDE` statement then an `EXCLUDE` statement (save the results!). Execute the OSPF discovery, save the data file, examine the view and then reconfigure the `Ospfg.cfg` file for `EXCLUDE` and re-run `ospfdis.cfg`.
7. Compare the current `ospfdis.data` file with the one just prior (Limited Discovery compared to the Full Discovery).
8. The results of the limited discovery should show something similar to:





LAB B Limited OSPF Discovery Questions:

1. How can you limit OSPF Discovery?

   Answer:

   Use of either the `EXCLUDE` or `INCLUDE` statement in `$OV_CONF/nnmet/Ospf.cfg` file. Unmanage a device in NNM.

2. When would you use the INCLUDE statement?

   Answer:

   When the total number of OSPF Areas is LESS than the total number of areas that will be discovered.

3. When would you use the EXCLUDE statement?

   Answer:

   When the total number of OSPF Areas to exclude is less than the total number of areas to be discovered.

4. Must you INCLUDE area 0.0.0.0?

   Answer:

   No. Area 0.0.0.0 (backbone) is always included in an OSPF discovery.

5. What are the results when you do not explicitly `INCLUDE` the backbone area?

   Answer:

   Area 0.0.0.0 (Backbone) is included by default.

6. What is the result when you EXPLICITLY `EXCLUDE` the backbone (area 0.0.0.0)?

Answer:

The ospfdis.ovpl process complains:

```
Can't EXCLUDE the OSPF area 0.0.0.0 of SEED 10.96.27.32 in
/etc/opt/OV/share/conf/nnmet/Ospf.cfg.
```

7. Compare the results between the FULL OSPF discovery and the Limited (`INCLUDE`) OSPF Discovery.

   What differences are visible in the TABLE for OSPF Area 0 presented in the OSPF View? Explain:

   Answer:

   None, the table shows the same information since Area 0.0.0.0 is discovered as before.

STOP

Please wait for the instructor before proceeding.

(Optional) Lab C: OSPF Partial Discovery (incomplete SNMP access)

Objectives:

Explore the results of running the OSPF discovery when a device fails to respond due to improper SNMP community names and/or device is otherwise not available.

At the completion of this module you will be able to:

- Understand the importance and impact of SNMP Community Strings regarding OSPF discovery.

CAUTION Successful execution of this lab depends on following the directions given by your instructor AT THE TIME those instructions are issued.

Do not proceed until your Instructor has said to continue.

Assumptions:

- Student has previously successfully completed the Limited OSPF Discovery Lab.

Directions

1. Change your working directory to `$OV_CONTRIB/NNM7labs/Lab_ospf`:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_ospf
   ```

2. Replace the modified `Ospf.cfg` file with the original:

   ```
   cp –p $OV_CONTRIB/NNM7labs/Lab_ospf/configFiles/Ospf.cfg
   $OV_CONF/nnmet/Ospf.cfg
   ```

3. When the instructor has re-configured the MIMIC Simulation Server and says to continue, execute a new OSPF discovery:

   ```
   ospfdis.ovpl
   ```

   You will see the following:

   ```
   #> ospfdis.ovpl
   SEED is 10.96.27.32
   INCLUDE list is
   EXCLUDE list is
   OSPF discovery results, 20 paths found, 4 errors found.
   See $OV_PRIV_LOG/ospfdis.err for any errors reported.
   ```

   If you do not get these results, notify the instructor for assistance.

4. View error log file. What is the root cause of the error? HINT: run `etrestart.ovpl`.

   Answer:

   There is an SNMP access failure to the device AMCS3; this is the root cause.

5. View the resulting OSPF dynamic view. [Tip: if you kept the prior web browser active, just click reload | refresh].

   What is different?

   What device(s) and/or areas is(are) no longer available?

   What OSPF area(s) is(are) missing?

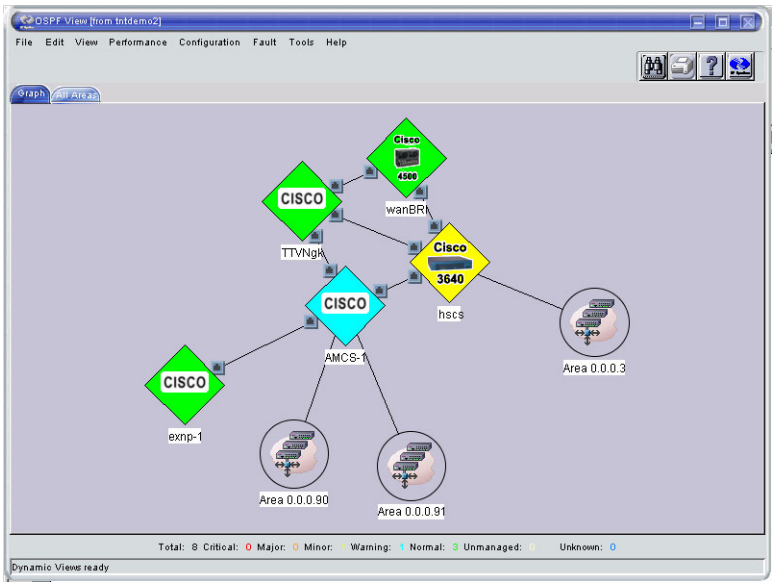
6. How can you tell by reading the `$OV_BIN/ospf/ospfdis.data` file?
   Hint: notice there are two files:

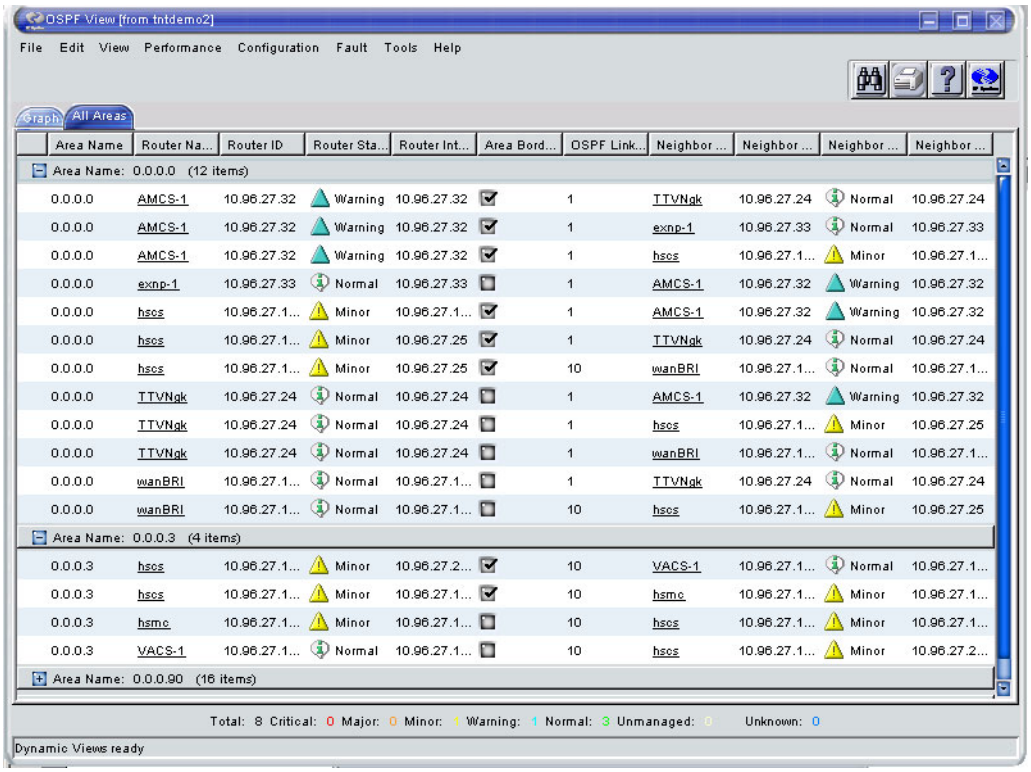
```
#> ll $OV_DB/ospf
total 16
-rw-r--r--   1 root       sys            2899 Feb  6 16:14 ospfdis.data <<latest discovery
-rw-r--r--   1 root       sys            4260 Feb  6 15:51 ospfdis.old  <<previous discovery
```

   Can you use the contents to help discover what the differences are?

7. The expected results for the OSPF view are shown:

   OSPF Partial Discovery (snmp)





Summary

NNM Extended Topology OSPF discovery relies on a “well known” SEED Address and snmp access to neighboring devices within the OSPF environment for proper and complete discovery. When one or more devices fail to respond, errors are logged and the resulting view is likely incomplete.

STOP

Please wait for the instructor before proceeding.

(Optional) Lab D: OSPF Partial Discovery (Unmanaged Device)

Objectives:

Explore the results of running the OSPF discovery when a device, which should be included in the discovery process, is not managed by NNM or is otherwise not in topology.

At the completion of this module you will be able to:

- Understand the importance and impact of having all relevant devices in the NNM topology and in the "managed" state.

CAUTION Successful execution of this lab depends on following the directions given by your instructor AT THE TIME those instructions are issued.

Do not proceed until your Instructor has said to continue.

Assumptions:

- Student has previously successfully completed the Partial (no snmp access) OSPF Discovery Lab.

Directions

1. Change your working directory to `$OV_CONTRIB/NNM7labs/Lab_ospf`:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_ospf
   ```

2. Using either an `ovw` map or the Internet view, locate and UNMANAGE the device amcs3.
3. When the instructor has re-configured the MIMIC Simulation Server and given the "go-ahead", execute a new OSPF discovery:

   ```
   ospfdis.ovpl
   ```

   You will see the following:

   ```
   #> ospfdis.ovpl
   SEED is 10.96.27.32
   INCLUDE list is
   EXCLUDE list is
   OSPF discovery results, 20 paths found, 0 errors found.
   See $OV_PRIV_LOG/ospfdis.err for any errors reported.
   ```

   If you do not get these results, notify the instructor for assistance.
4. What does the error log show?

   Answer:

   Nothing, since there are no errors.
5. View the resulting OSPF dynamic view. [Tip: if you kept the prior web browser active, just click reload or refresh.]

   What is different?

   What device(s) and/or areas is(are) no longer available?

   What OSPF area(s) is(are) missing?
6. How can you tell by reading the `$OV_DB/ospf/ospfdis.data` file?

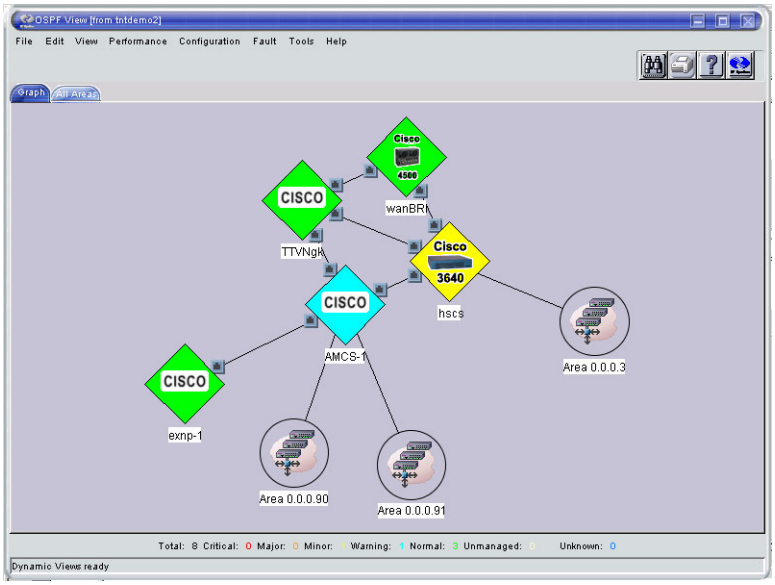
   Hint: notice there are two files:

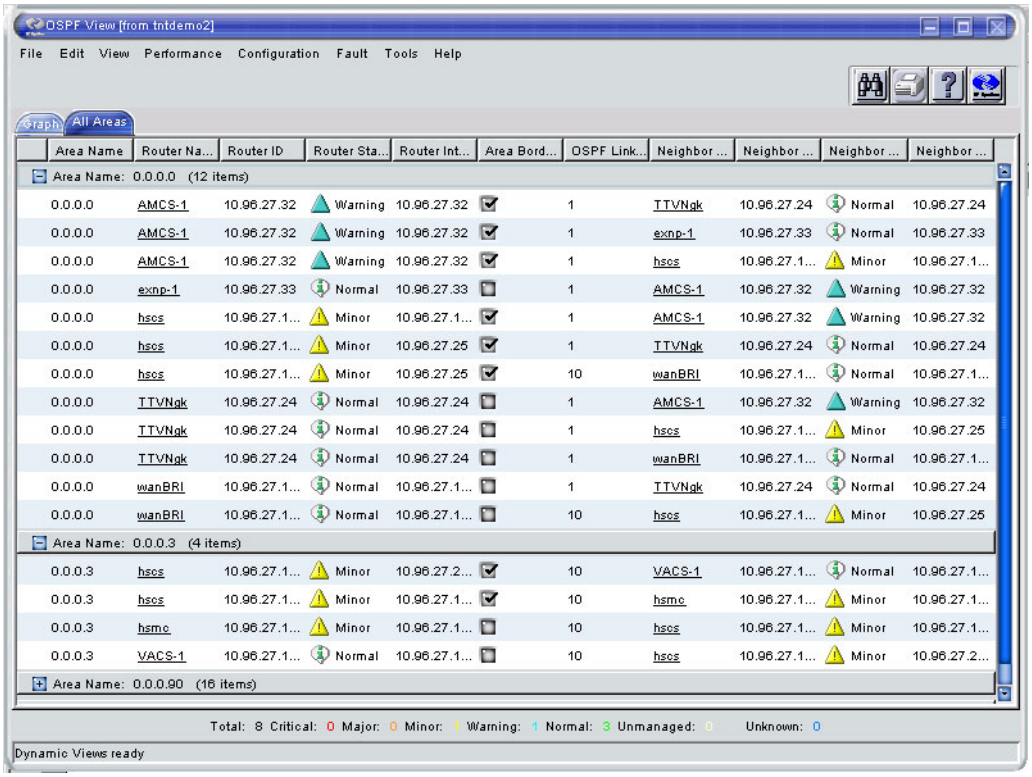
```
#> ll $OV_DB/ospf
total 16
-rw-rw-r--   1 root       sys            3279 Mar 18 16:58 ospfdis.data
-rw-rw-r--   1 root       sys            2900 Mar 18 16:12 ospfdis.old
```

Can you use the contents to help discover what the differences are?

The expected results for the OSPF view are shown:

OSPF View unmanaged device





Summary

Devices that are not reachable (down, no snmp response, etc) yet managed in topology will result in discovery errors. Devices that are not managed (or not in topology) will result in OSPF discovery completing without errors but perhaps without the desired result.

Chapter 10, Configuring Extended Topology Discovery of HSRP

HSRP Lab A: Full HSRP Discovery

Objectives:

- Recognize and understand NNM Extended Topology HSRP discovery.

At the completion of this module you will be able to:

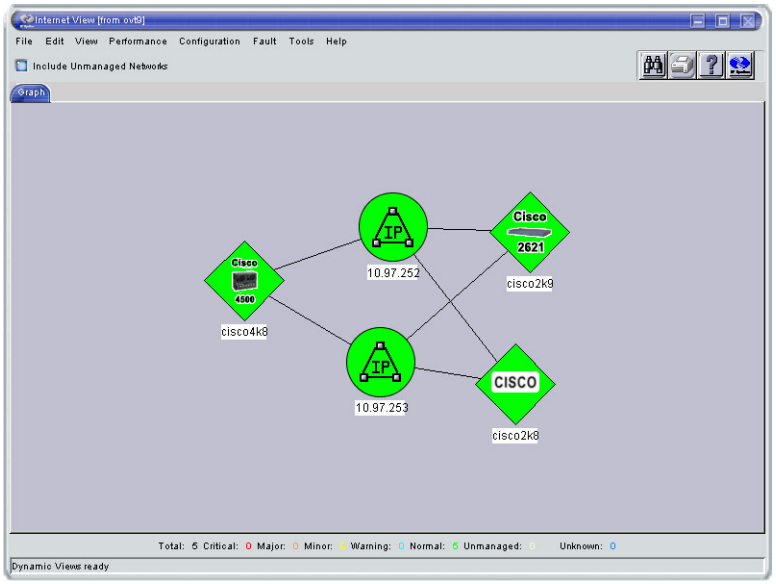
- Recognize a complete HSRP discovery.
- Observe what happens as HSRP states change.
- Increase familiarity with the NNM Extended Topology support tools.

CAUTION Successful execution of this lab depends on following the directions given by your instructor AT THE TIME those instructions are issued.

Directions

1. `cd $OV_CONTRIB/NNM7labs/Lab_hsrp`
2. When the instructor informs you to proceed, execute the setup script:

   `setupLab_hsrp.ovpl`
3. When the setup script completes start Home Base. Launch the Internet View.
4. The map should appear as shown.



5. Commence the Extended Topology discovery. Use the Home Base Discovery tab or execute:

   `etrestart.ovpl`

6. Since this topology only contains three nodes, the discovery will complete quickly. In order to view the results of the support commands such as

   `dumpAgentProgress.ovpl` *{agentName}*

   you must act quickly while discovery is in progress.

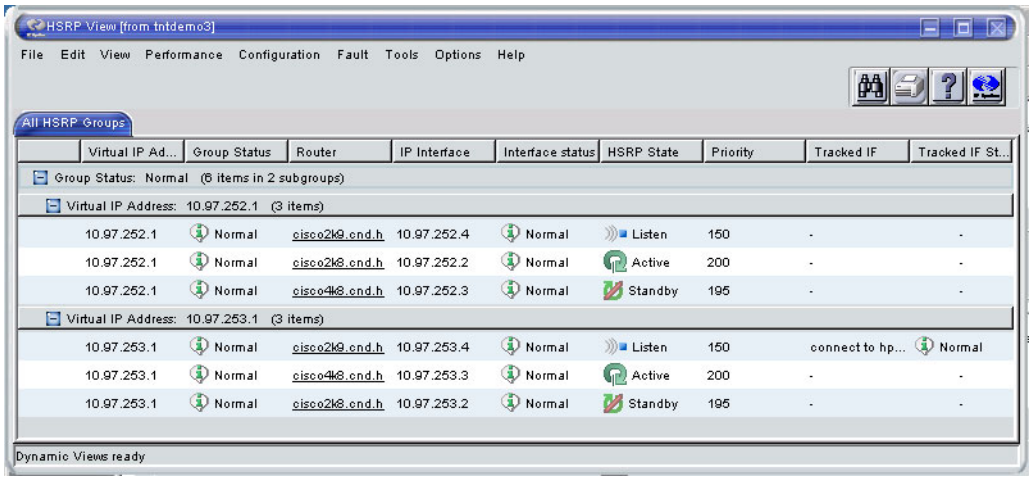
7. View the `dumpDiscoStatus.ovpl` results to see what agents are included in the discovery process.

8. Use the appropriate command to view the agents. Run this HSRP base discovery several times so that you have a chance to view the status of the agents.

9. After discovery has completed (`ovstatus -v ovet_disco` shows Awaiting next discovery) dump and review the contents of the database using:

   `ovet_topodump.ovpl -info`

10. Launch the HSRP dynamic view.



11. These views collectively show an accurate and correctly discovered HSRP environment. Each HSRP group has at least two interfaces of which one is "`active`" and the other is "`stand-by`" (or other HSRP state).

12. Setup up your desktop environment so that you can view the All Alarms browser and the HSRP View for both groups.

    Now execute: `$OV_CONTRIB/NNM7labs/Lab_hsrp/bin/sendStateChangeEvents.ovpl`

    This simulates state changes on the devices. You should see the active devices go down and their HSRP states become unknown. Then the standby devices become active, whereas the listen devices become the new standby devices.

13. When you have completed running the HSRP discovery and have analyzed each view, inform the instructor that you are done with this portion of the HSRP lab.

CAUTION Do not proceed to the next phase of the HSRP lab until the Instructor tells you to do so. The Instructor must configure teh MIMIC server for the next phase of the lab.

LAB A2: Automatic HSRP state change

Objectives:

- Recognize and understand how the HSRP view follows real-world state changes.

At the completion of this module you will be able to:

- Recognize automatic HSRP state changes.

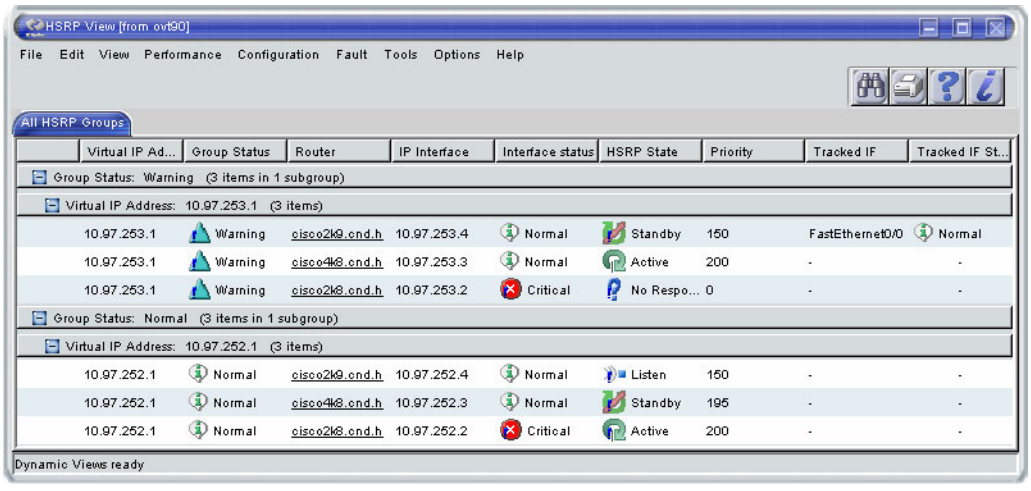
CAUTION Successful execution of this lab depends on following the directions given by your instructor AT THE TIME those instructions are issued.

Assumptions:

- Student has completed Lab A.

Directions

1. Open the HSRP view and expand to see the detail of all groups.
2. When your instructor has prepared the simulation server, do a demand poll of cisco2k8:
   a. Highlight the device cisco2k8 in the 10.97.252.1 group.
   b. Select `Fault:Network Connectivity:Poll Node`. Let this time out.
3. You should see devices 10.97.252.2 and 10.97.253.2 become Critical.
4. Within approximately 5 minutes, you should see the HSRP state changes similar to before:
   - The down device now becomes unknown.
   - The previous standby device is now active.
   - The previous listen device is now standby.
5. In addition, this time you should see the entire HSRP group change its overall status.



6. When all students have seen this, the instructor will simulate cisco2k8 returning to Normal.
7. Redo step 2 (repoll the critical node). It should now become normal.
8. Within approximately 5 minutes you should see the HSRP states return to their correct state.
9. Note the alarms in the All Alarms Browser. Select an alarm and see which views are available

under `Actions:Views`.

(Optional) Lab B: Partial HSRP Discovery

Objectives:

Recognize and understand a commonly occurring Extended Topology HSRP discovery issue.

At the completion of this module you will be able to:

- Recognize an incomplete HSRP discovery.
- Use the Extended Topology support tools to troubleshoot the HSRP discovery.

CAUTION Successful execution of this lab depends on following the directions given by your instructor AT THE TIME those instructions are issued.

Assumptions:

- Student has completed Lab A.

Directions

1. Change your working directory to `$OV_CONTRIB/NNM7labs/Lab_hsrp`.
2. WHEN THE INSTRUCTOR says to continue, commence the Extended Topology discovery. From Home Base, use the Discovery tab or execute:

   ```
   etrestart.ovpl -verbose
   ```

3. Since this lab only contains three nodes, the discovery will complete fairly quickly, though slower than the previous lab. In order to view the results of the support commands such as

   `dumpAgentProgress.ovpl` *`{agentName}`*

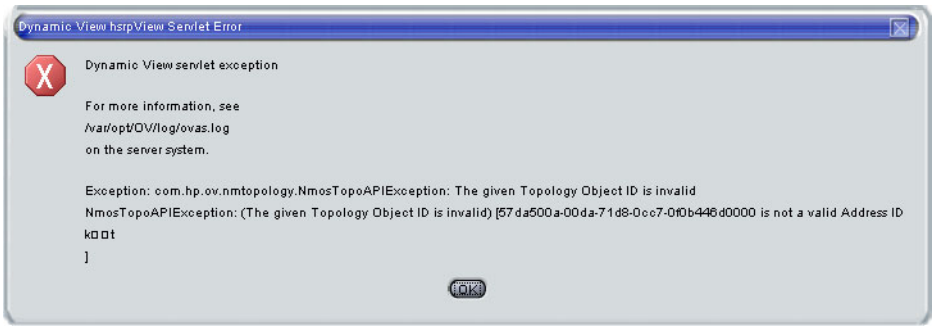
   you must act quickly while discovery is in progress.
4. View the `dumpDiscoStatus.ovpl` results to see what agents are included in the discovery process.
5. Use the appropriate command to view the agent progress. Run this HSRP discovery several times so that you have a chance to view the status of the various agents.
6. Can you determine which discovery agent(s) are having difficulty?
7. Which node(s) are causing the problem and why? Hint, from the Topology Summary, select `Doesn't Respond to SNMP`.
8. Dump and review the contents of the database using:

   ```
   $OV_BIN/ovet_topodump.ovpl -info
   ```

9. Tail the `$OV_PRIV_LOG/ovet_disco.log` during the HSRP discovery shows the following:

   ```
   In updateHsrpStatus
   In setHSRPStatus
   Cannot process HSRP group10.97.252.1 with fewer than 2 members
   ifOidListString is
   Could not process group 10.97.252.1 due to errors.
   In setHSRPStatus
   Cannot process HSRP group10.97.253.1 with fewer than 2 members
   ifOidListString is
   ```

```
Could not process group 10.97.253.1 due to errors.
```



10. Attempt an HSRP View from Home Base.
11. These views collectively show a commonly occurring problem with HSRP discovery and one that you should be able to recognize. Without appropriate SNMP access (and/or all devices within the HSRP group have been discovered by NNM AND managed) to the routers configured in an HSRP group, NNM Extended Topology will show only the partial configuration.

Chapter 11, Introduction to Event Reduction

Objective:

Describe the concepts of event correlation and ECS.

Review Questions:

1. What is event correlation?

   Answer:

   Event correlation is the analysis of the relationship between events with the goal of generating fewer output events with higher information content.

2. What are the benefits of event correlation?

   Answer:

   - Reduce the number of events. Events can be "allowed through" based on previous, current or subsequent events. Contrast this with simple event filtering which does not compare events with other events.
   - Detect events by their signatures, where a signature typically consists of multiple, time-separated events possibly delayed by the network or delivered out of order.
   - Modify events to contain more information.
   - Create new events. A problem may have several symptoms which generate events. By analyzing the relationships between events the root cause or problem can be identified.
   - Events can be sorted. For example, based upon latency in the network some events may take longer to arrive than others.

3. What is ECS?

   Answer:

   ECS or Event Correlation Services is an OpenView product integrated with NNM's pmd to event correlation capabilities. It is delivered as two distinct products: ECS Engine and ECS Designer.

4. What is an ECS correlation?

   Answer:

   An ECS correlation is a set of rules to control the flow of events. The correlation rules consist of interconnected logic blocks known as "correlation nodes". Each "correlation node" has attributes and values that can be defined to control its behavior on how the events are handled.

5. What are "Flows" and "Streams"?

   Answer:

   Processes listening to events in NNM can specify one of three event flows:

- RAW -The set of all events that come into NNM
- CORRELATED-The set of events coming out of all correlations
- ALL-The set of all RAW events plus all events potentially created by correlations.

The correlated event flow is further divided into streams. A "stream" is a distinct, logically independent, flow of events through an ECS engine. NNM uses only one called "default".

6. How can you improve on the results from event correlation?

   Answer:

   You can improve results by de-duplicating repeated events from the Alarm Browser.

7. How does Composer relate to ECS?

   Answer:

   Composer is a super-correlation within ECS which has an additional user interface and many modifyable parameters.

8. How many times does Composer evaluate each incoming event?

   Answer:

   Each event is evaluated only once by Composer.

9. What qualifying questions would you ask to decide whether a problem was amenable to a Composer solution?

   Answer:

   Can the problem be solved more easily by de-duplication or setting events to log only?

   Does the problem involve the relationship of events to other events, or could I just filter the Alarm Browser?

   Do I know how to identify the incoming events of interest?

   Can I determine how to match events from the same source?

   Do I know what I want to do when the events arrive?

10. List the event reduction mechanism in order of least load imposed on `pmd`.

    Answer:

    Adding a new ECS correlation adds the most overhead to `pmd`.

    Adding a Composer correlator has less impact on `pmd`.

    Setting an event to LogOnly or Ignore has the least load on `pmd`.

    De-duplication and Pattern Deletion run in `ovalarmsrv` and place no load on `pmd`.

Chapter 12, Configuring Event Correlation

Objective:

Manage the standard ECS correlations supplied with HP OpenView NNM.

Tip: You may choose to manually generate SNMP events for some exercises or you can use the lab scripts provided. This can easily be done from the command line, using either `snmptrap` or `ovevent`, combined with the following information:

Generic Trap Name	Number	Event ID
Cold Start	0	.1.3.6.1.6.3.1.1.5.1
Warm Start	1	.1.3.6.1.6.3.1.1.5.2
Link Down	2	.1.3.6.1.6.3.1.1.5.3
Link Up	3	.1.3.6.1.6.3.1.1.5.4
Authentication Failure	4	.1.3.6.1.6.3.1.1.5.5
EGP Neighbor Loss	5	.1.3.6.1.6.3.1.1.5.6

For example, to send a "Warm Start" to the local management station, use either:

```
snmptrap "" ".1.3.6.1.6.3.1.1.5" "" 1 0 0
```

or

```
ovevent "" .1.3.6.1.6.3.1.1.5.2
```

Examining the ConnectorDown Correlation

1. Examine the configurable items in this correlation. What are the parameters used by this correlation?

   Answer:

   - InputEventTypeList
   - NodeEvThenIfaceEv
   - MaxNodeStatusWait
   - ExtraAnalysis
   - ImportanceTimeout
   - NodeStatusEventTypeList
   - IfaceStatusEventTypeList
   - CorrelateOnNonSuppress

2. Which events are processed by this correlation?

Answer:

By looking at the InputEventTypeList parameter we can see that the correlation uses the OpenView enterprise events: OV_Node_Warning, OV_Node_Major, OV_Node_Up, OV_Node_Down, OV_IF_Up, OV_IF_Down, OV_Node_Marginal, OV_IF_Unknown, OV_Node_Unknown, OV_Node_Primary. (You can look up the event numbers in `Options:Event Configuration`.)

3. Which events are treated as primary events for interface cards?

Answer:

Look at the IfaceStatusEventTypeList variable to see OV_IF_Up and OV_IF_Down

Using and Configuring the PairWise correlation

1. Examine the default configuration of the PairWise correlation's parameters.

   a. Which event, "linkUp" or "linkDown", would be suppressed by this correlation? Which would be expected to occur first?

   Answer:

   "linkDown," which would be expected first, would be suppressed by the "linkUp" event.

   b. Which six events are "cancelled out" by Interface Up?

   Answer:

   Look in the `InputEventTypeListStringSources` field to see the currently configured events. You can look up the event numbers in `Options:Event Configuration`. Interface Up would cancel out Interface Down, Interface Testing, Interface Unknown, Interface Major, Interface Marginal, and Interface Warning if they were all enabled (Accept column).

2. Observe the action of the default configuration.

   a. Use `Options->Event Configuration` to modify the default configurations of the linkUp and linkDown SNMP events so that a message will be displayed in the Status Alarms category and display a popup message.

   Answer:

   Windows:

   1. Choose `Options->Event Configuration`.
   2. Select "snmpTraps" in the Enterprise Identification table.
   3. Select the event you would like to modify, SNMP_Link_Up or SNMP_Link_Down, then `Edit->Events->Modify`.
   4. Use the Event Message tab to have the message "Log and Display in Category: Status Alarms". Use the Actions tab to define Popup Window Text. Click on [`OK`].
   5. Remember to `File:Save` in the Event Configuration dialog window.

   UNIX:

   1. Choose `Options->Event Configuration`.
   2. Select "snmpTraps" in the Enterprises table.
   3. Select the event you would like to modify, SNMP_Link_Up or SNMP_Link_Down,

then Edit:Modify Event.

4. Change the Category drop down list to Status Alarms, add your Pop-up Notification and click on [OK].
5. Remember to File:Save in the Event Configuration dialog window.

b. Open the Status Alarms browser. Send a linkUp-LinkDown-linkUp sequence of events. Watch the Status Alarms browser carefully between each. You may use the tool $OV_CONTRIB/OVTraining/NNM3/linkupdown.ovpl or send the events from the command line. For example

```
snmptrap "" ".1.3.6.1.6.3.1.1.5" "" 3 0 0 or
ovevent "" .1.3.6.1.6.3.1.1.5.4

snmptrap "" ".1.3.6.1.6.3.1.1.5" "" 2 0 0 or
ovevent "" .1.3.6.1.6.3.1.1.5.3

snmptrap "" ".1.3.6.1.6.3.1.1.5" "" 3 0 0 or
ovevent "" .1.3.6.1.6.3.1.1.5.4
```

What did you expect to happen? What happened?

Delete the alarms when you are done to keep the next labs clear.

Answer:

The first LinkUp appears in the Alarm Browser. (Pop-up messages appear only if you have the native Alarm Browser running, not from the web Alarm Browser.)

The LinkDown appears in the Alarm Browser.

The second LinkUp appears in the Alarm Browser and cancels out the LinkDown.

c. Open the ECS Configuration interface (Options:Event Configuration, Edit:Event Correlation). Examine the parameters for the PairWise Correlation by selecting PairWise and choosing [Modify].

What is the current value of the "DeleteOrAcknowledge" parameter?

Answer:

The default value for "DeleteOrAcknowledge" is Delete.

d. Change ChildEventImmediateOutput to false. Repeat the link up and link down messages. What happens? Return the value to true when you finish. Delete the alarms when you are finished.

Answer:

The first linkUp generated a pop-up and message in the Status Alarms browser. The linkDown was held internally and did not generate a pop-up and a message. (Depending on the speed with which you are executing these labs, you may see an intermittent status alarm indicating that a link is repeatedly going up and down. It is not relevant to this lab.) The second linkUp generated a pop-up, correlated the linkDown and de-duplicated the first LinkUp.

e. Change the DeleteOrAcknowledge parameter so that the child event will get acknowledged. Remember to [Apply] your changes. Test this configuration by sending another sequence of linkUp-LinkDown-linkUp events while observing the Status Alarms browser between each event.

What happened? Delete the events when you are done. Return DeleteOrAcknowledge to its default.

Answer:

The first linkUp generated a pop-up and message in the Status Alarms browser. The linkDown generated a pop-up and a message. The second linkUp generated a pop-up, acknowledged the preceding linkDown and associated the preceding linkDown as a child/correlated event as well as de-duplicating the first LinkUp.

Observing Pattern Deletion

The Pattern Delete function helps handle pairs of events that would ordinarily be ignored by the Pair-Wise correlation. In this exercise, you will set up and use this function.

For this exercise, you may also use the standard SNMP Link Down (Event ID: .1.3.6.1.6.3.1.1.3) and Link Up (Event ID: .1.3.6.1.6.3.1.1.4) events. You will need to change their logging as well as the ECS definition for these events. As these events may occur in a normal setting, you may want to reset them after this lab.

1. Using the `Options:Event Configuration` GUI, configure a new pair of events under the OpenView enterprise.

   For example: `ChildTestEvent .1.3.6.1.4.1.11.2.17.1.0.2004`

   `ParentTestEvent .1.3.6.1.4.1.11.2.17.1.0.2005`

   Make sure that the event is displayed in one of the alarm categories. You may want to have a pop-up message and a specific severity.

   Issue each event using `ovevent` to be sure of its operation.

   Answer:

   From the Options:Event Configuration GUI, add a new alarm category and two new events under the OpenView Enterprise with the following attributes:

   - Alarm Category: Lab Alarms
   - Child Event:
     - — Name: ChildTestEvent
     - — Specific Trap Number: 2004
     - — Description: optional
     - — Log and Display in category: Lab Alarms
     - — Severity: Minor
     - — Event Log Message: Received Child Test Event
     - — Pop-up Window Message: Received Child Test Event
   - Parent Event:
     - — Name: ParentTestEvent
     - — Specific Trap Number: 2005
     - — Description: optional
     - — Log and Display in category: Lab Alarms
     - — Sevrity: Normal
     - — Event Log Message: Received Parent Test Event

— Pop-up Window Message: Received Parent Test Event

Save the event configuration and test the events by issuing the following commands from a command prompt:

```
ovevent hostname .1.3.6.1.4.1.11.2.17.1.0.2004
ovevent hostname .1.3.6.1.4.1.11.2.17.1.0.2005
```

where *hostname* is your NNM system name.

2. Using the ECS Configuration GUI, select the Pair-Wise correlation and make the following adjustments for the lab:

   Answer:

   Launch the ECS configuration GUI from the Event Configuration window.

   Select the PairWise correlation and click the Modify button.

   a. Modify the time window to 1 minute.

      Answer:

      Select the "PairedTimeWindow" parameter and click the View/Modify button.

      Change the current value to 1m (the default is 10m).

      Then select the `Close` button.

   b. Add the child and parent events to the table of InputEventTypeListStringSources.

      Answer:

      1. Select the InputEventTypeListStringSources parameter and click the View/Modify button.
      2. Click the `Add Row` button.
      3. In the first column, enter the Parent Test Event number, without the leading dot.
      4. Press Enter to accept the value.
      5. Repeat this in the second column for the Child Test Event number.

   c. You can use "agent-addr" for Key 1 and 0 for Key 2 for each event.

      Answer:

      1. In the third column (Parent String Source Key 1), enter "agent-addr" (quotes included).
      2. In the fourth column (Parent String Source Key 2), enter 0 (no quotes).
      3. Repeat this for columns five and six for the Child String Source keys.

   d. Be sure to set the line in the Accept column to true.

      Answer:

      In the seventh, and last, column (Accept (True/False)), set the value to true.

   e. Verify the table and select Close and then OK.

      Answer:

      1. Click the Verify Table button and make sure that there are no syntax errors.
      2. Correct any errors before continuing with the exercise.

3. Click the Close button to close the InputEventTypeListStringSources parameter window.
4. Click the OK button to close the PairWise correlation window.
5. Click the Apply button to accept the changes and reload the correlation.

3. Issue the events within the time window for the Pair-Wise correlation. This should result in the normal, or usual, operation of the correlation.

   Answer:

   Open the Lab Alarms category so that the events can be observed.

   Issue the following commands within 1 minute of each other.

   ```
   ovevent <hostname> .1.3.6.1.4.1.11.2.17.1.0.2004
   ovevent <hostname> .1.3.6.1.4.1.11.2.17.1.0.2005
   ```

   Since the events were issued within the time window, the arrival of the Parent event caused the Child event to be discarded along with the Parent.

4. Now issue the child event followed by the parent event outside the time window specified in the correlation. Note the results in the browser.

   Answer:

   The Child event appears in the browser. When the Parent event arrives it also shows up in the browser. This indicates that no correlation occurred when the events arrived outside the defined time window. ***Defect in NNM: Pattern Deletion does not clean up this pair on UNIX or Windows.

5. Finally, issue multiple occurrences of these events both within and outside the time window in the correlation. Note the impact in the browser.

6. Be sure to reset the time window in the Pair-Wise correlation after this lab.

   Answer:

   Because all of the PairWise Parent-Child definitions use the same time window, the value needs to be reset to the default of 10m. Otherwise, you may experience inconsistencies in the behavior of NNM during the rest of the course.

Using and Configuring the RepeatedEvent correlation

1. Configure the SNMP_EGP_Down event so that you will recognize it when it is received.

   a. Configure the event so that it will be displayed in the Status Alarms category and a pop-up message will be generated.

   b. Test your configuration by generating by executing:

      ```
      snmptrap "" ".1.3.6.1.6.3.1.1.5" "" 5 0 0
      ```

      or

      ```
      ovevent "" .1.3.6.1.6.3.1.1.5.6
      ```

2. Examine the existing, default, configuration of the RepeatedEvent correlation.

   a. Which SNMP protocol events are configured for processing by this correlation? Which specific OpenView events?

      Answer:

Looking at the InputEventTypeList, all the generic SNMP traps are configured, however they are not enabled. OpenView's OV_Node_Up, OV_PhysAddr_Mismatch, and OV_Node_Added are configured among others, however OV_Node_Up is not enabled.

b. Will any events be placed in the Alarm Browser, and then removed later?

Answer:

No, since CreateUpdateEvent is false by default, an UpdateEvent will not be generated and the original event will not be deleted.

c. What is the longest period of time during which reported events will be suppressed by this correlation? Is it possible to have a longer period of suppression without changing the RepeatedTimeWindow?

Answer:

10 minutes. Yes, when RollingWindow is true, events will be suppressed until the period between consecutive events is larger than the RepeatedTimeWindow. This means that if events occur frequently they could be suppressed indefinitely.

3. Modify the window for this correlation to 1 minute.

a. Select the RepeatedTimeWindow and change the current value to 1 minute.

b. Apply the change. What happens?

Answer:

Since RepeatedTimeWindow is a static parameter, the circuit must be reloaded to have the change applied.

c. Send 4 EGP_Neighbor_Loss events, 25 seconds apart. You may use the script $OV_CONTRIB/OVTraining/NNM3/egpLoss.ovpl. Which events are observed? Explain.

Answer:

All are observed, since EGP Neighbor Loss is not yet enabled in the InputEventTypeList.

4. Observe the use of this correlation.

a. In the InputEventTypeList, change the Enable Event Type to true for the EGP Neighbor Loss event.

b. Close the InputEventTypeList window and [Apply] the change. Was the correlation restarted? Explain.

Answer:

The correlation was not restarted since InputEventTypeList is a dynamic parameter.

c. Send 4 EGP_Neighbor_Loss events, 25 seconds apart.

- When does the number occur in the Correlation column of the Alarms browser?
- How many popups are generated?
- Look at the correlated events. How many are listed?

Answer:

The correlation appears when the second event is received. Only 2 popups are observed, relating to EGP Neighbor Losses #1 and #4. A final update event is not generated at the expiration of the window due to CreateUpdateEvent being set to false.

Three events are listed in Show Correlations: the first raw event and the two "child" events which were suppressed during the 1 minute window.

d. Change `CreateUpdateEvent` from `false` (the default) to `true`. Send 4 EGP_Neighbor_Loss events, 25 seconds apart.

- When does the number occur in the Correlation column of the Alarms browser?
- How many popups are generated?
- Look at the correlated events. How many are listed?

Answer:

The correlation appears when the second event is received. Three popups are generated. The first is for the initial EGP_Neighbor_Loss event. The second is for the final correlated event which represents the first three EGP_Neighbor_Loss events, all of which occurred within the 1 minute window. The last popup is for the fourth EGP_Neighbor_Loss, which occurred after the 1 minute window. This fourth event, of course, has started a new 1 minute window.

Four correlated items are listed:

1. The final, correlated (update) event, occurring 1 minute after the starting event, i.e. when the window expires. (Note the timestamps on the correlated events.)
2. The first raw event.
3. The two "child" events which were suppressed during the 1 minute window.

5. Change `CreateUpdateEvent` back to the default: `false`. Change the `RollingWindow` parameter from `false` to `true` and apply the changes.

a. Send 4 EGP_Neighbor_Loss events, 25 seconds apart, as before.

- How many popups did you observe? Why?
- What is different about the correlated events?
- Summarize the effect of the RollingWindow.

Answer:

There is only one popup, for the first EGP_Neighbor_Loss. Since each subsequent event occurs within 1 minute of the previous event, the rolling window configuration causes them all to be "child" or "suppressed" events.

b. Change `CreateUpdateEvent` back to `true`. Send 4 EGP_Neighbor_Loss events, 25 seconds apart, again.

- How many popups did you observe? Why?
- What is different about the correlated events?
- Summarize the effect of the RollingWindow.

Answer:

There are two popups. The first for the first EGP_Neighbor_Loss. The second for the update event that was created after the rolling window expired.

6. Return the values to their defaults: RollingWindow, false; CreateUpdateEvent: false; RepeatedTimeWindow: 10m.

De-Duplicating Events

Event de-duplication is preconfigured in NNM for a pre-defined set of events. In this exercise, you will expand that list to include a user-defined event.

1. (Optional) Using the Event Configuration GUI, create a new Alarm Category for this set of exercises.

   While this step is optional, it helps keep the focus of attention on the desired exercise.

2. Using the Event Configuration GUI, create a new event under the OpenView Enterprise. For example: `DeDupTestEvent .1.3.6.1.4.1.11.2.17.1.0.2003`. Direct the event to the new Alarm Category, and optionally add a pop-up message and whatever severity you desire.

3. Use `ovevent` to send a few of these events to your browser.

   Answer:

   ```
   ovevent "" .1.3.6.1.4.1.11.2.17.1.0.2003
   ```

4. Modify the `$OV_CONF/dedup.conf` file and add this event definition to the list of events to be de-duplicated.

   Answer:

   Add the following to the `dedup.conf` file:

   ```
   # Test of de-duplication of events

   <.1.3.6.1.4.1.11.2.17.1.0.2003, $r>
   ```

5. Stop and then re-start the `ovalarmsrv` service. This re-reads the `dedup.conf` file.

6. Re-issue a series of the test events. Note the effect in the Alarm Browser.

   Answer:

   The events roll up into a single event, with a count of the total number of events.

   Also, note the time stamps on these events. The time stamp visible in the Alarm Browser is the most recent occurrence of the event.

Using and Configuring the Scheduled Maintenance correlation

1. Define a Scheduled Maintenance correlation for the systems in your classroom.

   a. Add a row to the `OutageTimeSpecification` parameter table and create a time period starting roughly 5 minutes from now and lasting for 10 minutes.

      Answer:

      Modify the ScheduledMaintenance correlation. [`View/Modify`] the `OutageTimeSpecification` parameter table. Click on [`Add Row`]. Then add an appropriate entry for each table entry. For example, if it was now 10:00 a.m. Wednesday:

      Specification Name: “Lab”
      Year:"*"
      Month:"*"
      Month Day:"*"
      Week Day:3
      Hour:10
      Minute:5
      Outage Duration:10m

Click on [Verify Table] to the right, then [Close].

b. Modify the MaintenanceList parameter table to specify that the classroom hosts have a scheduled maintenance planned during the defined OutageTimeSpecification.

Answer:

[View/Modify] the MaintenanceList parameter table. [Add Row], then enter then add appropriate entries to the new row. For example, if the classroom hosts were in the range of 172.16.0.100-125 the entries would be:

Host Specification: 172.16.0.100-125
Outage Specification Name: "Lab"
Schedule Active: true

Verify the table, then close the MaintenanceList - Modify window.

c. Apply the changes. Was the correlation restarted? Explain.

Answer:

The correlation was not restarted, since it is not enabled by default.

d. Enable the ScheduledMaintenance correlation. You may want to reconfigure the Status Polling cycle of your management station to be about 30s.

- Your instructor will take one system offline.

- Experiment with generating events. You may also use the script $OV_CONTRIB/OVTraining/NNM3/maint.ovpl.

- Look at the correlated events for the scheduled maintenance

Chapter 13, Introduction to Composer Development

1. Start the Composer developer interface.

   Answer:

   Type `ovcomposer -m d`.

2. Open the NNM Composer configuration file in the Composer Information Base for the basic correlators.

   Answer:

   Select `File:Open` and browse to
   UNIX: `/etc/opt/OV/share/conf/ecs/CIB/NNMBasic.fs`
   Windows: *`install_dir`*`\conf\ecs\CIB/NNMBasic.fs`

3. Briefly examine each template definition area. Cancel out of each screen without saving any changes.

4. Examine the shipped Suppress correlator OV_Connector_IntermittentStatus. Do not make any changes.

   a. Double-click on the correlator.

   b. Review the Description tab. This is the help for the correlator. Which parameter is modifiable by customers?

      Answer:

      The Count parameter is configurable.

   c. Select the Definition tab. Which specific trap numbers are processed by this correlator?

      Answer:

      OV_IF_Down is processed by this correlator.

   d. What other data is used to further qualify which events are processed?

      Answer:

      It looks at varbind 12 to make sure this is a connector device.

   e. What is the content of that varbind?

      Answer:

      When you look in the Event Configuration interface, look at varbind 13 since that interface does not start counting with varbind 0. Varbind 8 contains a list of node capabilities.

   f. Cancel out of the Composer screen.

5. Exit the Composer interface.

Chapter 14, Creating a Basic Correlator

Lab Case: Suppress Correlation

Movement traps in general need investigation. However if the movement event are from exchanges emitted from the City offices, they can be discarded as there is always movement and the Alarm Browser is filled with these events. The requirement is to discard all movement events emitted from the City offices.

A sample SNMP trap PDU appears in a log as:

```
Trap PDU {
enterprise{1 2 3 4 999},
agent-addr internet : "\x0A\x00\x01\x7F",
generic-trap 6,
specific-trap 1,
time-stamp 414746291,
variable-bindings{
{
name {1 3 6 1 4 1 11 2 17 2 1 0},
value simple : number 2
},
{
name {13 6 1 4 1 11 2 17 2 2 0},
value simple : "City-Bangalore"
}
{
name {1 3 6 1 4 11 2 17 2 17 0},
value simple : string : "There is movement"
}
}
}
```

What do you need to know from the requirements?

1. How do you identify which events are to be Suppressed?

   Answer:

   All movements will have the following attributes which will identify them

   - enterprise id is set to 1.2.3.4.999
   - generic trap is set to 6
   - specific trap is set to 1

- variable bindings[1].value will have the string “City”
- variable bindings[2].value will have the string “There is Movement”.

View Pre-Correlator Results

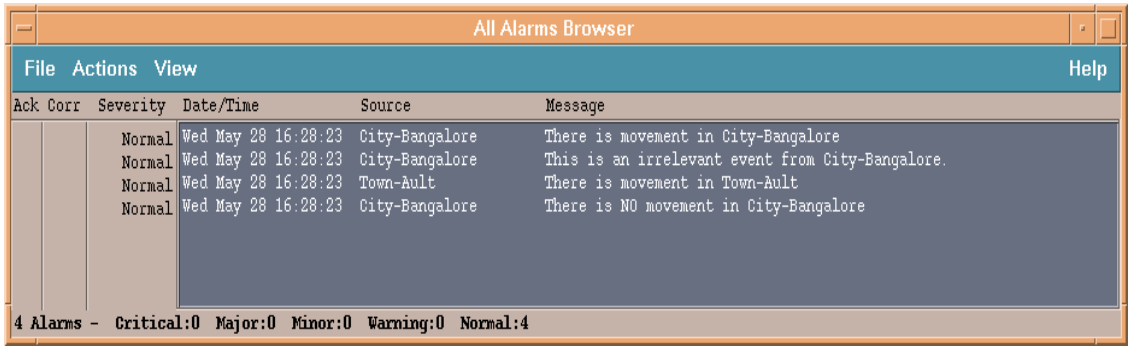
1. Event configurations for events used in the labs for this course are provided in `$OV_CONTRIB/OVTraining/NNM3/composerLab_trapd.conf`. To add these to your system configuration, type the commands:

```
cd $OV_CONTRIB/OVTraining/NNM3

xnmevents -merge composerLab_trapd.conf

xnmevents &
```

2. On Windows, correct the path for your system to include `%OV_BIN%\Perl\bin` if necessary.
   a. Right click My Computer on the desktop.
   b. Select Properties.
   c. Select the Advanced Tab.
   d. Click [`Environment Variables`].
   e. In the System variables window, select Path.
   f. Click [`Edit`].
   g. At the end of the Variable Value, append `;C:\Program Files\HP OpenView\bin\Perl\bin`
   h. Click [`OK`].
   i. Click [`OK`].
   j. Click [`OK`].
   k. Log out and log back in.
3. A sample event stream from this environment is available for testing. Copy the following file from `$OV_CONTRIB/OVTraining/NNM3` to `$OV_CONTRIB/ecs`:
   - `movement.evt`: sends the following events:
     — Movement from a city.
     — Wrong specific ID.
     — “City” replaced by “Town”.
     — String contains “There is NO movement”.
4. Open the Composer Labs Alarms Browser.
5. Empty all old alarms by selecting `Actions:Delete->All Alarms in Category`.
6. Type the commands:

```
cd $OV_CONTRIB/ecs

ecsevgen -n movement.evt
```

7. Review the alarms in the Alarm Browser.



8. To mark which alarms came from this pre-test run, select `Actions:Acknowledge->All Alarms in Category`.

Configure the Correlation

Follow the procedure given below to define the Suppress Correlator Template:

9. Start the Composer development interface.

   Answer:

   Type **`ovcomposer -m d &`**.

10. Create a new correlator store with SNMP events.

    Answer:

    a. Select `File:New`.

    b. Click the SNMP button and click [OK].

11. Select `Correlations:Correlator Templates->Suppress` from the Correlator Store window. The Suppress Correlator Template window opens.

    You can also click on the Suppress icon in the Correlator Templates toolbar.

12. Enter the name of the Correlator in the Name text field.

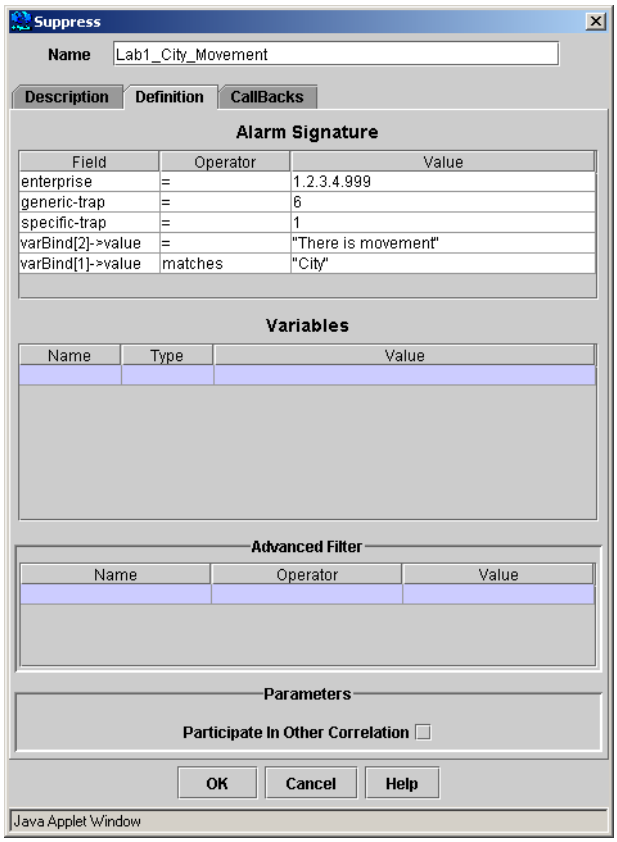
13. Enter the description of the Correlator. The Description can briefly state what the Correlator is expected to do.

14. Set the following values to define the Alarm Signature

    - enterprise = 1.2.3.4.999
    - generic-trap = 6
    - specific-trap = 1
    - variable-binding [2].value = “There is movement”
    - variable-binding [1].value matches “City”

15. Click [OK] to complete the Correlator. Notice that the correlation you have just defined is

displayed in the Correlator Store table.



16. Save your Correlator Store in the `$OV_CONF/ecs/CIB` directory.

    Answer:

    a. Select `File:Save As.`

    b. Browse to `$OV_CONF/ecs/CIB` and name your file `suppressLab`. The .fs extension is added automatically.

17. Exit the Composer developer interface.

18. Review the files created by saving the Correlator Store.

    Answer:

    `ovcomposer` created `suppressLab.fs`. In addition, it created the default file as a backup. Finally, it created a sample security file, `suppressLab.sec`.

19. Provide Operator access to your correlator using the existing files.

    Answer:

    a. Edit the `$OV_CONF/ecs/CIB/NameSpace.conf` file.

    b. At the bottom of the file add a line giving a short name and the path to your Correlator Store (relative to `$OV_CONF/ecs/CIB`).

    ```
    Lab_Suppress=suppressLab.fs
    ```

    c. You may verify the contents of `suppressLab.sec`, but they do not need to be changed.

    d. You do not need to edit or create a deploy file since you added your correlator to the

existing `NameSpace.conf` file. You may review the `deploy.conf` file to see how it is linked to the NameSpace file name.

20. As an Operator, enable and deploy your correlation.

Answer:

a. Start the Operator GUI, either from the ECS GUI or through `ovcomposer -m o`.

b. Select the NameSpace Lab_Suppress in the NameSpace Table.

c. Check the Enable box for the correlator.

d. Close the Lab_Suppress correlator store by selecting `File:Close`.

e. Select `Correlations:Deploy`. Click [`OK`]. You may also use the deploy toolbar button.

Defect filed on Windows GUI deployment. Workaround is to use the command line utility `csdeploy.ovpl`.

f. Exit the Operator GUI.

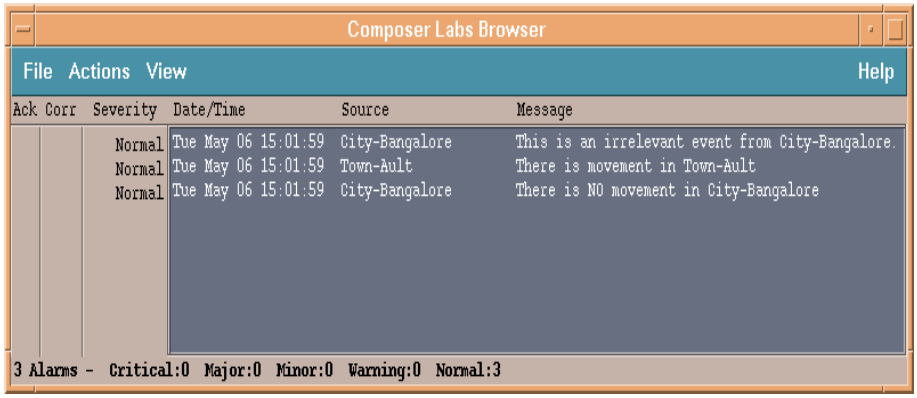
Test Your Correlator

21. Watch your Composer Labs Alarms Browser to watch the operation of your correlation.

22. Type the commands:

```
cd $OV_CONTRIB/ecs

ecsevgen -n movement.evt
```

23. Examine the Alarms Browser. You should see no indication of the first event, which was suppressed. The rest of the events fail to suppress and show in the Alarm Browser. Look at the information there about the actual contents of the various varbinds.



a. update the line for the suppress lab to include the subdirectory.

24.

Chapter 15, Using Variables in Correlators

LAB: Suppress Correlator

This Lab Session requires:

- Suppress Correlator
- Alarm Signature
- Advanced Filter
- Operators
- Constant Variable
- Lookup Variable

Correlation Requirement

Discard all interface-down events from a list of interfaces on the first day of January in all the years. These interfaces will go down for maintenance during this day.

The list of interfaces will be stored in a datastore in the following format with the key "IF_Failure":

`Composer.ds` DataStore:

```
ADD DATA ("IF_Failure", ["First", "Second", "Third", "Fifth"])
```

Example Trap-PDU

```
Trap-PDU {
    enterprise {1 6 7 8},
    agent-addr internet : "\x7f\x00\x00\x02",
    generic-trap 6,
    time-stamp 0,
    variable-bindings {
{
    name {1 3 6 1 4},
    value simple : string : "Second"
},
{
    name {1 3 6 1 4 11},
    value simple : string : "01:01:03:01:00:01"
},
{
    name {1 2 6 5 8},
```

```
    value simple : number : 20
}
    }
}
```

Event Definition

An interface-down trap is identified by

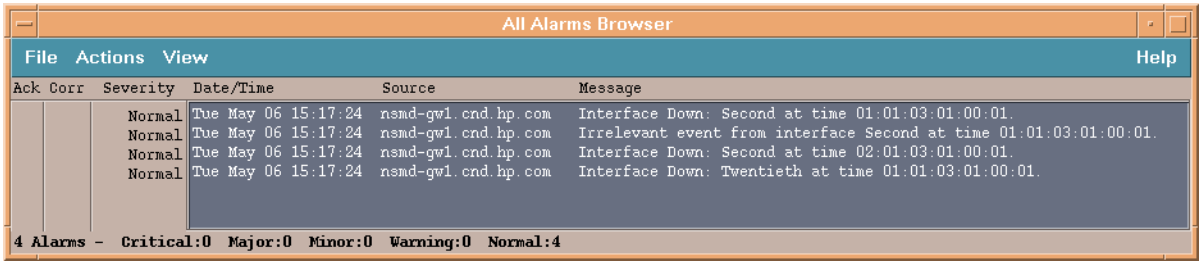
- enterprise id set to 1.6.7.8
- generic-trap set to 6
- varBind [0] contains interface number
- varBind [1] contains the time-stamp in the format dd:mm:yy:hh:mi:ss

View Pre-Correlator Results

1. A sample event stream from this environment is available for testing. Copy the following file from `$OV_CONTRIB/OVTraining/NNM3` to `$OV_CONTRIB/ecs`:
   - `maintIF.evt`: sends the following events:
     — An event that matches the signature for this lab.
     — An event with a different specific ID.
     — An event from a different date.
     — An event from a different interface.
2. Open the Composer Labs Alarms Browser.
3. Empty all old alarms by selecting `Actions:Delete->All Alarms in Category`.
4. Type the commands:

   ```
   cd $OV_CONTRIB/ecs
   ecsevgen -n maintIF.evt
   ```

5. Review the alarms in the Alarm Browser.



6. To mark which alarms came from this pre-test run, select `Actions:Acknowledge->All Alarms in Category`.

Directions

Execute the following steps to solve the Lab Session:

7. Start the Composer development interface.

   Answer:

Type **ovcomposer -m d &**.

8. Create a new correlator store with SNMP events.

   Answer:

   a. Select File:New.

   b. Click the SNMP button and click [OK].

9. Create a Suppress correlator.

   Answer:

   Select Correlations:Correlator Templates->Suppress from the Correlator Store window. The Suppress Correlator Template window opens.

   You can also click on the Suppress icon in the Correlator Templates toolbar.

10. In the Alarm Signature section, enter the following to filter the interface down events:

Field	Operator	Value
enterprise	=	1.6.7.8
generic-trap	=	6

11. The list of interfaces are available in the datastore with a key value “IF_Failure”. Use Lookup variable and pass the key value to look for in the datastore to get the list of interface failures.

    a. Create variables to pass the key and to get the interface failures from datastore.

Name	**Type**	**Value**
IFKey	Constant	“IF_Failure”
IFLookup	Lookup	IFKey

    b. Create a variable to filter the events with time stamp matches the 1st of January as shown below.

    Variables

Name	**Type**	**Value**
TimeStamp	Constant	"^01:01"

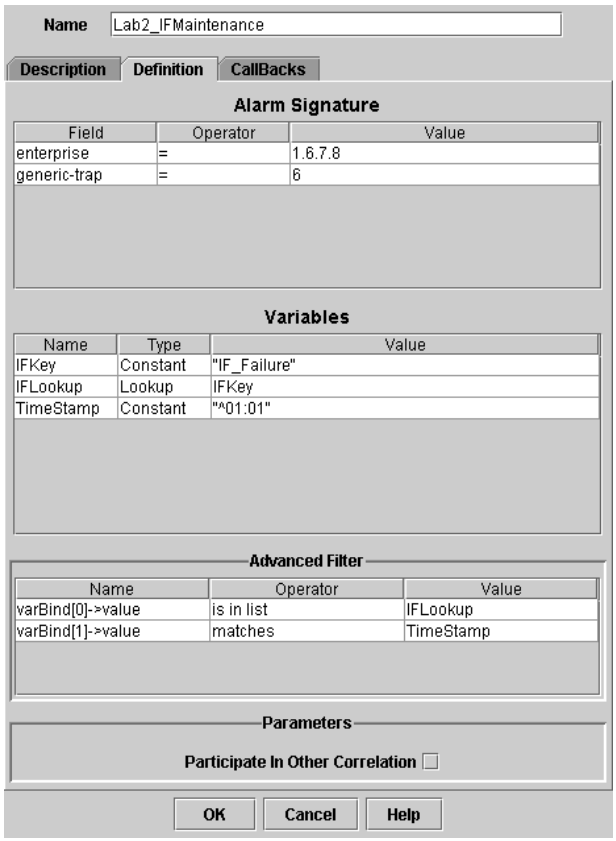
12. In the Advanced Filter, evaluate the incoming event’s varBind[0] value with datastore lookup value to check whether the interface is under maintenance. Also evaluate the incoming event’s varBind[1] value matches the TimeStamp. Enter the values in the Advanced Filter as shown below.

    Advanced Filter

Name	**Operator**	**Value**
VarBind[0]->value	is in list	IFLookup
VarBind[1]->value	matches	TimeStamp

    If the incoming event passes the Alarm Signature and Advanced Filter, then it will be suppressed. In the Advanced Filter, the operator “matches” checks the incoming event’s

varBind [1] value for the exact pattern of first five characters specified in the TimeStamp value. For this the "^" is used. The first 5 characters of varBind[1] contains "01:01" then, only those events will be passed through this Advance filter and all those events are discarded(suppressed).



13. Save your Correlator Store in the `$OV_CONF/ecs/CIB` directory.

    Answer:

    a. Select `File:Save As.`

    b. Browse to `$OV_CONF/ecs/CIB` and name your file `lookupLab`. The .fs extension is added automatically.

14. Close the file in the developer interface. You can leave the interface running.

15. Add your data to the `Composer.ds` datastore.

    Answer:

    Edit the `Composer.ds` DataStore and add the line:

    ```
    ADD DATA ("IF_Failure", ["First", "Second", "Third", "Fifth" ])
    ```

    Disable and Enable Composer in the ECS GUI to have it reread your data store file.

16. Provide Operator access to your correlator.

    Answer:

    a. Edit the `$OV_CONF/ecs/CIB/NameSpace.conf` file.

    b. At the bottom of the file add a line giving a short name and the path to your Correlator Store.

```
Lab_Lookup=lookupLab.fs
```

17. Deploy your correlation from the command line.

    Answer:

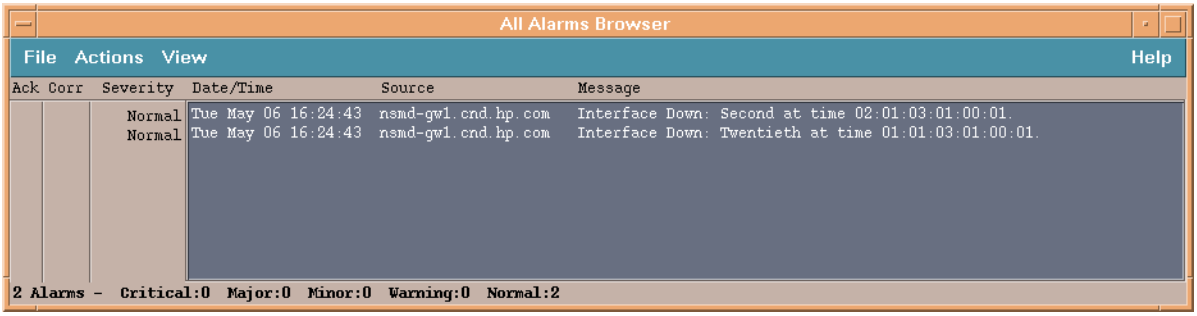
    Type `csdeploy.ovpl`.

Test Your Correlator

18. Type the commands:

    ```
    cd $OV_CONTRIB/ecs
    ecsevgen -n maintIF.evt
    ```

19. Examine the Composer Labs Alarms Browser. You should see a no indication of the first event. There should be no indication of the second event since the enterprise ID and generic ID still match the signature. The third event should not be correlated since it is not from January, so you see it in the Alarm Browser. The fouth event also shows in the Alarm Browser since the interface is not in your data store.



LAB: Enhance Correlator Template (Optional)

This Lab Session requires:

- Enhance Correlator
- Combine Variable
- Lookup Variable
- New Alarm Section
- Global Constants

Correlation Requirement

Assume a wide area network with many small networks interconnected by routers (primary and secondary routers to share the load). Some of the sub-networks are connected only by one router (primary) as shown.

If a Router down event is emitted, the correlator should check if the problematic router has a secondary router to share the load.

- If yes, the event should be enhanced with the information that “the problematic router is backed-up by secondary router and all the traffic will be routed through the secondary router”.
- If no, the event should be enhanced with warning information that “no backup router is available and the sub-network is isolated”.

The router is uniquely identified by a combination of enterprise ID + Router ID (varBind[0] value). The secondary router’s id is available in the datastore and can be retrieved by passing the primary router id.

The datastore looks like:

```
ADD DATA ("<enterpriseid + Rotuerid>", "<Secondary Router information>" )
```

Trap Definition

A router fail trap is identified by:

- enterprise id set to 1.5.6.7.8
- specific-trap set to 6
- var-bind0 contains router id

Example Trap-PDU

```
Trap-PDU {
    enterprise {1 5 6 7 8},
    agent-addr internet : "\x7f\x00\x00\x02",
    specific-trap 6,
    time-stamp 414746291,
    variable-bindings {
{
    name {1 3 6 1},
    value simple : number : 5
```

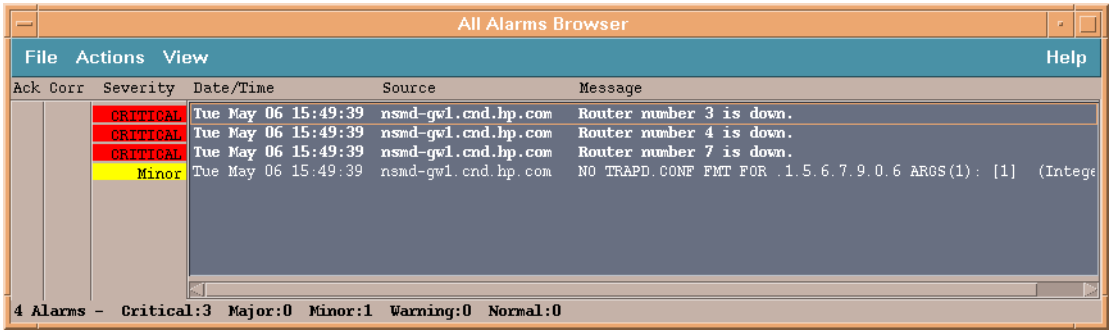
```
}
    }
}
```

View Pre-Correlator Results

1. A sample event stream from this environment is available for testing. Copy the following files from `$OV_CONTRIB/OVTraining/NNM3` to `$OV_CONTRIB/ecs`:
   - `routerBackup.evt`: sends the following events:
     - — From a router that has a secondary.
     - — From a router that does not have a secondary.
     - — The Router ID is not in the Lookup list.
     - — The enterprise does not match.
2. Open the All Alarms Browser. (Since one of these events is purposely not listed in trapd.conf, the configuration alarm for that only appears in the All Alarms Browser.)
3. Empty all old alarms by selecting `Actions:Delete->All Alarms in Category`.
4. Type the commands:

   ```
   cd $OV_CONTRIB/ecs
   ecsevgen -n routerBackup.evt
   ```

5. Review the alarms in the Alarm Browser. (Hint: The event with a different enterprise ID only shows in the All Alarm Browser.)



6. To mark which alarms came from this pre-test run, select `Actions:Acknowledge->All Alarms in Category`.

Directions

Execute the following steps to solve the Lab Session:

1. Start the Composer development interface.

   Answer:

   Type **`ovcomposer -m d &`**.

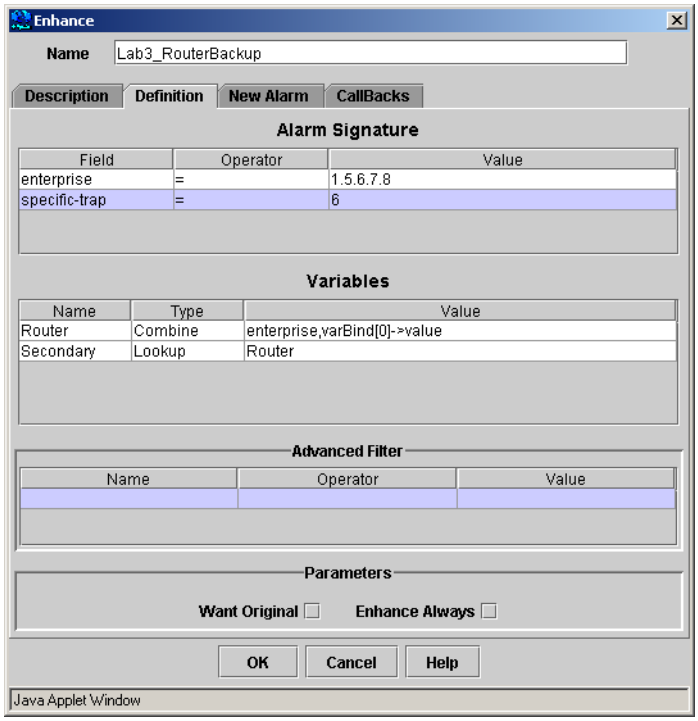
2. Create a new correlator store with SNMP events.

   Answer:

   a. Select `File:New`.

b. Click the SNMP button and click [OK].

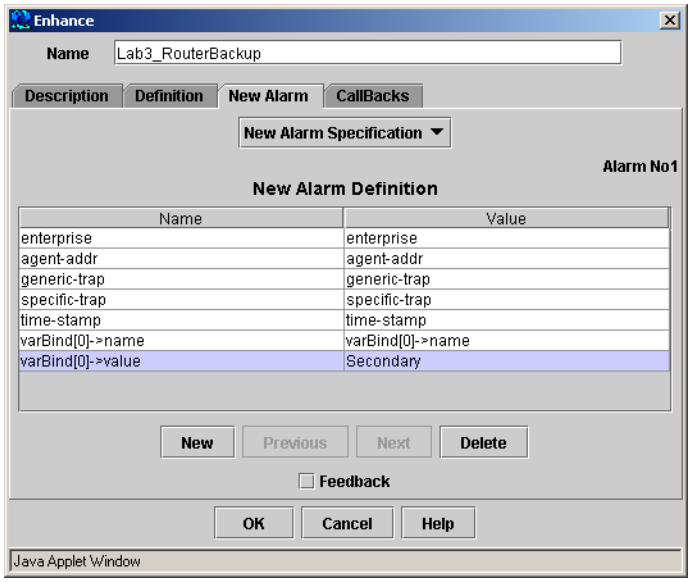
3. Use Enhance Correaltor Template.
4. In Alarm Signature section, filter the router down events from a network.
5. The router is uniquely identified by the combination of enterprise id and router id (varBind[0] value). Create a Combine variable and add these two attributes.
6. The secondary router detail is available in the datastore. To get the secondary details, create a Lookup variable and pass the unique router created in the last step.



7. Use the New Alarm Section and enhance the event with the secondary router information. Add a varbind name and value to contain the Secondary variable information.

   If the incoming event passes the Alarm Signature and Advanced Filter, the variables which are used in the New Alarm section are evaluated and the enhanced event is output from the

Correlator.



8. Save your Correlator Store in the `$OV_CONF/ecs/CIB` directory.

   Answer:

   a. Select `File:Save As`.

   b. Browse to `$OV_CONF/ecs/CIB` and name your file `enhanceLab`. The .fs extension is added automatically.

9. Exit the Composer developer interface.

10. Add your data to the datastore. Refer to the lab exercise slide for the connectivity diagram.

    Answer:

    Place the ADD DATA lines in `Composer.ds`.

    For example, the datastore will have values like:

    ```
    ADD DATA ("1.5.6.7.83", "This has Secondary Router ")
    ADD DATA ("1.5.6.7.84", " This does not have Secondary Router ")
    ```

    Disable and Enable Composer in the ECS GUI to have it reread your data store file.

11. Provide Operator access to your correlator.

    Answer:

    a. Edit the `$OV_CONF/ecs/CIB/NameSpace.conf` file.

    b. At the bottom of the file add a line giving a short name and the path to your Correlator Store (relative to `$OV_CONF/ecs/CIB`).

    ```
    Lab_Enhance=enhanceLab.fs
    ```

12. Deploy your correlation.

    Answer:

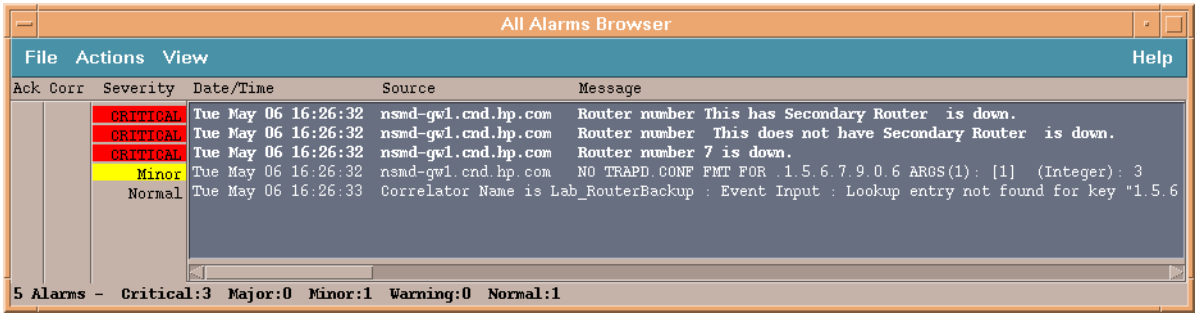
    Type `csdeploy.ovpl`.

Test Your Correlator

13. Type the commands:

```
cd $OV_CONTRIB/ecs

ecsevgen -n routerBackup.evt
```

14. Examine the Composer Labs Alarms Browser. You should see a new event stating that there is a secondary, a new event stating that there is no secondary, an original event with a router ID not in the list, and an original event from a different enterprise. Composer also displays an alarm that the router ID not in the list failed the Lookup in the All Alarms Browser.



LAB: Extract Variable

This Lab Session requires:

- Enhance Correlator
- Alarm Signature
- Advanced Filter
- Operators
- Extract Variable

Correlation Requirement

A network spooler emits an event whenever a new job is submitted. This event can be ignored unless the event indicates that the specified output device is unavailable and the job has high priority (over 5). In this case, emit a new event indicating whom to call and why.

Example Trap-PDU

```
Trap-PDU{

enterprise {1 2 3 4},

agent-addr internet : "\x7f\x00\x00\x02",
generic-trap 6,

specific-trap 10,

variable-bindings{

{

name {1 2 3 4 1},

value simple : string : Location=Building 41, Roswell; Contact=John Bigboote, 729-315-4545

},

{

name {1 3 6 1 4 11 2 17 2 1 0},

value simple : string : Model=5317;MaxSpace=6420000K;MaxQueues=12;MaxJobs=512
},

{

name {1 3 6 1 4 1 11 2 17 2 1 0}

value simple : string : CurJobs=5;CurFreeSpace=23980;NonEmptyQueues=2;CurMaxDepth=3
},

{

name {1 3 6 1 4 1 11 2 17 2 1 0}

value simple : string :
JobID=345;Target=ljet1;Size=45135;Submit=20030519.114525;Prio=7;Type=Binary
},

{

name {1 3 6 1 4 1 11 2 17 2 1 0}

value simple : string : Type=printer;Model=Laserjet 5 MX;Avail=N;Status=TonerLow;Error=37
}

}
```

```
}
```

Event Definition

An interface-down trap is identified by

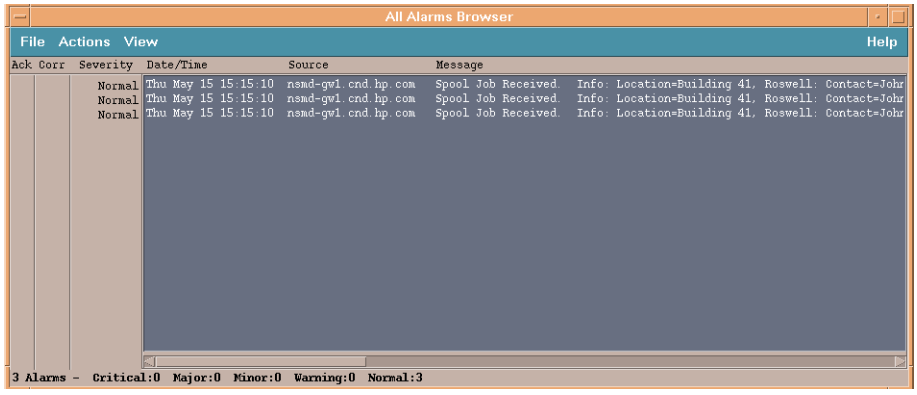
- enterprise is set to 1.2.3.4
- generic-trap is set to 6
- specific-trap is set to 10
- var-bind[0] of the event provides the location and contact of the spooler.
- var-bind[1] provides information about the spooler.
- var-bind[2] provides information about the spooler status.
- var-bind[3] provides information about the job.
- var-bind[4] provides information about the status of the output device.

View Pre-Correlator Results

1. A sample event stream from this environment is available for testing. Copy the following file from `$OV_CONTRIB/OVTraining/NNM3` to `$OV_CONTRIB/ecs`:
   - `spooler.evt`: sends the following events:
     — high priority job with printer down
     — high priority job with printer Normal
     — low priority job with printer down
2. Open the Composer Labs Alarms Browser.
3. Empty all old alarms by selecting `Actions:Delete->All Alarms in Category`.
4. Type the commands:

   ```
   cd $OV_CONTRIB/ecs
   ecsevgen -n spooler.evt
   ```

5. Review the alarms in the Alarm Browser.



6. To mark which alarms came from this pre-test run, select `Actions:Acknowledge->All Alarms in Category`.

Directions

Execute the following steps to solve the Lab Session:

7. Start the Composer development interface.

   Answer:

   Type **`ovcomposer -m d &`**.

8. Create a new correlator store with SNMP events.

   Answer:

   a. Select `File:New`.

   b. Click the SNMP button and click [`OK`].

9. Create an Enhance correlator.

   Answer:

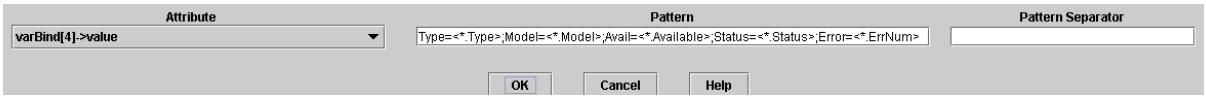
   Select `Correlations:Correlator Templates->Enhance` from the Correlator Store window. The Enhance Correlator Template window opens.

   You can also click on the Enhance icon in the Correlator Templates toolbar.

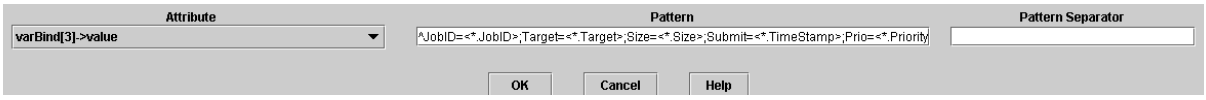
10. In the Alarm Signature section, enter the following to filter the interface down events:

Field	Operator	Value
enterprise	=	1.2.3.4
generic-trap	=	6
specific-trap	=	10

11. Create a variable named `OutputDevice`. Extract the output device information into the variables `OutputDevice.Status` and `OutputDevice.Name` from varbind[4].

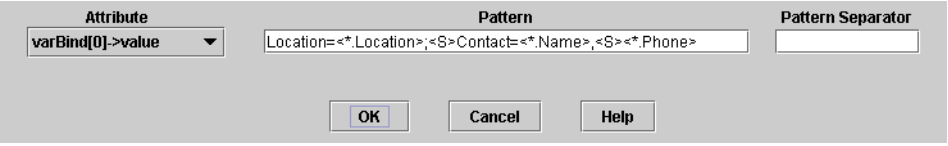


12. Create a variable named `JobInfo`. Extract the job priority from varbind[3] into `JobInfo.Priority`. and `JobInfo.Target` and `JobInfo.JobID`.

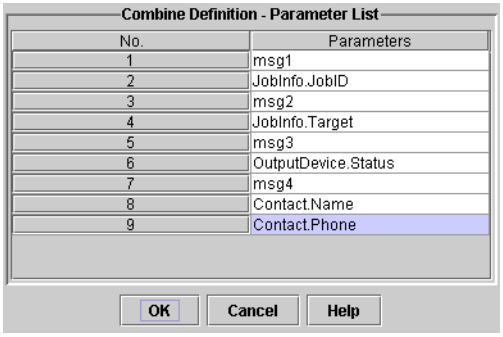


13. Create a Constant variable, `HiPri`, to hold the minimum interesting priority, “5”. Create a variable to hold the string “Normal”.

14. Extract the contact information from varbind [0] into `Contact.Name` and `Contact.Phone`.



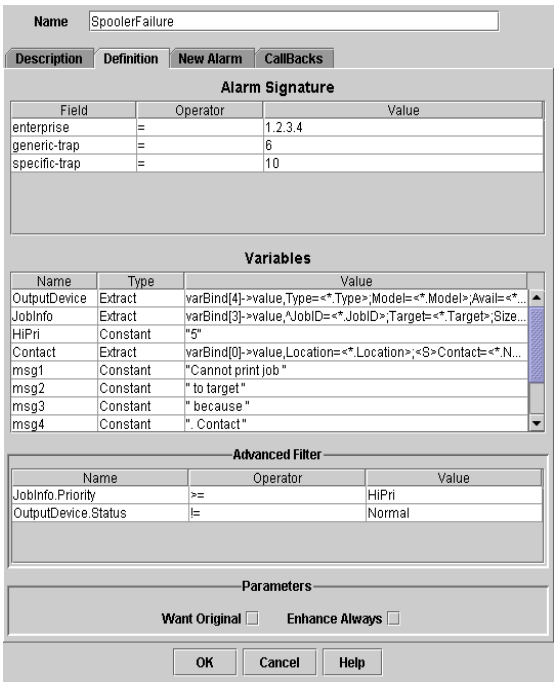
15. Create constant variables to hold the parts of the message string. The final message string should read, “Cannot print job 345 to target ljet1because Toner Low. Contact John Bigboote, 729-315-4545.” The strings are “Cannot print job ”, “ to target ”, “ because ” “. Contact ”.
16. Create a Combine variable named ErrStr to build the message string.



17. Create a variable `NewSpecific` to hold the specific ID for the new event, 11.
18. In the Advanced Filter, evaluate the incoming event’s priority against `HiPri`. Also evaluate the incoming event’s device status. Enter the following values:

Name	**Operator**	**Value**
JobInfo.Priority	>=	HiPri
OutputDevice.Status	!=	Normal

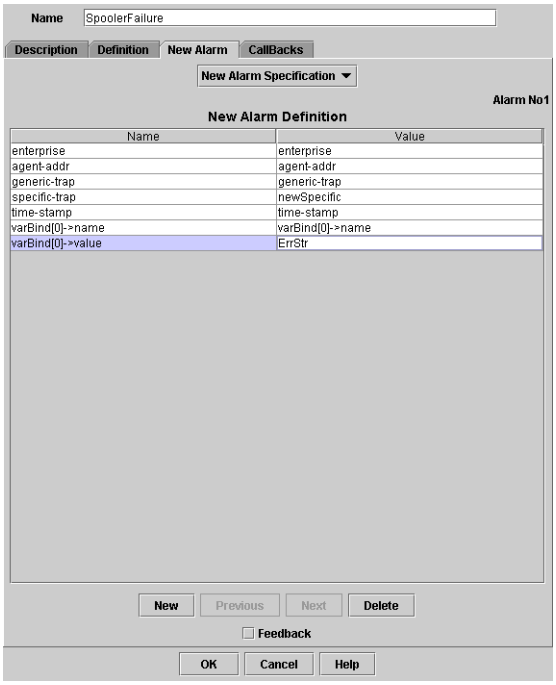
19. Allow the incoming event to be discarded.



Answer:

Do not check the box for Want Original.

20. Create the new event, placing the message string in varbind[0].



21. Save your Correlator Store in the `$OV_CONF/ecs/CIB` directory.

Answer:

a. Select `File:Save As`.

b. Browse to `$OV_CONF/ecs/CIB` and name your file `extractLab`. The .fs extension is added automatically.

22. Exit the Composer developer interface.

23. Provide Operator access to your correlator.

Answer:

a. Edit the `$OV_CONF/ecs/CIB/NameSpace.conf` file.

b. At the bottom of the file add a line giving a short name and the path to your Correlator Store (relative to `$OV_CONF/ecs/CIB`).

```
Lab_Extract=extractLab.fs
```

24. Deploy your correlation.

Answer:

Type `csdeploy.ovpl`.

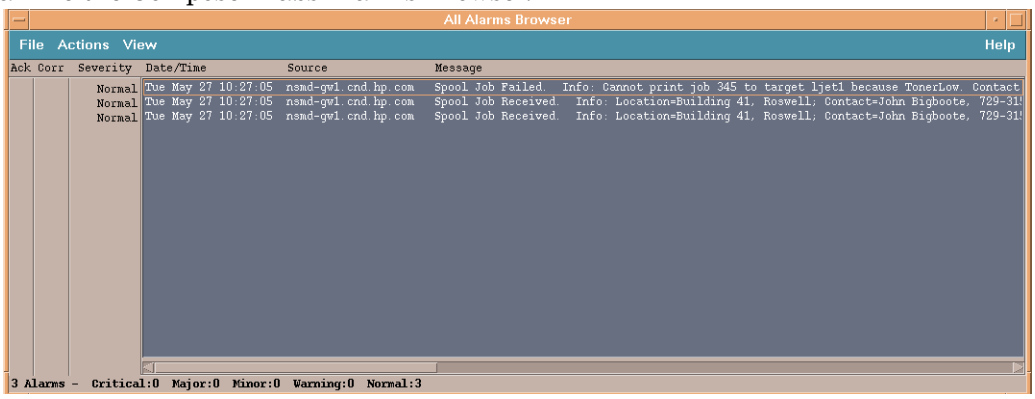
Test Your Correlator

25. Type the commands:

```
cd $OV_CONTRIB/ecs
```

```
ecsevgen -n spooler.evt
```

26. Examine the Composer Labs Alarms Browser.



Chapter 16, Using Additional Correlators

Lab Case: Rate Correlation

Robotic arms on production lines may report failures during high volume conditions. The requirement is to discard all robotic arm failures if the rate of failure is below 5 failures in 30 minutes. If the rate exceeds this threshold, forward the last event to the browser after annotating the event with the rate.

Requirements

A sample SNMP trap PDU for an arm failure could appear in an event log as below:

```
Trap-PDU{
enterprise {1 2 3 4 }
agent-addr internet : "\x0A\x00\x01\x7F",
generic-trap 6,
specific trap 80,
time-stam 414746291,
variable-bindings{
{
name{1 3 6 1 4 11 2 17 2 1 0},
value simple:number : 2
},
{
name { 13 6 1 4 11 2 17 2 2 0},
value simple : string : "Arm#10#CTRL#20"
}
}
}
```

What do you need to know?

1. How do you identify the events for which the count will be maintained?

   Answer:

   A count will be maintained for events whose attributes have the following values

   - `enterprise is 1.2.3.4`
   - `generic-trap is 6`
   - `specific-trap is either 40 or 80`
   - `variable-bindings[0].value is in the range 0 to 350.`

2. What do you with the events?

Answer:

Duplicate events can be discarded. However, the correlation has to be defined to monitor the time the event is discarded or output.

3. How will you match the traps?

Answer:

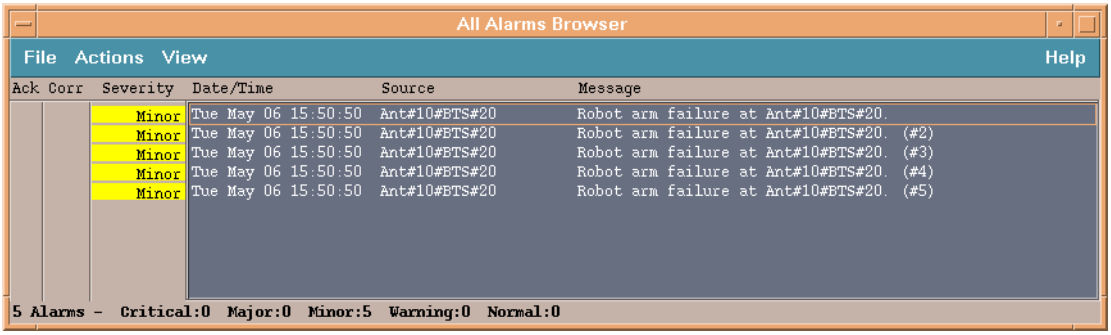
Two events are emitted from the same robotic arm and controller if their corresponding armid, ctrlid and agent-addr are the same.

View Pre-Correlator Results

1. A sample event stream from this environment is available for testing. Copy the following file from `$OV_CONTRIB/OVTraining/NNM3` to `$OV_CONTRIB/ecs`:
   - `robot_5.evt`: sends 5 traps.
2. Open the Composer Labs Alarms Browser.
3. Empty all old alarms by selecting `Actions:Delete->All Alarms in Category`.
4. Type the commands:

```
cd $OV_CONTRIB/ecs
ecsevgen -n robot_5.evt
```

5. Review the alarms in the Alarm Browser.



6. To mark which alarms came from this pre-test run, select `Actions:Acknowledge->All Alarms in Category`.

Directions

Follow the procedure given below to define the Rate Correlator Template:

7. Select `Correlations:Correlator Templates->Rate` from the Correlator Store. The Rate Correlator Template window opens. You can also click on the Rate Correlator icon in the Correlator Templates toolbar.
8. Enter the Name and Description for the Correlator.
9. Enter the following values to identify the Alarm Signature
   - enterprise
   - generic-trap
   - specific-trap

10. Declare the following variables

    - arm - This is a variable that in combination of the extracted pattern will specify the Arm ID and Controller ID.

      — Extract the Arm ID and Controller from variable-bindings[1].value. In the extract pattern window enter `Arm#<*.armid>#CTRL#<*.ctrlid>`

      — Leave the Pattern Separator field blank.

    - `mkey` - This is the unique field that will combine all the above attributes into one and constitute the Message Key.

      — Combine the attributes `arm.armid`, `arm.ctrlid`, `agent-addr`.

11. Select the Message Key. Click in the MessageKey window. A pop up menu displays all attributes and pre-defined variables. Select `mkey` from the menu.

12. Define the parameters for the correlation.

    - `Window Period` = 30 minutes
    - `Count` = 6

13. Select the Discard button. Although the events are discarded, the count of event arrival is maintained.

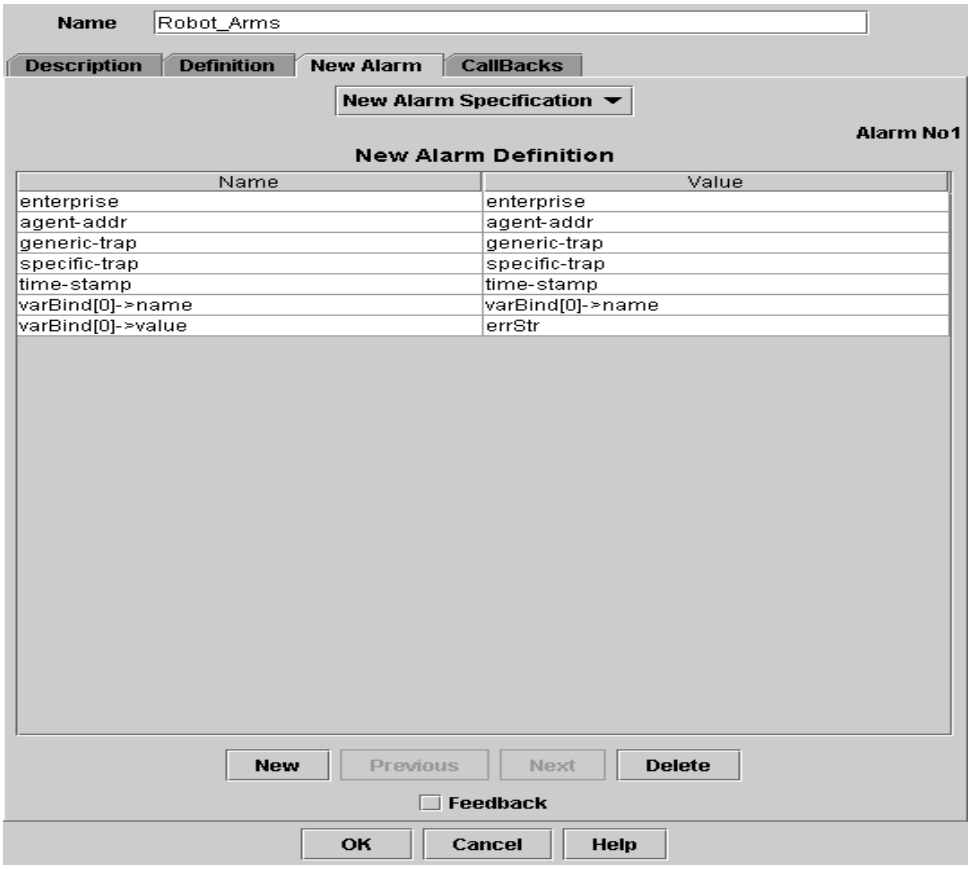
14. Before a new event is created, it is necessary to define the error string that declares the problem. Define the following variables in the Variable table.

    - `str1` constant “The threshold has been breached for the robotic arm ”
    - `str2` constant “ from Controller ”
    - `errstr` combine of `str1`, `arm.armid`, `str2`, `arm.ctrlid`

    The above definition creates an errstr which will look like “The threshold has been breached

for the robotic arm 10 from Controller 20”.

15. Define the new event. Click on the New Alarms tab to alter the event. The New Alarm panel opens.
16. Select New Alarm Specification from the drop down menu. The New Alarm Definition table is displayed.
17. Select the following to define the change
    - enterprise = enterprise
    - agent-addr = agent-addr
    - generic-trap = generic-trap
    - specific-trap = specific-trap
    - time-stamp = time-stamp
    - varBind[0]->name=varBind[0]->name
    - varBind[0]->value = errstr
18. Click on [OK] to complete the definition of the Correlator. Notice that the Correlator you have just defined is displayed in the Correlator Store table.



19. Save your Correlator Store in the `$OV_CONF/ecs/CIB` directory.

    Answer:

    a. Select `File:Save As.`

    b. Browse to `$OV_CONF/ecs/CIB` and name your file `rateLab`. The .fs extension is added automatically.

20. Exit the Composer developer interface.

21. Provide Operator access to your correlator.

    Answer:

    a. Edit the `$OV_CONF/ecs/CIB/NameSpace.conf` file.

    b. At the bottom of the file add a line giving a short name and the path to your Correlator Store (relative to `$OV_CONF/ecs/CIB`).

    ```
    Lab_Rate=rateLab.fs
    ```

22. Deploy your correlation.

    Answer:

    Type `csdeploy.ovpl`.

Test Your Correlator

23. Demonstrate that your correlation holds traps during the window. Type the commands:

```
cd $OV_CONTRIB/ecs

ecsevgen -n robot_5.evt
```

Answer:

No alarms show up in the Alarm Browser because the first 5 are suppressed and do not trigger the threshold.

24. Demonstrate that your correlation creates a new event when too many arrive. Type the `ecsevgen` command one more time.

LAB: Using Transient Correlator

This Lab Session requires:

- Transient Correlator
- New Alarm Section
- Global Constant `AlarmCnt`

Correlation Requirement

One of the A/C machines in a floor is having problems. It runs for a while and stops and starts again after a while. The facility management system manages all this equipment. It emits up and down events for the A/C machine.

- The varBind [0] value of the up and down events provides the floor number.
- The varBind [1] value of the up and down events provides the room temperature.

Create a correlator that suppresses transient events indicating ups and downs, when the room temperature is with in an acceptable range. If the temperature is not in the acceptable range, then send a new event for every 10 minutes with information of number of ups and downs in that window and suppress all the ups and downs events. If 3 up and down events emitted within 10 minutes, then create a new event.

Alarm Definition

Example Trap-PDU

```
Trap-PDU {
    enterprise {1 2 3 4},
    agent-addr internet : "\x7f\x00\x00\x02",
    generic-trap 6
    specific-trap 12,
    time-stamp 414746291,
    variable-bindings {
{
    name {1 3 6 1},
    value simple : number : 2
}
{
    name {1 3 6 8},
    value simple : number : 27
}

    }
}
```

AC Up and Down events are identified by:

- enterprise id set to 1.2.3.4

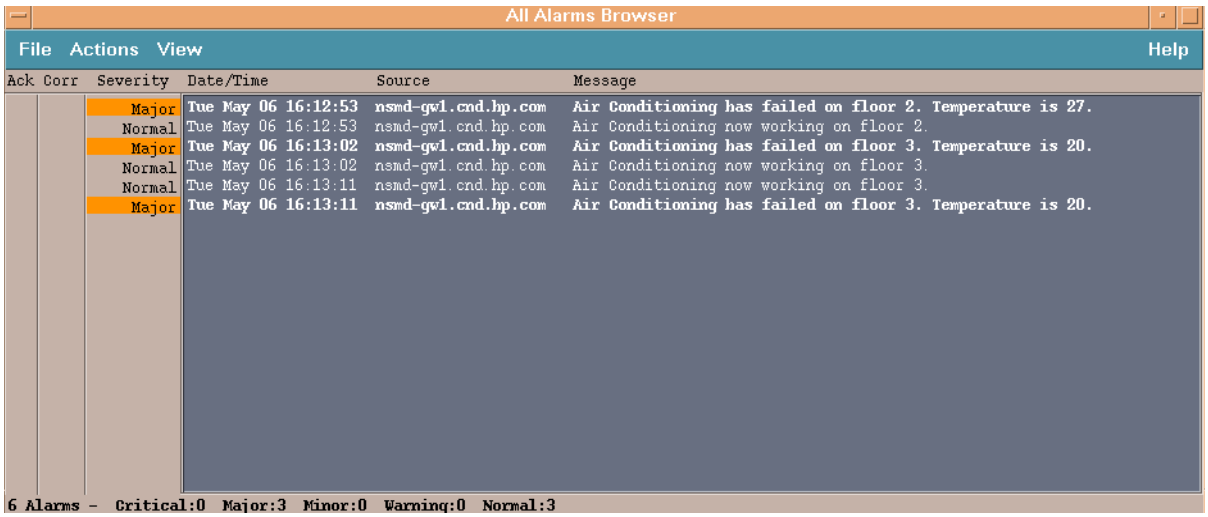
- generic-trap set to 6
- specific-trap set to 12 or 13 (12 is down event and 13 is up event)
- var-bind[0] contains Floor number
- var-bind[1] contains temperature

View Pre-Correlator Results

1. A sample event stream from this environment is available for testing. Copy the following files from `$OV_CONTRIB/OVTraining/NNM3` to `$OV_CONTRIB/ecs`:
   - `AC_hot.evt`: send a down, then an up trap with the temperature set to 27.
   - `AC_cool.evt`: send a down, then an up trap with the temperature set to 20.
   - `AC_reverse.evt`: send an up then a down trap.
2. Open the Composer Labs Alarms Browser.
3. Empty all old alarms by selecting `Actions:Delete->All Alarms in Category`.
4. Type the commands:

```
cd $OV_CONTRIB/ecs

ecsevgen -n AC_hot.evt

ecsevgen -n AC_cool.evt

ecsevgen -n AC_reverse.evt
```

5. Review the alarms in the Alarm Browser.



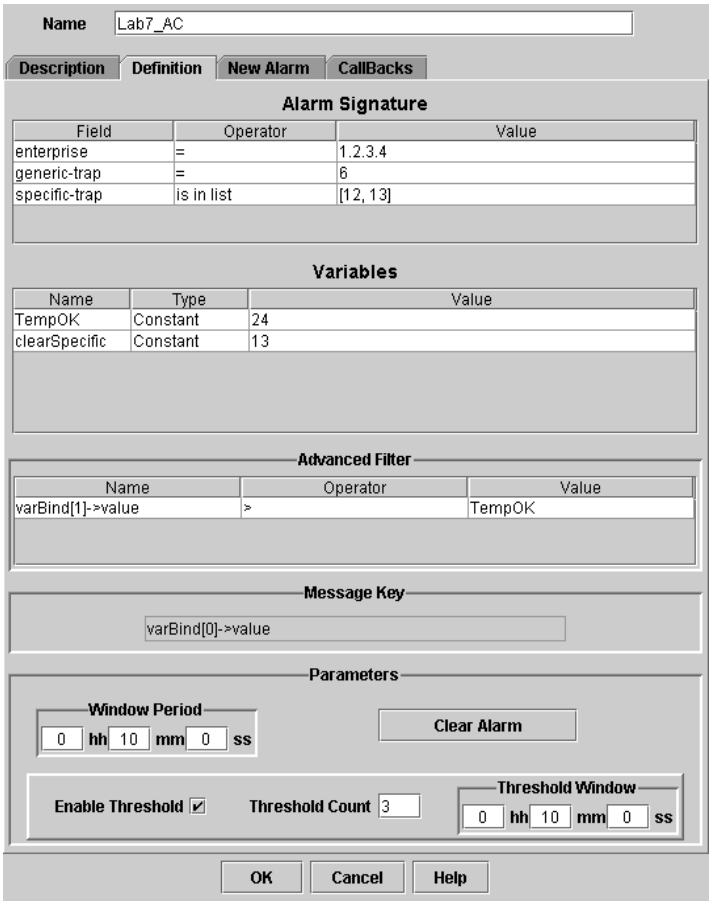
6. To mark which alarms came from this pre-test run, select `Actions:Acknowledge->All Alarms in Category`.

Directions

Execute the following steps to solve the lab:

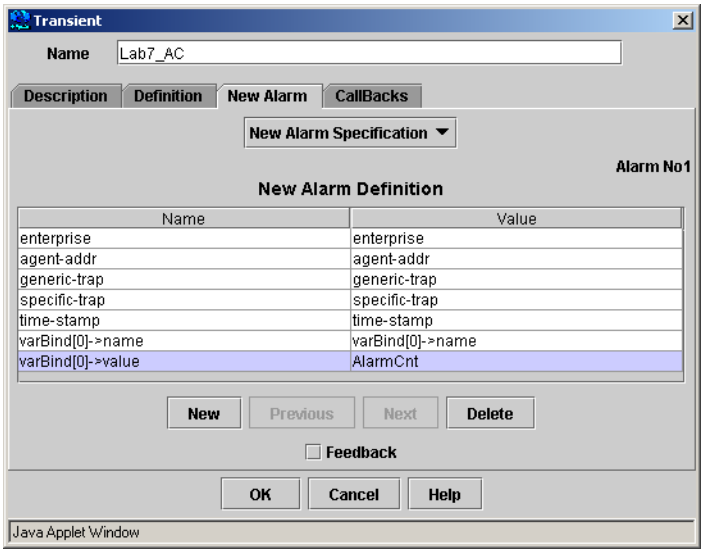
7. Use the Transient Correlator Template.
8. In the Alarm Signature section, filter the AC up/down events.
9. Create a variable to hold the acceptable temperature value. For example, if the temperature is >= 24, then it is considered outside the acceptable range.

10. The varbind[1].value contains the temperature value. In the Advanced Filter, compare the varbind[1].value with the temperature variable.
11. The varbind[0].value contains the floor number. To correlate all events coming from the same floor, use varbind[0].value as the Message Key.
12. Specify the Window Period.
13. Specify the Clear Alarm. In this case, specific trap 13 is the clear event.
14. Enable the Threshold Count and specify count and Threshold Window period.



15. Create a new event and map the required attributes and bind automatic variable `AlarmCnt` to

any attribute in the new event.



16. Save your Correlator Store in the `$OV_CONF/ecs/CIB` directory.

    Answer:

    a. Select `File:Save As.`

    b. Browse to `$OV_CONF/ecs/CIB` and name your file `transientLab`. The .fs extension is added automatically.

17. Exit the Composer developer interface.

18. Provide Operator access to your correlator.

    Answer:

    a. Edit the `$OV_CONF/ecs/CIB/NameSpace.conf` file.

    b. At the bottom of the file add a line giving a short name and the path to your Correlator Store (relative to `$OV_CONF/ecs/CIB`).

    ```
    Lab_Transient=transientLab.fs
    ```

19. Deploy your correlation.

    Answer:

    Type `csdeploy.ovpl`.

Test Your Correlator

20. Demonstrate that your correlation suppresses an initial down event when an up event arrives. Type the commands:

    ```
    cd $OV_CONTRIB/ecs
    ecsevgen -n AC_hot.evt
    ```

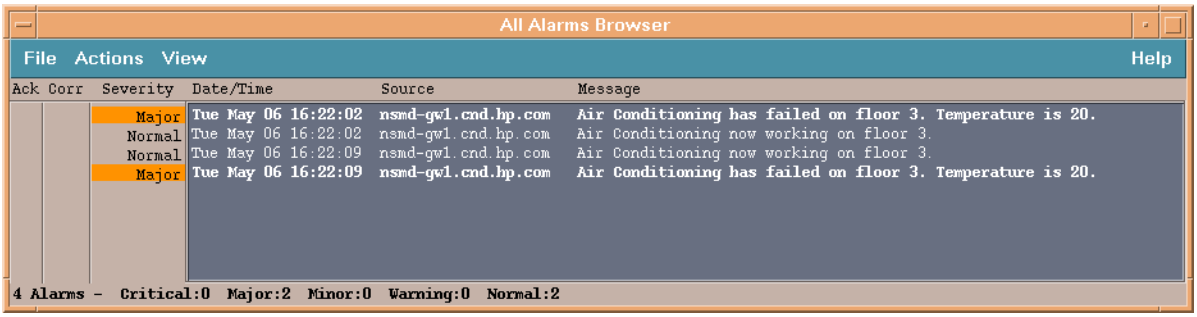
21. Demonstrate that your correlation creates a new event if the A/C unit goes up and down several times. Type the `ecsevgen` command two more times. (Not shown in results below.)

22. Demonstrate that your correlation does not suppress the down event if the room is not overheating. Type the command:

    ```
    ecsevgen -n AC_cool.evt
    ```

23. Demonstrate that your correlation does not suppress the events if the down arrives before the up event. Type the command:

```
ecsevgen -n AC_reverse.evt
```



Chapter 17, Relating Events from Multiple Sources

Lab requires:

- Multi Source Correlator

Requirements

Assume a wide area network with many small networks interconnected by routers (primary and secondary routers to share the load). Some sub-networks are connected by one router (primary) as shown in the slide.

If primary router is down and if a secondary router is connected, then all the traffic is diverted through secondary router. Due to heavy traffic, the secondary router will generate many “Traffic-Threshold-Breached” traps. The requirements are:

- When the Primary Router down trap is emitted, discard all the “Traffic-Threshold-Breached” traps from the secondary router for 30 seconds.
- Output the original Primary Router Down trap.
- Make sure that the trap sent is from the same primary and secondary router connecting the same 2 networks.

The router is uniquely identified by a combination of enterprise ID + Router ID (varBind[0] value). If the secondary router is connected, then only from those routers, the threshold alarms should be discarded.

Whether the secondary router is available or not is stored in the datastore. Passing the unique router as a key to the datastore can identify it.

The dataStore looks like:

```
ADD DATA ("<enterpriseid + Routerid>", "<Yes/No>" )
```

Trap Definitions

A router down trap is identified by

- enterprise id set to 1.5.6.7.8
- specific-trap set to 8
- var-bind0 contains router id

Example Trap-PDU

```
Trap-PDU {
    enterprise {1 5 6 7 8},
    agent-addr internet : "\x7f\x00\x00\x02",
    specific-trap 8,
```

```
    time-stamp 414746291,
    variable-bindings {
{
    name {1 3 6 1 4},
    value simple : number : 10
}
}
}
```

A Threshold breached trap is identified by

- enterprise id set to 1.5.6.7.8
- specific-trap set to 9
- var-bind0 contains router id of the primary router it is backing up

Example Trap-PDU

```
Trap-PDU {
    enterprise {1 5 6 7 8},
    agent-addr internet : "\x7f\x00\x00\x03",
    specific-trap 9,
    time-stamp 414746291,
    variable-bindings {
{
    name {1 3 6 1 4},
    value simple : number : 10
}
}
}
```

View Pre-Correlator Results

1. A sample event stream from this environment is available for testing. Copy the following file from `$OV_CONTRIB/OVTraining/NNM3` to `$OV_CONTRIB/ecs`:
   - `secondaryRouter.evt`: sends the following events:

     Router down

     f

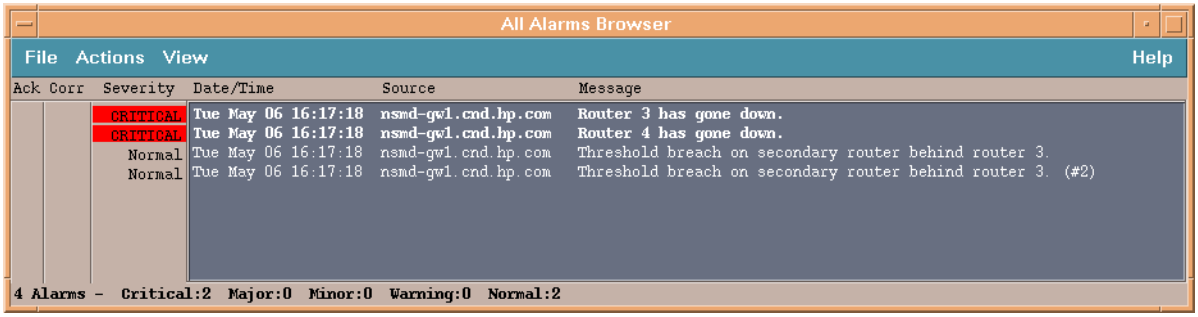
     — rom a router that has a secondary.

     — Router down from a router that does not have a secondary.

     — Threshold breach from secondary router (2 times).
2. Open the Composer Labs Alarms Browser.
3. Empty all old alarms by selecting `Actions:Delete->All Alarms in Category`.
4. Type the commands:

```
cd $OV_CONTRIB/ecs

ecsevgen -n secondaryRouter.evt
```

5. Review the alarms in the Alarm Browser.



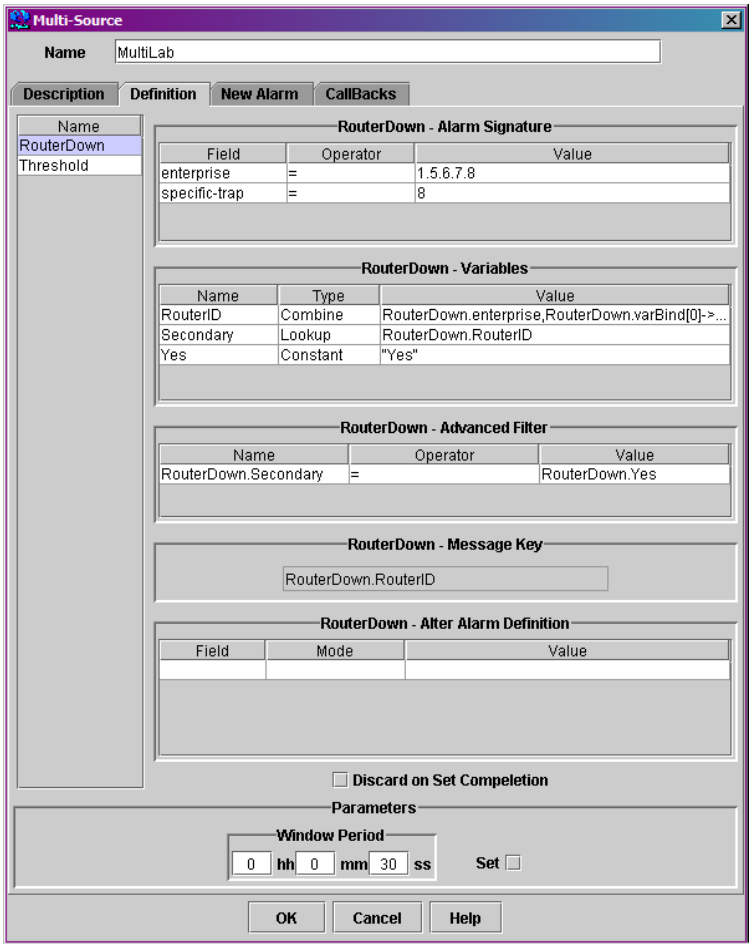
6. To mark which alarms came from this pre-test run, select `Actions:Acknowledge->All Alarms in Category`.

Directions

Execute the following steps to solve the exercise:

7. Use the Multi-Source Correaltor Template.
8. Create two types of events to filter:
   a. Router Down traps
   b. Threshold Breach traps
9. Specify the Alarm Signature for each type of event.
10. A router is uniquely identified by the combination of enterprise ID + Router ID (varBind[0] value). Use a Combine variable and combine these attributes for each type of event.
11. Create a Lookup variable and pass the unique id (created in last step), to check for secondary router.
12. In the Advanced Filter section, check for secondary router's availability.
13. In the Message Key section, specify the unique router id (which is created in step 10) for each type of event. This will correlate the alarms from the same router together.
14. To discard threshold traps when the set is completed, check `Discard on Set Completion` box for Threshold Breach events .

15. Specify the Window Period of 30 seconds.



16. Save your Correlator Store in the `$OV_CONF/ecs/CIB` directory.

    Answer:

    a. Select `File:Save As.`

    b. Browse to `$OV_CONF/ecs/CIB` and name your file `multiLab`. The .fs extension is added automatically.

17. Exit the Composer developer interface.

18. Add your data to the datastore.

    Answer:

    Place the ADD DATA lines in `Composer.ds`.

    For example, the datastore will have values like:

```
ADD DATA ("1.5.6.7.81", "Yes")
ADD DATA ("1.5.6.7.82", "No")
ADD DATA ("1.5.6.7.83", "Yes")
ADD DATA ("1.5.6.7.84", "No")
```

    Note: If you have data with a similar key from previous exercises, delete those lines.

    Disable and enable Composer from the ECS GUI to have it read your file changes.

19. Provide Operator access to your correlator.

    Answer:

    a. Edit the `$OV_CONF/ecs/CIB/NameSpace.conf` file.

    b. At the bottom of the file add a line giving a short name and the path to your Correlator Store (relative to `$OV_CONF/ecs/CIB`).

```
Lab_Multisource=multiLab.fs
```

20. Deploy your correlation.

    Answer:

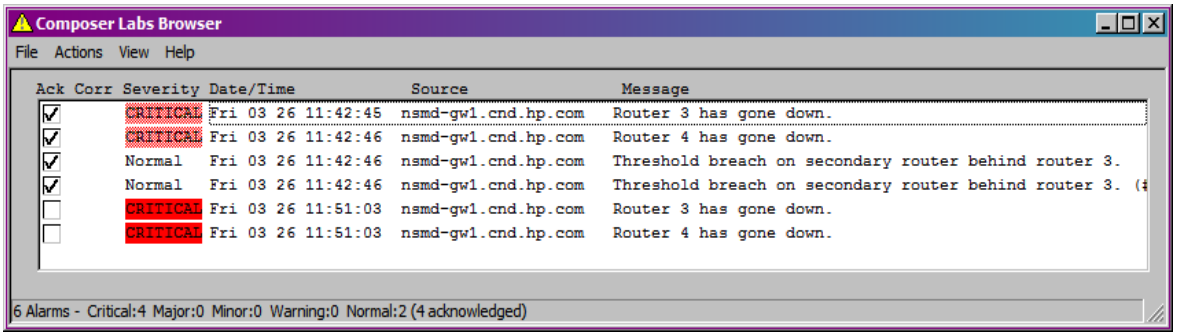
    Type `csdeploy.ovpl`.

Test Your Correlator

21. Type the commands:

```
cd $OV_CONTRIB/ecs
ecsevgen -n secondaryRouter.evt
```

22. Examine the All Alarms Browser. You should see only the first two events come through. .



Chapter 18, Using Callbacks and Built-In Functions

Lab requires:

- Multi Source Correlator
- Discard Callback
- Built-in Functions storeStr, retrieveStr
- Create Callback
- Feedback

Requirements

NOTE This lab builds on the lab for MultiSource. If you have already done that lab, you may use it as the basis for this lab.

Assume a wide area network with many small networks interconnected by routers (primary and secondary routers to share the load). Some sub-networks are connected by one router (primary) as shown in the slide.

If primary router is down and if a secondary router is connected, then all the traffic is diverted through secondary router. Due to heavy traffic, the secondary router will generate many "Traffic-Threshold-Breached" traps. The requirements are:

- When the Primary Router down trap is emitted, discard all the "Traffic-Threshold-Breached" traps from the secondary router for 30 seconds.
- Output the original Primary Router Down trap.
- Make sure that the trap sent is from the same primary and secondary router connecting the same 2 networks.
- The other requirement is to concatenate and store all the varBind[0] values of the discarded traps and create a new event after the window period. The new event should contain concatenated string of all the varBind[0] values of the discarded traps.

The router is uniquely identified by a combination of enterprise ID + Router ID (varBind[0] value). If the secondary router is connected, then only from those routers, the threshold alarms should be discarded. Whether the secondary router is available or not is stored in the datastore. Passing the unique router as a key to the datastore can identify it.

The data store looks like:

```
ADD DATA ("<enterpriseid + Routerid>", "<Yes/No>" )
```

Trap Definitions

A router down trap is identified by

- enterprise id set to 1.5.6.7.8
- specific-trap set to 8
- var-bind0 contains router id

Example Trap-PDU

```
Trap-PDU {
    enterprise {1 5 6 7 8},
    agent-addr internet : "\x7f\x00\x00\x02",
    specific-trap 8,
    time-stamp 414746291,
    variable-bindings {
{
    name {1 3 6 1 4},
    value simple : number : 10
}
}
}
```

A Threshold breached trap is identified by

- enterprise id set to 1.5.6.7.8
- specific-trap set to 9
- var-bind0 contains router id of the primary router it is backing up

Example Trap-PDU

```
Trap-PDU {
    enterprise {1 5 6 7 8},
    agent-addr internet : "\x7f\x00\x00\x03",
    specific-trap 9,
    time-stamp 414746291,
    variable-bindings {
{
    name {1 3 6 1 4},
    value simple : number : 10
}
}
}
```

View Pre-Correlator Results

1. A sample event stream from this environment is available for testing. Copy the following file from `$OV_CONTRIB/OVTraining/NNM3` to `$OV_CONTRIB/ecs`:
   - `secondaryRouter.evt`: sends the following events:
     — Router down from a router that has a secondary.

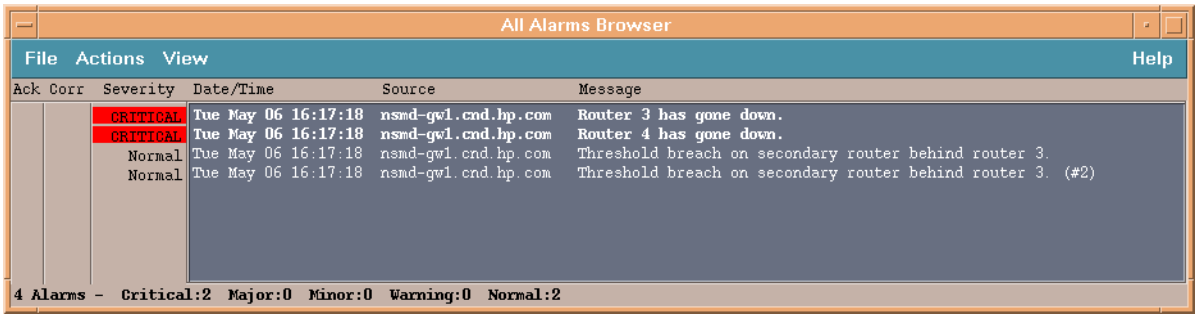
— Router down from a router that does not have a secondary.

— Threshold breach from secondary router (2 times).

2. Open the Composer Labs Alarms Browser.
3. Empty all old alarms by selecting `Actions:Delete->All Alarms in Category`.
4. Type the commands:

```
cd $OV_CONTRIB/ecs

ecsevgen -n secondaryRouter.evt
```

5. Review the alarms in the Alarm Browser.



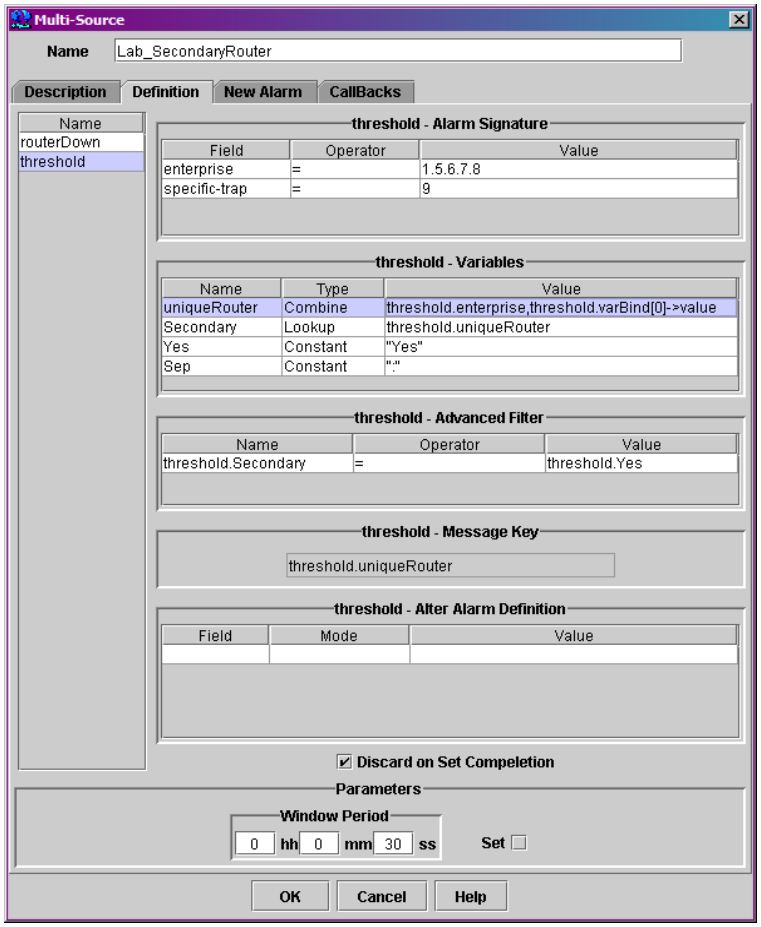
6. To mark which alarms came from this pre-test run, select `Actions:Acknowledge->All Alarms in Category`.

Directions

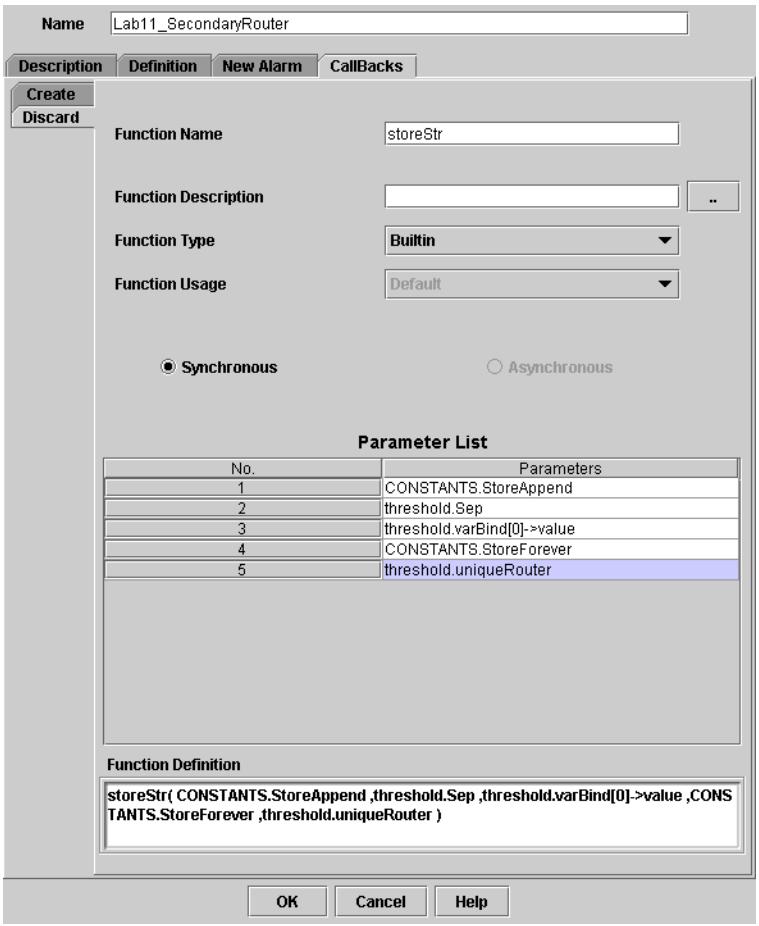
Execute the following steps to solve the exercise:

7. Use the Multi-Source Correaltor Template.
8. Create two types of events to filter:
   a. Router Down traps
   b. Threshold Breach traps
9. Specify the Alarm Signature for each type of event.
10. A router is uniquely identified by the combination of enterprise ID + Router ID (varBind[0] value). Use a Combine variable and combine these attributes for each type of event.
11. Create a Lookup variable and pass the unique id (created in last step), to check for secondary router.
12. In the Advanced Filter section, check for secondary router's availability.
13. In the Message Key section, specify the unique router id (which is created in step 10) for each type of event. This will correlate the alarms from the same router together.
14. To discard threshold traps when the set is completed, check `Discard on Set Completion` box for Threshold Breach events .

15. Specify the Window Period to 30 seconds.



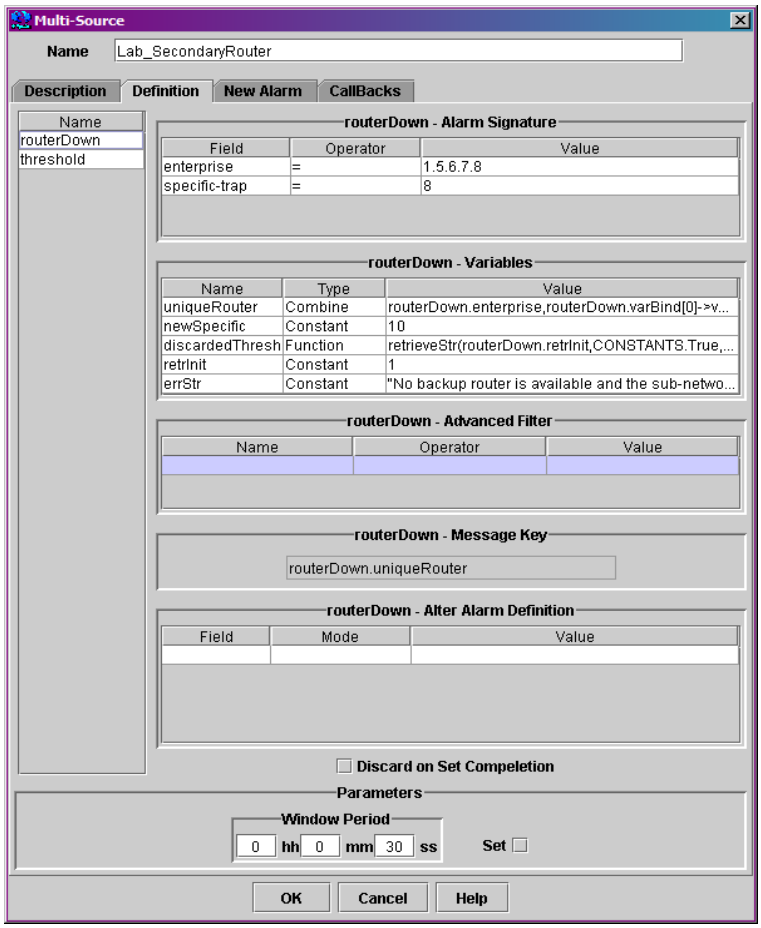
16. To concatenate all the varBind [0] values of the discarded traps, use Discard Callbacks section and call the Built-in function `storeStr` and pass the varBind[0] value. You will need constants defined (either Globally or within the variables table) to instruct the built-in to append your value to the storage space and to store it forever. Review the manual online for the values to use.

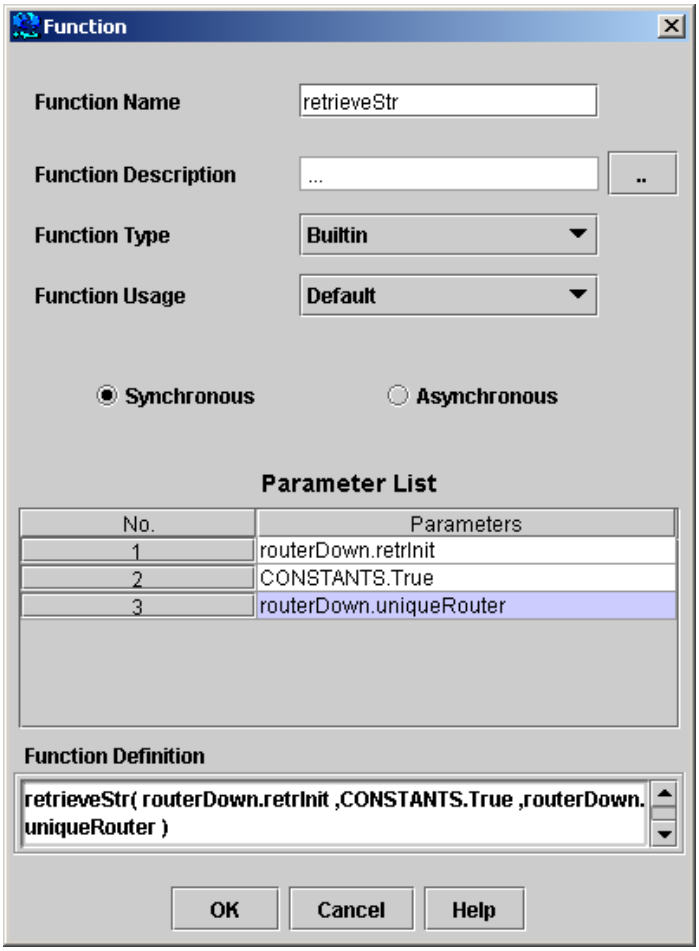


Answer:

The constant values to use are 0 for StoreAppend and -1 for StoreForever.

17. To retrieve the concatenated string of all the varBind [0] values of the discarded traps, use function variable type and call the Built-in function `retrieveStr` by passing the same key values as specified for `storeStr` function.You will need a constant value of your choosing for the function to return if the data is not available for your key.

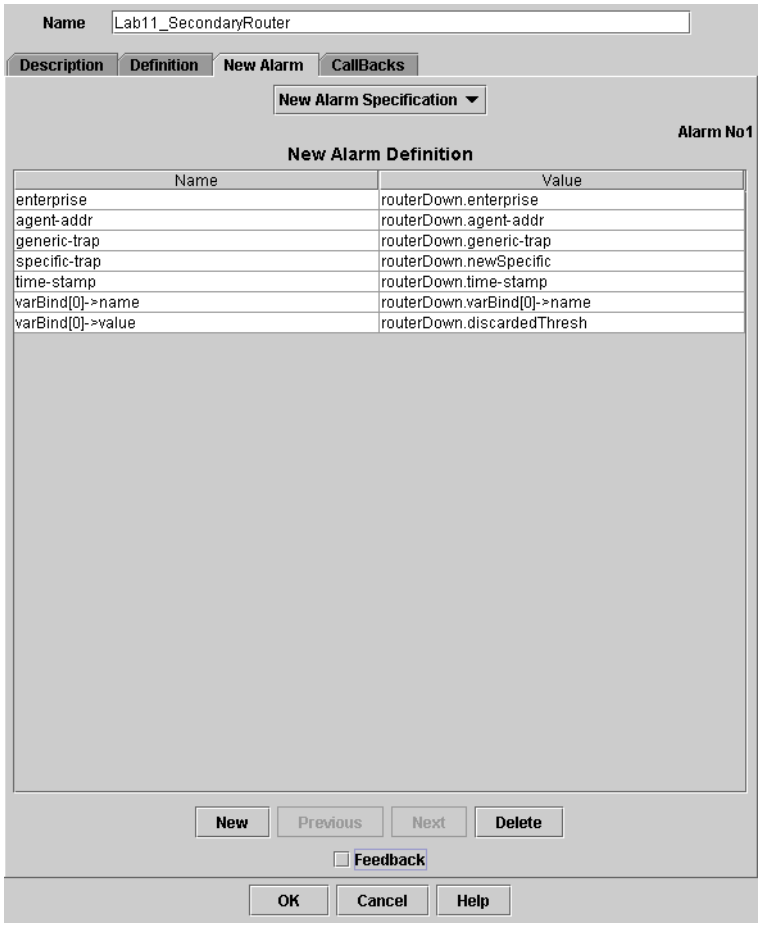




Answer:

The constant for True is 1. The value for RetrieveInitialize is 1.

18. Create New Alarm and bind the `retrieveStr` variable to any of the attributes to display all strings of all the varBind [0] values.



19. Save your Correlator Store in the `$OV_CONF/ecs/CIB` directory.

    Answer:

    a. Select `File:Save As.`

    b. Browse to `$OV_CONF/ecs/CIB` and name your file `functionLab`. The .fs extension is added automatically.

20. Exit the Composer developer interface.

21. Add your data to the datastore.

    Answer:

    Place the ADD DATA lines in `Composer.ds`.

    For example, the datastore will have values like:

```
ADD DATA ("1.5.6.7.81", "Yes")

ADD DATA ("1.5.6.7.82", "No")

ADD DATA ("1.5.6.7.83", "Yes")

ADD DATA ("1.5.6.7.84", "No")
```

    Note: If you have similar data from a previous lab, delete those lines.

Disable and enable Composer from the ECS GUI to have it read your file changes.

22. Provide Operator access to your correlator.

    Answer:

    a. Edit the `$OV_CONF/ecs/CIB/NameSpace.conf` file.

    b. At the bottom of the file add a line giving a short name and the path to your Correlator Store (relative to `$OV_CONF/ecs/CIB`).

    ```
    Lab_Function=functionLab.fs
    ```

23. Deploy your correlation.

    Answer:

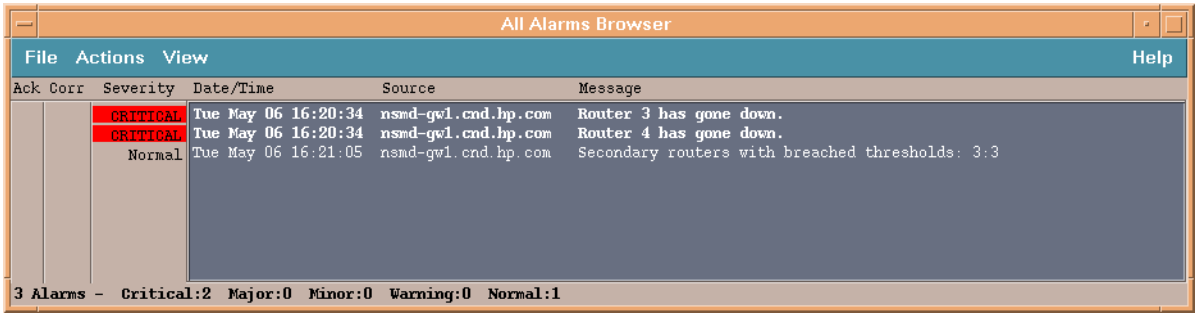
    Type `csdeploy.ovpl`.

Test Your Correlator

24. Type the commands:

    ```
    cd $OV_CONTRIB/ecs
    ecsevgen -n secondaryRouter.evt
    ```

25. Examine the All Alarms Browser. You should see the first two events come through. After the timeout, you will see a new event with the router IDs from the two suppressed threshold traps.



Chapter 19, Best Practices and Tools

1. Review the contents of your current Binary Event Store to see if you have any candidates for correlation or de-duplication.

   a. Use the following commands to see the contents of the BES:

      1. `cd /opt/OV/support`.
      2. `ovdumpevents -s "default" >eventStoreDump`
      3. `ovdumpevents -c "default" >correlationLogDump`
      4. Briefly review the files to see their formats.

   b. (UNIX only) Create the helper files for the analysis tools

      1. Create the `logonly` file using the command

         ```
         grep 'LOGONLY' $OV_CONF/C/trapd.conf | cut -d ' ' -f 3 | grep '17\.1' | sed -e \
         's/\.1\.3\.6\.1\.4\.1\.11\.2\.//' >logonly
         ```

      2. Create the `ov_events` file using the command

         ```
         grep 'OV_' $OV_CONF/C/trapd.conf | cut -d ' ' -f 2-5 | \
         grep '^OV_' > ov_events
         ```

   c. (UNIX only) Analyze the event log.

      1. `cd /opt/OV/support`
      2. `./processEvents eventStoreDump summaryOutput`
      3. Execute `more summaryOutput`.
      4. What are your top 5 events by frequency?

   d. (UNIX only) Analyze the correlation log.

      1. `cd /opt/OV/support`
      2. `./processCorrEvents correlationLogDump summaryCorrResults`.
      3. What are the top 5 already being de-duplicated in `summaryCorrResults`?
      4. What are the top 5 events already being correlated?
      5. How would you use these two files together in designing correlators?

         Answer:

         Look for correlation candidates. However, if those same events are already being correlated, ensure that your correlator does not break the existing correlation.

2. The file `$OV_CONTRIB/OVTraining/NNM3/MultipleReboot.evt` simulates the capture of a series of starup events from a connector. Examine `MultipleReboot.evt` to see what the capture looks like. How many events are in the file?

3. Turn on tracing. Run `MultipleReboot.evt` to see how the tracing works. Note when the functions get evaluated.

a. Turn on tracing.

   1. Ensure that your Composer and ECS configuration GUIs are closed.
   2. Load the debugging fact store for Composer by running `ecsmgr -fact_update Composer $OV_CONTRIB/ecs/CO/CompTraceOn.fs`.
   3. `ecsmgr -i 1 -trace 65536`
   4. UNIX: `pmdmgr -Secss\;T0xffffffff -Qt -Ql`

      Windows: `pmdmgr -Secss;T0xffffffff -Qt -Ql`

b. Run `MultipleReboot.evt`.

   1. Open your All Alarms Browser.
   2. `cd $OV_CONTRIB/ecs`
   3. `$OV_CONTRIB/ecs/ecsevgen -n MultipleReboot.evt`
   4. What did you see in the Alarms Browser?

c. Review the trace file in `$OV_LOG/pmd.trc0`.

d. (UNIX only) Find the trace messages relevant to Composer and this correlator by typing `grep 'Composer' $OV_LOG/pmd.trc0 | grep 'OV_MultipleReboots'`.

4. Modify MultipleReboot correlator to have `count=5` and rerun `MultipleReboot.evt`. Run it again. Return `count=4`, the original value.

   Answer: When the count is 5, the first test run does not exceed the threshold. The second test run breaches the threshold and causes the event.

5. Run the unit test `NodeIF.evt` and monitor the trace output to see how the correlators work together.

   a. Clear the `pmd` trace file using `pmdmgr -Secss\;T0xffffffff -Qt -Ql` (UNIX) or `pmdmgr -Secss;T0xffffffff -Qt -Ql` (Windows).

   b. Open your All Alarms Browser.

   c. `$OV_CONTRIB/ecs/ecsevgen -n NodeIf.evt`

   d. Explain what you see in your Alarm Browser.

   e. (UNIX only) Find the trace messages relevant to Composer and this correlator by typing `grep 'Composer' $OV_LOG/pmd.trc0 | grep 'OV_NodeIf'`. Which actual correlators were activated?

6. Turn off tracing and remove the debugging Composer.fs.

   a. `ecsmgr -i 1 -trace 0`

   b. UNIX: `pmdmgr -Secss\;T0x0`

      Windows: `pmdmgr -Secss;T0x0`

   c. `ecsmgr -fact_update Composer $OV_CONTRIB/ecs/CO/CompTraceOff.fs`

7. Last: Delete all class-created correlators before continuing class.

Chapter 21, Configuring syslog Messages for SNMP

These exercises are designed to assist you in understanding the operation of the syslogTrap component of NNM.

1. Enable syslog in a stand-alone NNM environment (no OVO).

   Answer:

   The command `$OV_BIN/setupSyslog.ovpl -standalone` installs the OVO agent, generates the syslog template, and starts `syslogTrap` NNM process. (Go get coffee.)

2. The `$OV_CONTRIB/OVTraining/NNM3` directory contains a file of new patterns (`patterns`). You also have a sample data input line in the file `value`. See how this value is parsed by the various patterns by using `opcpat`.

   Answer:

   a. `cd $OV_CONTRIB/OVTraining/NNM3`

   b. `/opt/OV/bin/OpC/utils/opcpat -fp patterns -fv value`

   c. Note how the input value string is assigned to the various segments of the pattern matching and the tagged variables. The input value string actually only matches the first and last patterns tested.

3. Launch the syslog configuration GUI and browse the default conditions.

   Answer:

   a. `$OV_BIN/ovsyslogcfg` starts the syslog configuration GUI.

   b. Select the condition, then click "`Modify`…" to view Description and Condition Text for each one.

4. Test by injecting a `linkdown` syslog message which emits a trap.

   Answer:

   a. Launch NNM All Alarms Browser (from Home Base or `ovw`).

   b. `cd $OV_CONTRIB/OVTraining/NNM3`

   c. `linkdown_test` sends a `linkdown` message to `/var/adm/syslog/syslog.log`. The event will appear in the browser (be patient).

5. Using the Cisco web-site for syslog messages: http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/sem/emabout.htm, map SYS-5-CONFIG to a new trap.

   Answer:

   a. According to the section "How to Read System Error Messages," each message begins with a %, followed by the code for the facility, the severity, and the mnemonic for the error message. In our case a SYS-5-CONFIG message would appear as:

   ```
   %SYS-5-CONFIG: message text
   ```

b. In the syslog configuration GUI, click [Add] to add a new pattern.

c. Since we want to map it to a new trap, we don't want to suppress it, so the Condition Type is Message on match.

d. Using the pattern for LINKUP as a guide, set your Condition Text to

```
<*><_><*><_><*><_><@.node><_><@.pid><*>
```

e. NNM event identifiers 1 through 10,000 are available for you to use. For the new trap, use the OpenView enterprise (1.3.6.1.4.1.11.2.17.1), the enterpriseSpecific generic identifier, and specific identifier 2001.

f. Add a varbind with OID 1.3.6.1.4.1.11.2.17.2.1, value 33, type Integer.

g. Add a varbind with OID 1.3.6.1.4.1.11.2.17.2.2, value <node>, type Octet String. This places the node name in the second varbind, which is customary for NNM.

h. Save your changes.

6. Deploy your configuration.

   Answer:

   `setupSyslog.ovpl -standalone -deploy` deploys new template created via the syslog configuration GUI.

7. Use `logger` to test your pattern and watch for the result in the Alarm Browser.

   Answer:

   From the command line, type `logger %SYS-5-CONFIG: something terrible happened!`

   If your pattern is matched, the Alarm Browser shows "Received event .1.3.6.1.4.1.11.2.17.1.0.2001..."

8. Use `Options:Event Configuration` to recognize event 2001 and display the node name where the failure occurred.

   Answer:

   a. Name the event SyslogLab.

   b. Use Event Object Identifier .1.3.6.1.4.1.11.2.17.1.0.2001.

   c. Set the Category to Configuration Alarams.

   d. Make the message say "System configuration trap severity 5 on Cisco device $2.

   e. Click [`OK`] and select `File:Save`.

9. Retest your event using `logger`.

Chapter A, Viewing Your Environment with Dynamic Views

1. Start Home Base.

   a. What other views can be launched from here?

      Answer:

      You can go to Internet, Network, Segment, Neighbor, Node, Path View. If the Extended Topology software is enabled, you also have access to views for VLANs, OSPF, HSRP, Overlapping Address Domains and Problem Diagnosis.

   b. Launch the "Internet" View. What menus are available from this View?

      Answer:

      Select Internet View and [`Launch View`].

      This launches the Internet Dynamic View. The menus available are: File, Edit, View, Performance, Configuration, Fault, Tools, Options and Help.

   c. From the Tools menu, select the Views submenu. What Views are available from this menu?

      Answer:

      The same views are available from Dynamic Views menus as from Home Base.

      At the `ovw` Root submap, the only view available is Home Base. On other `ovw` submaps, the views available are: Neighbor, Path, and Home Base.

   d. From the Internet View, double click the symbol for your network. What is the result?

      Answer:

      The result is a "Tabular" view of the segments in that network.

   e. Click the + to open a segment. Double click one of the segment names listed. What are the results of this selection?

      Answer:

      This results in a submap of that segment.

   f. Take a few minutes to explore this navigation of elements.

2. Visit a view that uses the Table Presenter (e.g. Network View) and do sorting, etc. Note that when you restart the view all customizations are gone.

   Answer:

   a. From Home Base, select Network View and click [`Launch View`].

   b. Click [`Refresh`] to start the view.

   c. Click the + in front of the first segment.

   d. Click the column header for IP Address.

A defect has been filed against Active Tables on Windows (fixed in 7.01). This step currently only works on UNIX.

e. Reopen the segment and notice the sorting on IP Address.

f. Click on the column header for Node Status.

g. Close the view.

h. Relaunch the view from Home Base. Notice that the devices are sorted by node name now.

3. Go to a Path View and use Expand Neighbors.

   Answer:

   a. Select Path View and click [`Launch View`].

   b. Enter your management station as the Source Node and a device name and click [`Refresh`].

   c. Select one end of the path and select `View:Expand Connecting Neighbors`. If there are none, expand the neighbors of a connecting device in the view.

4. Configure poster printing.

   Answer:

   a. Open the Internet View.

   b. Select `File:Poster Print Options`.

   c. Set the number of Poster Rows to 3.

   d. Set the number of Poster Columns to 4.

   e. Click [`OK`].

   f. Select `File:Print Preview`.

   g. Click [`Next`] to look at each sheet that would print.

   h. Click [`Close`].

5. Turn port labels on and off.

   Answer:

   a. Start Segment View for segment 1.

   b. Select `View:Labels->Toggle Port Labels`.

   c. You may need to set the display labels to long names and back for this to show.

6. Close all of the Dynamic Views windows except Home Base. Then open the All Alarms browser window from the Home Base Alarms tab.

   a. Select an alarm from your system. Then using `Actions:Views`, open up a Neighbor View.

      Answer:

      Select an alarm with a single click. Then select `Actions:Views -> Neighbor`.

   b. Experiment with the number of hops and showing end nodes. How does this influence the display?

      Answer:

      Increasing the number of hops and adding end nodes adds items to the display. To have each change take effect, select the [`Refresh`] button in the window.

Chapter B, Securing Dynamic Views

Use the demo or classroom topology to complete the following lab exercises.

NOTE If you have not enabled Extended Topology, run `dvUsersManager.ovpl` before starting the lab.

1. Enable existing roles and users to require a password for access to all dynamic views.

   Answer:

   a. UNIX: `cd $OV_AS/webapps/topology/WEB-INF`.

      Windows: In Windows Explorer, browse to *install_dir*\tomcat\jakarta-tomcat-4.04\webapps\topology\WEB-INF.

   b. Make a backup copy of `web.xml` and `dynamicViewsUsers.xml`.

   c. Edit `web.xml`. Near the bottom of the file, remove the commenting-out lines around the second block of security constraints. Save the file.

```
 <!--
      Uncomment this entry to configure authentication on all servlets
      or you can specify individual servlets in the <url-pattern> to
      require login on a subset of servlets.
    -->

<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>
```

   d. Edit `dynamicViewsUsers.xml`. Add at least one user in each group (operator and administrator) defined in `web.xml`. Save the file.

```
<tomcat-users>
     <user name="admin1" password="admin1" roles="administrator" />
     <user name="oper1" password="oper1" roles="operator" />
</tomcat-users>
```

   e. Open each file from your web browser to verify the XML syntax.

   f. `ovstop ovas` and `ovstart ovas`.

   g. Start Home Base. Enter your user and password combination for the Operator role. Notice that if you have Extended Topology enabled, the Extended Topology discovery and polling status tabs do appear. However, if you select the configuration button, access is denied.

2. Create a role named specialist and add a user for it. You need to do these two steps together because `ovas` requires a user for each defined role. The specialist should be allowed to add and delete nodes in Dynamic Views.

Answer:

a. UNIX: `cd $OV_AS/webapps/topology/WEB-INF`.

Windows: In Windows Explorer, browse to *install_dir*\tomcat\jakarta-tomcat-4.04\webapps\topology\WEB-INF.

b. Edit `web.xml`.

1. Near the bottom of the file, add an additional security constraint block. You can start by copying the original block that was active in the file.
2. Keep the url-patterns for add and delete, but remove the other ones.
3. In the auth-constraint block, change the role-name from administrator to specialist.

```
<security-constraint>
<!-- Only allow administrators to perform configuration operations -->
<web-resource-collection>
    <web-resource-name>Dynamic View Admin Access</web-resource-name>
    <url-pattern>/add/*</url-pattern>
    <url-pattern>/delete/*</url-pattern>
</web-resource-collection>
<auth-constraint>
    <role-name>specialist</role-name>
</auth-constraint>
</security-constraint>
```

4. Save the file.

c. Edit `dynamicViewsUsers.xml`.

1. In the list of tomcat-users at the bottom of the file (not commented out), make a copy of the administrator line.
2. In the copy line, change the user and password to special1.
3. Change the role from administrator to specialist.

```
<tomcat-users>
     <user name="admin1" password="admin1" roles="administrator" />
     <user name="oper1" password="oper1" roles="operator" />
     <user name="special1" password="special1" roles="specialist" />
</tomcat-users>
```

4. Save the file.

d. Open each file from your web browser to verify the XML syntax.

e. `ovstop ovas` and `ovstart ovas`.

f. Start Home Base. Enter the specialist user and password combination.

Your access is denied because the specialist only has access to add and delete, not to the rest of the dynamic views.

g. Edit `web.xml` and add the specialist role to the security constraint which has access to all URLs.

1. Inside the auth-constraint for the /* group of URLs, add a role-name line for specialist.

```
<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>
        <role-name>specialist</role-name>
    </auth-constraint>
    </security-constraint>
```

2. Save the file.
3. Stop and restart `ovas`.

h. Start Home Base and log in as specialist.

3. Create a user, super, who can do everything an administrator can do and everything a specialist can do.

Answer:

a. Edit `dynamicViewsUsers.xml`.

1. In the list of tomcat-users at the bottom of the file (not commented out), make a copy of the administrator line.
2. In the copy line, change the user and password to super1.
3. Change the role from "administrator" to "administrator,specialist".

```
<tomcat-users>
     <user name="admin1" password="admin1" roles="administrator" />
     <user name="oper1" password="oper1" roles="operator" />
     <user name="special1" password="special1" roles="specialist" />
     <user name="super1" password="super1" roles="administrator,specialist" />
</tomcat-users>
```

4. Save the file.

b. Open each file from your web browser to verify the XML syntax.

c. `ovstop ovas` and `ovstart ovas`.

d. Start Home Base. Enter the super user and password combination.

4. Configure encryption.

Answer:

a. Get an encrypted form of the administrator password.

UNIX: `"$OV_JRE"/bin/java –classpath \ "$OV_AS"/server/lib/catalina.jar org.apache.catalina.realm.RealmBase -a MD5 admin1`

Windows: `"%OV_JRE%"\bin\java –classpath \ "%OV_AS%"/server/lib/catalina.jar org.apache.catalina.realm.RealmBase -a MD5 admin1`

b. Copy the encrypted string that is returned (not the admin1: at the front) and paste it into `dynamicViewsUsers.xml` to replace the password for the admin1 user. (Don't replace the user name, just the password.)

c. Save the file.

d. Edit `server.xml` found in

UNIX: `$OV_AS/conf`

Windows: `%OV_AS\conf`

e. Near the bottom of the file, find the block that begins with <Context path="/topology".

f. In the Realm element, after the className definition, add digest="MD5". Your final block

should look like:

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

g. Stop and start `ovas`.

h. Open Home Base and log in as admin1 with password admin1.

i. Try to log in with another user and password which is not encrypted it in the file.

Chapter C, Using Problem Diagnosis

Extended Topology must be enabled for these exercises.

1. Start the Problem Diagnosis View from Home Base. Select your system as the Probe.

   Answer:

   a. From Home Base, select Problem Diagnosis View and [Launch View].

   b. For the Probe, select your management station from the drop-down list.

2. Add an endpoint to monitor by clicking [CONFIGURE].

   Answer:

   a. Click [CONFIGURE] to add a target.

   b. Click [Add].

   c. Edit the table of information for the new target. You may use the Internet View or ovw to assist in selecting a node on a different segment from your own.

   d. Click [OK].

   e. Your new target appears as the default endpoint.

3. View the Path List.

   Answer:

   a. On the Endpoints page, click [GO!}.

   b. Notice the list of paths, where the top one is the current path. Any path that matches it has an M in the last column.

4. View path information for the Current Path.

   Answer:

   a. Double-click the Current path to see the Path Map.

   b. Select the Path Detail tab to see the response times.

5. View the Trek detail.

   Answer:

   a. Select the Trek Detail tab to see the information that Problem Diagnostics keeps about its operation.

Chapter D, Configuring Problem Diagnosis

Extended Topology must be enabled to use Problem Diagnosis. If you have not already done so, run `setupExtTopo.ovpl`. Answer yes to all questions and use "ov" for the requested user and password.

1. Stop and restart the server

   Answer:

   a. Close all Problem Diagnostics Views.

   b. Type `ovstop pd`.

   c. Type `ovstart pd`.

2. Link the server to a probe on another system

   Answer:

   a. Edit the `pdconfig.xml` file located in
   UNIX: `/opt/OV/pdAE/config/`
   Windows: `install_dir\pdAE\config\`

   b. Scroll down to the PROBE_LIST. The first probe is the one on your management station and the list of DESTINATIONs reflecting the targets you configured.

   c. Copy the entire probe block from <PROBE> through </PROBE> and insert it below the current probe, still within the </PROBE_LIST>

   d. Modify the copied block to point to another system in the classroom.

   e. Modify the copied block to remove all destinations. You can configure targets for this new probe from the Endpoints configuration interface.

   f. Your final file will resemble the following. In this example, our server is ovt2.cnd.hp.com and the additional probe is on ovt4.cnd.hp.com.

```
<?xml version="1.0" encoding="UTF-8"?>
<PD_CONFIG>
<MY_HOST_NAME>ovt2.cnd.hp.com</MY_HOST_NAME>
<MY_IP_ADDRESS>15.11.68.102</MY_IP_ADDRESS>
<DEBUG>false</DEBUG>
<STDERR>false</STDERR>
<EXTERNAL_DB>true</EXTERNAL_DB>
<EXTERNAL_PROBE>false</EXTERNAL_PROBE>
<BROWNOUT_NUM_SAMPLES>15</BROWNOUT_NUM_SAMPLES>
<BROWNOUT_BAD_SAMPLES>8</BROWNOUT_BAD_SAMPLES>
<BROWNOUT_NUM_DEVIATIONS>3</BROWNOUT_NUM_DEVIATIONS>
<BROWNOUT_INTERVAL>86400000</BROWNOUT_INTERVAL>
<BUCKET_SIZE>24</BUCKET_SIZE>
<LOG_MAX>48</LOG_MAX>
<HTTP_PORT>8068</HTTP_PORT>
<TCP_PORT>8069</TCP_PORT>
<PROXY_SERVER>null</PROXY_SERVER>
<PROXY_PORT>0</PROXY_PORT>
<HTTP_TIMEOUT>50000</HTTP_TIMEOUT>
<ET_SERVERS>ovt2.cnd.hp.com</ET_SERVERS>
<PROBE_LIST>
```

```
<PROBE>
<HOST_NAME>ovt2.cnd.hp.com</HOST_NAME>
<IP_ADDRESS>15.11.68.102</IP_ADDRESS>
<TCP_PORT>8066</TCP_PORT>
<HTTP_PORT>8067</HTTP_PORT>
<DESTINATION_LIST>
<DESTINATION>
<HOST_NAME>10.96.26.235</HOST_NAME>
<FREQUENCY>5</FREQUENCY>
<DNS>true</DNS>
<LEVEL_TWO>true</LEVEL_TWO>
<BROWNOUT>true</BROWNOUT>
<RETRIES>3</RETRIES>
<TIMEOUT_LIMIT>5</TIMEOUT_LIMIT>
<TREAT_TIMEOUTS>ERROR</TREAT_TIMEOUTS>
</DESTINATION>
<DESTINATION>
<HOST_NAME>10.96.26.205</HOST_NAME>
<FREQUENCY>5</FREQUENCY>
<DNS>true</DNS>
<LEVEL_TWO>true</LEVEL_TWO>
<BROWNOUT>true</BROWNOUT>
<RETRIES>3</RETRIES>
<TIMEOUT_LIMIT>5</TIMEOUT_LIMIT>
<TREAT_TIMEOUTS>ERROR</TREAT_TIMEOUTS>
</DESTINATION>
</DESTINATION_LIST>
</PROBE>
<PROBE>
<HOST_NAME>ovt4.cnd.hp.com</HOST_NAME>
<IP_ADDRESS>0.0.0.0</IP_ADDRESS>
<TCP_PORT>8066</TCP_PORT>
<HTTP_PORT>8067</HTTP_PORT>
</PROBE>
</PROBE_LIST>
```

g. Save your file.

h. Stop and restart the server to activate your changes.

i. Start a Problem Diagnosis View and select the other probe to perform a path analysis to your system.

3. Link this system's probe to another server

Answer:

a. Make a backup copy of the `npprobe.conf` file located in
UNIX: `/opt/OV/pdAE/config`
Windows: `install_dir\pdAE\config`

b. Edit the file. Comment out the lines that refer to the current management station by hostname and IP Address by placing a `#` at the front of the line.

c. Add new lines for a server in the classroom you want this probe to report to. You should add a line for the server by hostname and one for the server by IP address.

d. Your file should resemble the following:

```
# SERVER=PD_MY_HOST

# SERVER_IP=0.0.0.0

SERVER=ovt3.cnd.hp.com

SERVER_IP=15.11.68.103

SERVER_PORT=8068
```

e. Save your changes.

f. Stop and restart the probe.
UNIX: `/opt/OV/pdAE/bin/pdcentral.sh -stop`
`/opt/OV/pdAE/bin/pdcentral.sh -start`
Windows: Open the Services applet and select NetPath. Click Stop, then click Start.

Chapter E, Constructing Advanced Filters

Objective: The purpose of this lab is to review the fundamental concepts behind NNM filters.

Review Questions

1. What object attributes are never available for filtering in NNM?

   Answer:

   Customized attributes are never available for filtering. The list of filterable attributes does not include all of the default attributes in the database. The *Guide to Scalability and Distribution for Network Node Manager* has the complete list of filterable attributes.

2. List and briefly describe the filters available in NNM.

   Answer:

   The following are the filters available in NNM:

   - Discovery: Used to limit the scope of discovery.
   - Topology: Used to define the set of objects that is sent to a management station from a collection station.
   - Map: Used to customize the user display on a per map basis.
   - Persistence: Used as a performance tool to control which submaps are in memory when a map is opened.
   - Failover: Used to define the mechanism for picking up management of nodes in the absence of a primary system.
   - Important Node: Used to define special actions relating to Secondary Failures.
   - DHCP: Used to more effectively manage a dynamic address environment.

3. Define an AVA, list the types of AVAs and give a brief description.

   Answer:

   The AVA is the Attribute Value Assertion. This is the mechanism by which each attribute in a filter is evaluated. There are four AVAs:

   - Boolean: These evaluate either true or false. There is nothing in between.
   - Integer: These are numeric and can be evaluated based on logical operators such as equal, less than or greater than.
   - Enum: These are string AVAs, but limited to a predefined list of possibilities.
   - String: These are true string variables in that they may be any set of characters.

4. What is the proper way to use "wild cards" with the AVAs, and on what are these "wild cards" based?

   Answer:

   Wild cards are based upon Regular Expressions. With regard to how they are used, the

answer depends upon the specific AVA involved.

- For String AVAs, you must use the "like" operator (~) and the wild card must be in double quotes.
  For example: ("IP Hostname" ~ ".*ovt5.*")
- For IP Addresses, you must use the "like" operator (~) but the wild card cannot be in double quotes.
  For example: ("IP Address" ~ 156.153.206.*)

5. What tools are available for testing filters, and what does each one test?

   Answer:

   There are three tools available for testing filters:

   - `ovfiltercheck` This tool verifies the syntax of the filter, and may be applied to the entire `filters` file. This is critical since a filter will not function if the syntax is not correct.
   - `ovfiltertest` This tool evaluates the filter against the topology or object database. It has a number of options that allow you to define the scope of the test. It shows you what objects in the database pass the filter.
   - `ovtopodump` This tool evaluates the filter against the topology database. It shows you what elements in the topology database will pass the filter. This tool is also the same tool that is used to read information out of the topology database.

6. Using the partial listing of hosts and IP Addresses shown below, define a filter definition for each of the following:

   Example host information:

```
jim.hp.com.uk  192.6.249.21
jim.hpicome    192.6.249.22
jim.hpicom     192.6.249.23
jim.hpcom      192.6.249.24
jim.hp.come    192.6.249.25
```

   a. Pass only hosts in the hp.com domain.

      Answer:

      The key is proper use of Regular Expressions. While there are several possible solutions, here is one possibility:

      The filter and AVA to pass only hosts from the hp.com domain:

```
hpDomain "Only hp.com" { "IP Hostname" ~ ".*\.hp\.com$" }
```

   b. Pass only hosts that have an odd IP Address in the last octet.

      Answer:

      The filter and AVA to pass only odd numbered IP Addresses:

```
OddIP "Odd IP Addresses" { "IP Address" ~ 192.6.249.[1-9][1-9][13579]$ }
```

Lab Exercises

Preparation

1. Review the default `filters` file, then copy the `filters` file to `filters.orig`. You may want to leverage portions of the file during these exercises. Note: Do NOT remove or clear the

original `filters` file. It contains filters that are in use by NNM.

2. To prepare for class labs, you will create some special purpose filters in this lab. In later labs you will build additional filters, so not all combinattions are covered here.

   Part of successful filter operations depend upon understanding the environment to be managed. Examples throughout this workbook may use example configurations that do not match your actual training environment. With the help of your instructor, be sure you can identify the following items:

   DNS Domain (if applicable): _____

   IP Address Range for classroom systems: _____

   IP Hostnames for the classroom: _____

   Additional classroom devices: _____

   Answer:

   Since each training environment may be quite different, the solutions could lead to some confusion. In addition, there is almost always more than one way to solve a particular filtering problem. To help limit the amount of confusion, the solutions presented for this class are based upon a fictional network with the following characteristics.

   The systems in the classroom have this layout:

Row 1:	r1node1	r1node2	r1node3	r1node4
Row 2:	r2node1	r2node2	r2node3	r2node4
Row 3:	r3node1	r3node2	r3node3	r3node4

   The addresses are: 156.153.206.110 through 156.153.206.121

   printer1 (printer)156.153.206.150

3. Create and test a filter, without using a set definition, that passes only the systems from your classroom. Verify its operation using `ovfiltertest` and view the results in a Node View.

   Answer:

   The filter looks something like this:

   ```
   OurRoom "" { "IP Hostname" ~ "^r[1-3node[1-4]" }
   ```

   Test the filter with the following commands:

   `ovfiltercheck` (to verify syntax)

   `ovfiltertest -f OurRoom` (to test against the object database)

   To see the results graphically, start Home Base and launch Node View. Select your filter from the list and set the severity to Normal.

4. Modify the above filter to use a Sets definition.

   Answer:

   Create a file, `roomlist.txt` containing:

   ```
   r204c21.atl.edunet.hp.com
   r204c22.atl.edunet.hp.com
   r204c23.atl.edunet.hp.com
   r204c24.atl.edunet.hp.com
   ```

```
      r204c25.atl.edunet.hp.com
      r204c26.atl.edunet.hp.com
      r204c28.atl.edunet.hp.com
      r204c29.atl.edunet.hp.com
      r204c30.atl.edunet.hp.com
      r204c31.atl.edunet.hp.com
      r204c32.atl.edunet.hp.com
      r204c33.atl.edunet.hp.com
```

UNIX:

```
Sets {
    ...
      RoomList "" { /etc/opt/OV/share/conf/roomlist.txt }
    ...
   }
Filters {
    ...
      OurRoom "" { "IP Hostname" in RoomList }
   }
```

Windows::

```
Sets {
    ...
      RoomList "" { /Program Files/HP OpenView/conf/roomlist.txt }
    ...
   }
Filters {
    ...
      OurRoom "" { "IP Hostname" in RoomList }
   }
```

As with any change in the filters file definition, verify the syntax with `ovfiltercheck` and test the output with `ovfiltertest`. You can also view it with Node View.

5. (Optional Exercises) Write and verify a filter to do each of the following:

   a. Show just topology equipment.

      Answer:

      ```
      TopologyEquip "" { isRepeater || isHub || isBridge || isRouter }
      ```

   b. Show only PCs (requires sysObject ID).

   c. Show a particular subnet, where the network mask is not at a byte boundary.

      Answer:

      ```
      # Subnet 15.19.80  (netmask 255.255.248.0)
      Subnet15_19_80 "15.19.80 Subnet" { "IP Address" ~ 15.19.80-87.* }
      ```

d. Show HP Series 800 systems usint the `sysDescr` attribute.

Answer:

```
Sys800 "800 Series system" { "SNMP sysDescr" ~ "HP-UX.*9000/8" }
```

6. (Advanced Optional Exercises) Write and verify a filter to perform the following:

a. Show only critical systems.

Answer:

```
# Critical Items
# NOTE:  Since nodes are super-objects, this pulls in all nodes
# which have at least one interface which cannot be pinged, or have
# an IP Status of Critical.  Unfortunately, interfaces like "ni0"
# are always critical, so this filter includes nearly everybody.
CriticalOops "Critical Nodes" { "IP Status" == "Critical" }


# Critical Cards
# Show IP cards which cannot be pinged.  This also causes the node
# super-objects containing them to pass the filter.
# NOTE:  isInterface includes pseudo-cards like "ni0", so we use isCard
# to avoid them.
CriticalCards "Non-Normal Nodes" { isCard && "IP Status" == "Critical" }
```

b. Show only systems whose IP address in the last octet is an EVEN number (trickier, since some objects not of interest have an empty IP address attribute, i.e. is 0, and you don't want those.)

Answer:

```
# Even last octet IP addresses
# NOTE:  This is a little more difficult, since some "pseudo" cards
# have an IP address of 0.0.0.0.  Consequently, we ensure that a
# trailing 0 in the last octet is preceded by a non-zero.
EvenIP "Even last octet in IP address" { "IP Address" ~ "[2468]$" ||

     "IP Address" ~ "[0-9][02468]$" }
```

Chapter G, Device Managment Details

Cisco Board Discovery

Assumptions:

- `$OV_CONTRIB/NNM7labs/initialsetup.ovpl` has been executed.

Directions

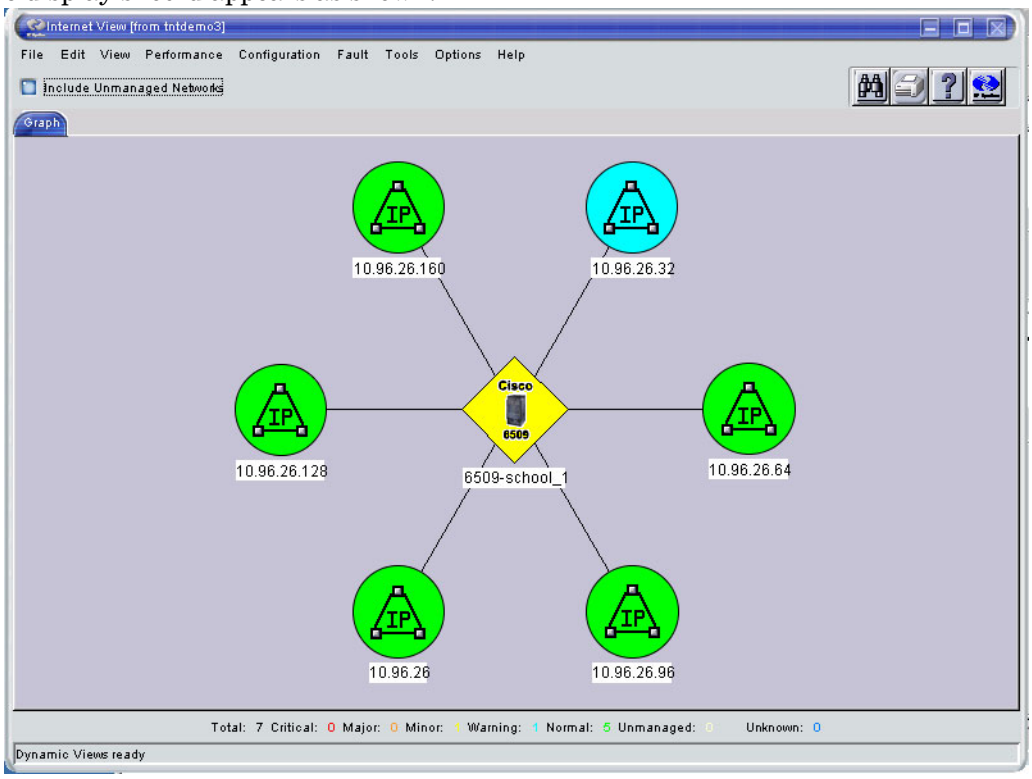
1. Change working directory to `$OV_CONTRIB/NNM7labs/Lab_deploy`:

   ```
   cd $OV_CONTRIB/NNM7labs/Lab_deploy
   ```

2. When the instructor has given you the go-ahead, execute the setup script:

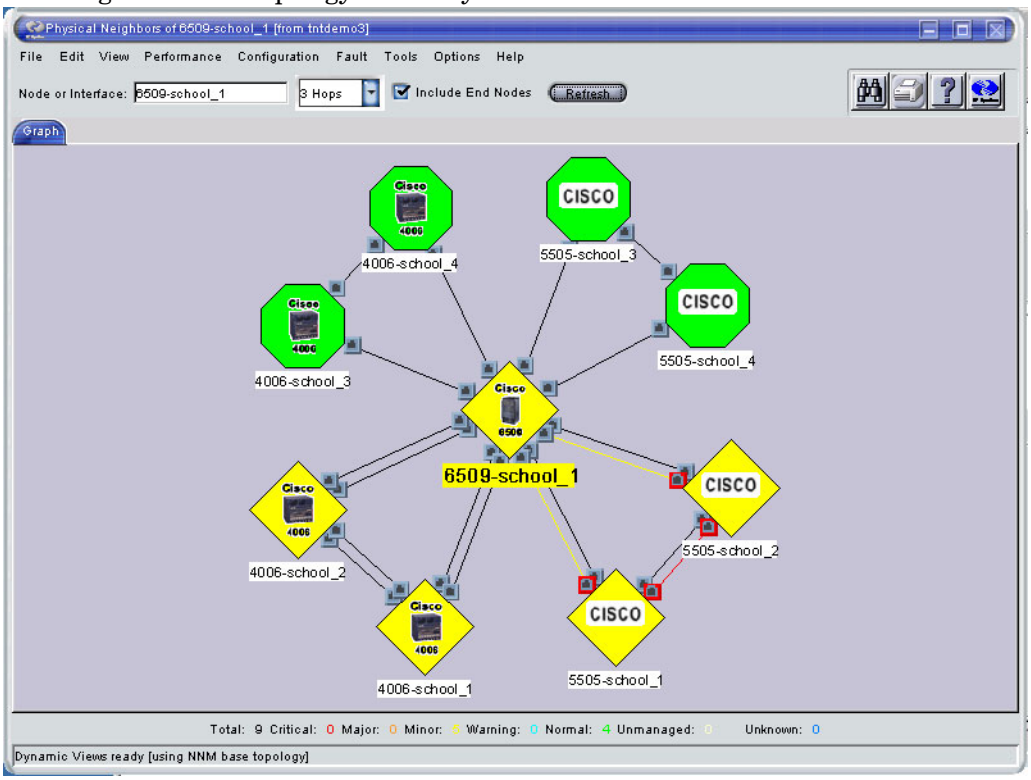
   ```
   setupLab_deploy.ovpl
   ```

3. When this completes, open an Internet View from Home Base.
4. Verify proper and complete discovery of the simulated network. Locate any symbol that may be unmanaged or unknown. If unmanaged, select it, and then use the `ovw` menu `Edit:Manage`, or the dynamic view menu `File:Topology:Manage`. For any node which shows as a blank square, select the node and `Fault:Network Connectivity:Poll Node`.
5. The display should appears as shown.



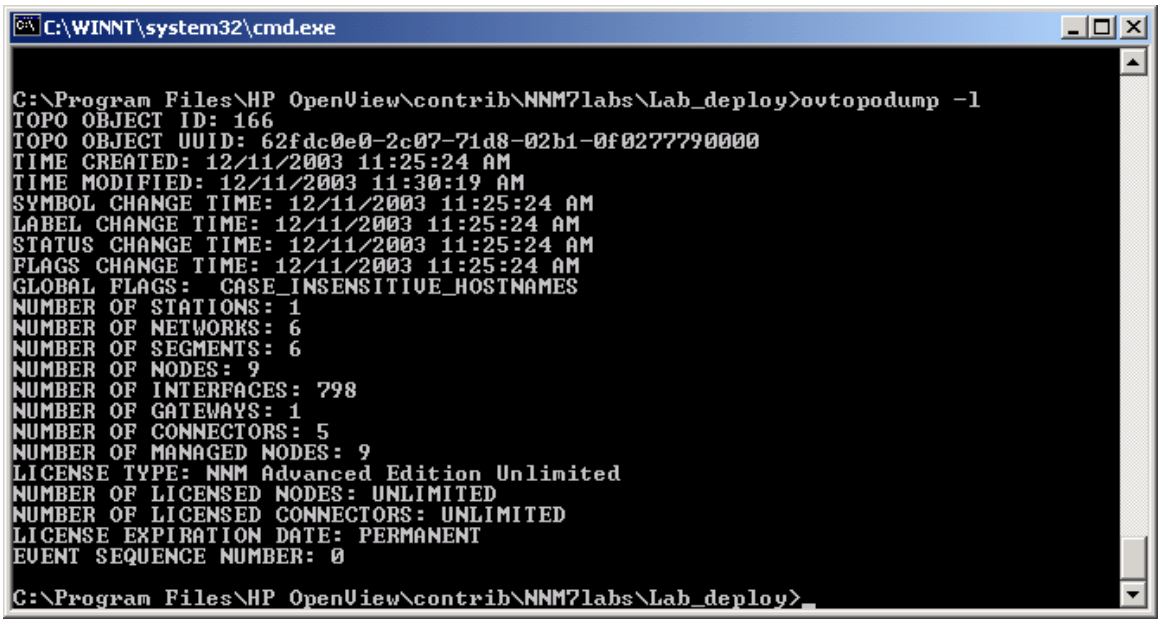
6. Execute a Neighbor view of the device 6509-school_1 with 3 hops and include end nodes, prior

to executing Extended Topology discovery.



7. Confirm correct topology has been discovered:

```
ovtopodump -l
```



8. When you have confirmed that all devices are managed and appear as shown in the above figures, commence the Extended Topology discovery from Home Base.

NOTE You will run several iterations of Extended Topology discovery throughout these labs. Each execution of the discovery will give you an opportunity to use the support tools to view progress of the discovery processes and agents. These

tools will be especially important in the labs that simulate discovery problems.

9. From the NNM Home Base, bring up a Node View. Select “All” for the Show Nodes field and “Normal” for the Status >= field, then hit the Refresh button. Note the status of the nodes and links in the Node View.
10. Right click on the 6509-school_1 node, and select Details.
11. In the Node Details window, click on the Boards link in the top frame.
12. Note the board detail information in this section of the Node Details window. Note the status of the board with interface ID 205.
13. Close the Node Details window.
14. When the instructor gives you the go-ahead, right click on the 6509-school_1 node. Select “APA Status Poll” from the resulting pop-up menu. Wait for the APA status poll to complete in the resulting APA Demand Poll window.
15. Repeat step 9to bring up a Node View of your entire network. Note the new status of the nodes and links in the Node View.
16. Right click on the 6509-school_1 node, and select Details.
17. In the Node Details window, click on the Boards link in the top frame.

Chapter H, IPv6 in Extended Topology

1. What are three reasons for choosing IPv6 addressing?

   Answer:

   Increased available address space.

   Improved router performance.

   Standardized security.

2. What steps are involved in preparing to manage IPv6 networks?

   Answer:

   On HP-UX 11.11, install the IPv6 OS patch.

   Run `setupExtTopo.ovpl` and enable IPv6 when prompted.

   Configure IPv6 configuration files.

3. How does Extended Topology handle routers that run only IPv6?

   Answer:

   If a router does not support IPv4, some connectivity information may not be available. Extended Topology has it only partially managed.

4. What should you place in the IPv6 seed file?

   Answer:

   The IPv6Seed.conf file contains the loopback address of all IPv6 routers that you want discovered.

5. How do you control the frequency of status polling for IPv6 nodes?

   Answer:

   Edit IPv6Polling.conf and enter the system(s) (either hostname or IP address), polling interval, and timeout values.

6. What is the IPv6Prefix.conf file used for?

   Answer:

   This file allows you to display meaningful names for groups of IPv6 nodes rather than long IPv6 address prefixes as lables.

Index

A

accessing
  varbinds, 14-20
Active Problem Analyzer, 1-21, 8-5, 8-6
active tables, A-14
add, 18-13
Advanced Edition, 1-5
Advanced Filter, 13-7, 14-8, 15-5
  usage, 15-9
Advanced Problem Analyzer, 7-10, 8-3
Advanced Routing SPI, 1-7
advantages of using APA, 8-14, 8-15
agent-addr, 14-12
agent-address, 14-12
aggregate port, G-15
  alarm, G-20
  configuration, G-18
  correlation, G-18
  polling, G-18
  visualization, G-20
aggregated port
  discovery, G-17
  status, G-17
alarm
  launching views, A-4
Alarm Browser, 1-26
  web, 1-29
Alarm Definition, 13-6, 14-7
Alarm Signature, 13-6, 14-7, 14-9
AlarmCnt, 16-5
  usage, 16-8
alarms
  browsing, 1-29
all VLANs view, A-11
altering event, 15-33
anycast, G-39
  status, G-40
APA, 8-3
  adjusting polling parameters, 8-22
  advantages, 8-14, 8-15
  alarm reduction, 8-41
  alarms
    defined, 8-41
  changing default polling interval, 8-23
  changing device class polling interval, 8-24
  compared to netmon, 8-43
  configuring, 8-22
  Connector Down correlation, 8-42
  cooperation with netmon, 8-14
  demand poll, 8-19
  disabling
    using ovet_apaConfig.ovpl script, 8-18
  do not enable on management stations, 8-14
  ECS correlations, 8-42
  enable on collection stations, 8-14
  enabling, 8-18
    paConfig.xml file, 8-18
    using ovet_apaConfig.ovpl script, 8-18
  enabling HSRP group polling, 8-28
  event reduction, 8-41
  failure analysis, 8-40
  general IPv4 interfaces, 8-14, 8-18
  HSRP, 10-10
  HSRP group polling
    disabling, 8-29
    enabling, 8-29
  ICMP polling
    disabling, 8-32
    enabling, 8-32
  neighbor analysis, 8-45
  netmon device discovery, 8-14
  no IPX support, 8-15
  node status, 8-16
  ovet_poll process, 8-41, 8-44
    starting and stopping, 8-24, 8-27
  Pair Wise correlation, 8-42
  Polling Engine, 8-23
  polling statistics, 8-57
    Active Analyzer Tasks, 8-57
    Addresses Polled, 8-58
    HSRP Groups Polled, 8-58
    Interfaces Polled, 8-58
    Waiting Analyzer Tasks, 8-58
    Waiting Poller Tasks, 8-57
  SNMP polling
    disabling, 8-31
    enabling, 8-31
  Status Analyzer, 8-23
  Status Bridge, 8-23
  Talker, 8-23
  unconnected interface, 8-47
  unconnected switch ports
    disable SNMP polling, G-4
    enable SNMP polling, G-4
  using topology filters, 8-25
APA adjusting polling parameters
  paConfig.xml file, 8-22
APA Configuration Poller, 8-51
APA disabling HSRP group polling, 8-28
APA important nodes filter, 8-49
APA unmonitored, 8-36
applet
  trusted, 1-31
architecture
  OAD, 7-43
area border router, 9-2, 9-3
area,OSPF, 9-2

Index

AS boundary router, 9-3
Asynchronous, 18-5
asynchronously, 18-3
ATM, G-30
Attribute Value Assertions (AVA), E-12
attributes, 18-7, 18-10
automatic variables, 16-5
automatic zone partitioning, 3-7, 3-8
Autonomous Border Rotuers, 9-3
autonomous system, 9-2

B

backbone router, 9-3
backup IP address, G-41
backup link, G-41
bitand, 18-13
bitinv, 18-13
bitor, 18-13
bitxor, 18-13
board, G-9
  correlation, G-12
  discovery, G-10
  events, G-12
  status, G-11
  visualization, G-13
Border Gateway Protocol, 9-2
bridge.noDiscover, 6-4
bridge.noDiscover file, 6-2
brownout, D-10
browsing
  alarms, 1-29
Built-in function, 18-2, 18-3
Built-in functions, 18-13

C

C function, 18-2
C functions
  caution, 13-38
C library
  specify, 18-30
Callback Function, 18-7
  usage, 18-9
Callback Section, 13-7
card
  Cisco, G-9
  events, G-12
Chassis correlation, 12-12
child events, 12-33
Cisco, 12-12
Clear Alarm
  usage, 16-19
Combine
  usage, 16-18
combine, 15-2, 15-27
  usage, 15-31
combining correlators, 20-2
communication address, 7-15
Composer, 13-2
  architecture, 11-19, 11-20
  backups, 19-5
  cautions, 13-38
  configure deployment, 13-33
  customer requirements, 13-8
  datastore, 15-11
  deploy, 13-35
  developer mode, 13-12
  event type, 13-14
  file versions, 19-5
  files, 13-18
  lock file, 13-22
  migration, 19-4
  operator mode, 13-10
  performance, 13-37
  references, 13-39
  security, 13-29
  templates, 13-16
Composer.ds, 15-11
configuration polling
  configuring, 8-53
configure
  Composer deployment, 13-33
  operator access, 13-24
configuring
  aggregate port, G-18
  configuration polling, 8-53
  connector down, 12-28
  ECS, 12-3, 12-46
  filters, E-5
  IPv6, H-30
  NameSpace, 13-27
  problem diagnosis probe, D-15
  recurring discovery, 3-15
  secondary failures, 12-28
  syslog, 21-15
  zones, 6-7
connector down, 12-14, 12-26
  configuring, 12-28
ConnectorDown
  with ET, 12-32
constant, 15-2
continuous discovery, 2-2
correlation
  aggregate port, G-18
  board events, G-12
CorrelationDuration, 16-5
  usage, 16-8
correlations, 11-8, 11-9
  connector down, 12-26

descriptions, 12-5, 12-14
enabling, 12-7
pairwise, 12-33
repeated events, 12-38
scheduled maintenance, 12-42
correlator, 11-22
combining, 19-2, 20-2
creating, 14-5
deploy, 13-35
description, 14-6
evaluation order, 11-20
HSRP, 10-12, 10-15
interaction, 13-39
interference, 19-2
Correlator Creation, 18-3, 18-4
Correlator Deletion, 18-3, 18-4
correlator name, 14-5
Correlator Store, 13-18
correlator store
merge, 19-7
Correlator Templates, 13-2
create
correlator, 14-5
new event, 15-28
create event, 16-2, 16-9, 16-15, 17-3
usage, 17-16
creating event, 15-34
csmerge, 19-3, 19-7
current path, C-12, C-14

D

datastore, 15-2, 15-11
creating, 15-11
DCE, 21-12
debugging, 19-20
dedup.conf, 12-40
de-duplication, 1-23, 11-21, 11-22, 12-40, 13-2
Default
evaluation, 18-4
Default function evaluation, 18-3
demand poll
APA, 8-19
deploy
Composer configuration, 13-33
correlators, 13-35
OAD, 7-30
syslog, 21-36
developer mode, 13-12
devices
ET, 2-6, 2-8
DHCP, E-5
disable
problem diagnosis probe, D-20
syslog, 21-43
disabled interface, 8-36
disabling
internal correlation, 12-10
Discard Duplicate, 16-10
usage, 16-13
discard event, 16-2, 16-15, 17-3
discard events, 14-2, 16-9
Discard Immediately, 16-10
usage, 16-13
discovery
aggregated port, G-17
ET, 1-9, 2-2
incremental, 3-13
IPv6, H-29
limiting, 6-2
limiting IPv6, H-33
recurring, 2-3
zones, 3-5, 6-5
discovery status, 1-26
display
label, A-13
distributed ET views, 4-4, 4-5
div, 18-13
DNS
restricting lookups, 5-14, 5-16
does NOT match, 14-11
dump
events, 19-10
dupip.conf, 7-21, 7-23
duplicate access points, G-43
duplicate IP, G-37
duplicate IP address, G-44
duplicate VLAN ID, A-11
Dynamic NAT, 7-7
dynamic parameters, 12-16
dynamic view
menu, 1-31
print, A-16
Dynamic Views, 1-25
security, B-2
dynamic views, 1-33, A-5
all VLANs, A-11
from alarm, A-4
index, A-3
internet, A-8
lines, A-5
menu, A-5
network, A-9
OSPF, 9-6
segment, A-10
starting, A-2
status, A-5
dynamicViewsUsers.xml, B-8

Index

E

ECS, 11-6
  architecture, 11-10, 11-12
  chassis correlation, 12-12
  configuring, 12-3, 12-46
  correlations, 12-14
  data types, 12-21
  dynamic parameters, 12-16
  integration, 11-6
  IntermittentStatus, 12-11
  internal correlations, 12-10
  NodeIF, 12-10
  static parameters, 12-16
  troubleshooting, 12-48
ECS architecture, 11-19
ECS Correlation, 11-22
ECS correlation, 1-23, 11-21
ECS Designer, 11-7, 11-14
ECS Event Configuration GUI, 12-4
ECS parameters
  modifying, 12-19
ecsevgen, 19-13, 19-16
ecsmgr, 12-46, 19-13, 19-18
enable
  Extended Topology, 3-4
  syslog, 21-35
enabling
  correlations, 12-7
  MgXServerDown, 12-14
enabling ET discovery, 3-4
endpoints, C-7
Enhance, 11-20
  usage, 15-31
Enhance template, 15-28
ET
  ConnectorDown, 12-32
  device support, 2-6, 2-8
  discovery, 1-9, 2-2
  enabling discovery, 3-4
  replication, 4-3
  starting discovery, 3-4
  stopping, 3-12
  supported devices, 2-6, 2-8
  views, 9-6, A-11
  views in DIM, 4-4, 4-5
etrestart.ovpl
  manpage, 3-3
  script, 3-3
evaluation of correlators, 11-20
event
  analyze, 19-11, 19-12
  analyzing, 19-10
  capture and playback, 19-13
  create, 15-34, 17-16
  dump, 19-10
  logging, 19-13
  modify, 15-33
  replay, 19-16
  tracing, 19-18
event attributes
  customizing, 14-9
Event Correlation Service, see ECS, 11-6
event correlations, 11-3
  modifying, 12-16
event de-duplication, 12-40
event flows, 11-10
Event In, 18-3, 18-4
  usage, 18-22
event pattern deletion, 12-36
event reduction, 1-23, 11-2, 11-21
event storm, 13-39
  simulating, 19-3
event storms, 1-17, 12-26
event streams, 11-10, 12-4
events
  holding, 13-39
  HSRP, 10-16
Expand Neighbors, A-15
Extended Topology, 1-3, 1-9
  configuring recurring discovery, 3-15
  discovery
    limiting, 6-2
    recurring, 3-15
    status of, 3-17
    zones, 3-5
  Discovery Exclusion List, 6-2
  discovery zones, 6-5
  enable, 3-4
  introducing, 1-9
  NNM and Extended Topology, 1-9
  Oracle, 6-29
  processes, 3-18
  requirements, 6-5
  using SNMP community strings, 3-3
Extract
  usage, 17-8
extract, 15-3, 15-15, 15-22, 21-18
  assignment, 15-19, 21-22
  usage, 15-25
Extreme switches, 2-8

F

file
  bridge.noDiscover, 6-2
filters, E-3
  and expressions, E-17
  attributes, E-7

Index

AVAs, E-12
configuring, E-5
important nodes, 12-29
operators, E-15
overview, E-4
pattern matching, E-19
sets, E-10
syntax, E-8, E-17, E-26
testing, E-26
wildcards, E-19
firewall, 7-17
flapping
board, G-12
flows of events, 11-10
frame relay, 1-7
FrameRelay, G-30
Function, 18-2
built-in, 18-13
Correlator Creation, 18-3
Correlator Deletion, 18-3
debug, 19-20
Default evaluation, 18-3
evaluation, 18-5
Event In, 18-3
parameters, 18-3
support, 18-2
usage, 18-9, 18-22
function
variable
function, 15-3
Function keys, 18-17

G

genannosrv, 18-3
general IPv4 interfaces, 8-14
polling, 8-16
getByIndex, 18-13
getCounter, 18-13
getHour, 18-13
getMinute, 18-13
getMonth, 18-13
getTime, 18-13
Global Constant, 14-7
usage, 18-22
global constant
variable
global constant, 15-4
grant access to applet, 1-31
graph
print, A-16

H

highlight VLAN, A-5
Home Base, 1-25
checking discovery status, 3-17
hosts.nnm, 2-12
hosts.nnm file, 8-15, 8-18
HP ProCurve stack, G-28
HSRP, 1-3, 3-5, 8-6, 10-2
correlator, 10-15
disable, 10-12
events, 10-16
example, 10-4
group status, 10-3
polling, 10-10
reference, 10-2
state, 10-3, 10-11
state transition, 10-3
view, 10-13
with OAD, 7-40
HSRP group polling
disabling, 8-28
enabling, 8-28
HSRP polling
disabling, 8-29
enabling, 8-29
HSRP status
not in an OAD, 8-7
HSRP view, 7-40

I

I/O Board, G-9
ICMP polling
disabling, 8-32
enabling, 8-32
important nodes
filters, 12-29
important nodes filter
APA, 8-49
incremental zone discovery, 3-13
index of dynamic views, A-3
installing
verifying, F-12
instance, 16-3
Intelligent Diagnostics, 1-21
Interface, G-9
interface
unmanage, 5-5
interface down, 12-10
interface renumbering, 8-54
IntermittentStatus, 12-11
internal correlation, 12-10
chassis, 12-12
disabling, 12-10
IntermittentStatus, 12-11
internal router, 9-3
internet submaps, A-6

internet view, 1-34, A-8
IP address
  private, 7-3
IP Telephony, 1-8
ipNoLookup.conf, 5-16
IPv6, 3-5, G-39
  addresses, H-4
  background, H-2
  configuring, H-30
  discovery, H-29
  extending discovery, H-33
  limiting discovery, H-33
  logfile, H-40
  prefix, H-8, H-39
  requirements, H-26
  scoping, H-7
  status calculation, H-18
  status polling configuration, H-37
  troubleshooting, H-34
  view, H-16, H-20
IPv6.conf, H-33
IPv6Polling.conf, H-37
IPv6Seed.conf, H-31
IPX
  submaps, A-7
is in list, 14-12, 14-20
is NOT in list, 14-12

J

Java versions
  checking, 1-31

K

keys, 18-17

L

label
  changing display, A-13
  port, A-13
LACP, G-15
LAN/WAN Edge SPI, 1-7
layer 2
  ET discovery, 1-9
layer 3 connection
  configuration, G-35
layer-3 connection, G-30
library name, 18-2
lock, 13-22
Log Only events, 1-23, 11-21
logfile
  IPv6, H-40
logfile encapsulator, 21-7
logger
  syslog
    testing, 21-37
logging, 6-30
LOGONLY, 11-21
lookup
  usage, 15-13, 15-32
  variable
    lookup, 15-2

M

makeList, 18-14
ManageXServer Down Correlation, 12-14
manpage
  etrestart.ovpl, 3-3
  xnmsnmpconf, 3-3
matches, 14-11
MD5, B-10
memory realm, B-9
merge
  correlator store, 19-7
merging Composer files, 19-3
Message Key, 13-7, 14-8, 16-3
  usage, 16-7, 16-13, 16-19, 17-6, 17-9
MgXServerDown Correlation, 12-14
MgXServerDown correlation
  enabling, 12-14
mod, 18-14
modify event, 17-3
  usage, 17-15
modifying
  ECS parameters, 12-19
  event correlations, 12-16
module, G-9
MPLS IP VPN SPI, 1-7
mul, 18-14
Multicast, 1-7
multicast, G-39
MultiLink Trunk (MLT), G-24
multiple reboots
  correlator
    multiple reboots, 12-13
Multi-Source, 16-3
  sets, 17-3
  usage, 17-6, 17-8, 17-12
Multi-Source template, 17-2

N

name
  correlator, 14-5
NameSpace, 19-8
  configuring, 13-27
NAT, 7-4
  support, 1-13, 7-9
  types, 7-7

Index

neighbor analysis, 8-45
Neighbor view, 1-26
neighbor view, 1-33, 4-4
netmon
  device discovery for APA, 8-14
  duplicate IP, G-37
netmon polling, 12-23
Network Address Translation, 7-4
network view, 1-34, A-9
New Alarm Section, 13-7
NNM
  Extended Topology and NNM, 1-9
NNM Extended Topology, 1-9
NNM Smart Plug-in
  for Frame Relay, 8-14
  for MPLS IP VPN, 8-14
node down, 12-10
Node view, 1-26
node view, 1-34, 4-4
NodeIF correlation, 12-10
nodes
  attributes, E-6

O

OAD, 7-18
  HSRP group, 7-40
  ID, 7-18
  VLANs, 7-39
object
  unmanage, 5-3
object IDs, 2-13
opcle, 21-8
opcmsga, 21-8
opcpat, 21-25, 21-37
opctla, 21-8
operator access
  configure, 13-24
operator mode, 13-10
operators, 14-11
Oracle, 6-29
OSPF, 1-3, 2-12, 9-2
  area, 9-2
  backbone, 9-3
OSPF view, 4-5, 9-6
OV_PollerPlus correlators
  contributed software, 8-41
OV_Syslog_FrameDLCI_Active, 21-10
OV_Syslog_FrameDLCI_Inactive, 21-10
OV_Syslog_LineProtoDown, 21-10
OV_Syslog_LineProtoUp, 21-10
OV_Syslog_LinkDown, 21-10
OV_Syslog_LinkUp, 21-10
OV_Syslog_OSPF_Neighbor_Down, 21-10
OV_Syslog_OSPF_Neighbor_Up, 21-10
ovautoifmgr, 5-5, 5-8
ovcomposer, 13-11, 13-12
ovdumpevents, 19-10
ovdupip, 7-27
Overlapping Address Domain, 7-10, 7-14, 7-18
  architecture, 7-43
  configuration, 7-19
  deploy, 7-30
  polling, 8-12
  seed file, 7-25
  understanding status, 7-10
  view, 7-34
Overlapping Address Domains
  support, 1-13, 7-9
  verify, 7-27
ovet_apaConfig.ovpl script, 8-18
ovet_bridge, 2-12
ovet_demandpoll.ovpl, 8-19
ovet_disco, 2-12
ovet_model, 2-12
ovet_poll, 8-3
ovet_poll process, 8-41, 8-44
  starting and stopping, 8-24, 8-27
ovet_topodump.ovpl, 6-20, G-22
ovet_topodump.ovpl script, 8-25, 8-26
ovfiltercheck, E-26
ovfiltertest, E-27
OVO, 21-4, 21-6
OVO agent, 21-5
ovsyslogcfg, 21-14, 21-15
ovtopodump, E-27

P

paConfig.xml file, 8-18, 8-22, 8-25
PAGP, G-15
PairWise, 12-11, G-12, G-19
pairwise, 12-33
Pairwise Correlation, 12-14
Parameters, 13-7, 14-8
parameters, 18-7
parameters for functions, 18-3
parent event, 12-33
partial path, C-11
password, B-2
patches
  reviewing, 1-31
path detail, C-12
Path Engine, 2-13
path list, C-8
path map, C-10
path view, 1-33, 4-4
pattern
  testing, 21-25, 21-37
pattern delete, 12-36
pattern-matching, 15-16, 21-19
pdconfig.xml, D-6

PDU
  sample, 14-4
performance, 19-3
periodic discovery, 2-2
Perl file
  specify, 18-30
Perl function, 18-2
perl scripts
  caution, 13-38
persistence, E-5
pie chart, 1-26
pmd
  crashing, 13-38
  tracing, 19-18
polling
  OAD, 8-12
polling order, 12-10
Port, G-9
Port Address Translation, 7-7
port aggregation, G-15
port labels, A-13
poster print, A-16
primary filter, 13-6, 14-9
print
  dynamic view, A-16
private address, 7-14
private IP addresses, 7-3
probe
  configure server, D-22
problem diagnosis
  add probe, D-5
  disable probe, D-20
  ports, D-8
  probe, C-5
  probe configuration, D-15, D-22
  probe installation, D-13
  server, C-4
  server installation, D-3
  start server, D-4
  starting probe, D-18
  stop server, D-4
  stopping probe, D-18
  uninstall, D-27
  view, C-6
processCorrEvents, 19-12
processEvents, 19-11
public address, 7-14

R

Rate
  usage, 16-7
Rate template, 16-2
rd0.nnm, 2-12
recurring discovery, 2-3
redundant link, G-41
reference
  Composer, 13-39
regular expression, 15-17, 21-20, E-21
remove
  syslog, 21-43
Repeated, 11-20, 16-9
  usage, 16-13
repeated events, 12-14, 12-38
Repeated template, 16-9
replication
  ET, 4-3
requirements
  Composer, 13-8
  IPv6, H-26
retrieve, 18-14, 18-17
retrieveStr, 18-17
  syntax, 18-19
  usage, 18-22
retrievstr, 18-14
role, B-4, B-6
Routable Overlapping NAT, 7-7

S

sample trap, 14-4
scheduled maintenance, 12-14, 12-42
scope of variable, 15-4
script
  etrestart.ovpl, 3-3
secondary events, 11-3
secondary failures, 12-26
  configuring, 12-28
secondary filter, 13-7
security
  dynamic views, B-2
  tomcat, B-2
security file
  Composer, 13-29
security-constraint, B-6
seed file
  OAD, 7-25
segment view, 1-34, A-10
set, 17-3
  usage, 17-13
setCounter, 18-14
setupExtTopo.ovpl, 3-5, H-32
setupSyslog.ovpl, 21-14
signed applet, 1-31
Smart Plug In, 1-7
SNMP
  unresponsive, 2-4
SNMP community strings
  used by Extended Topology, 3-3
SNMP polling
  disabling, 8-31

enabling, 8-31
snmpnolookupconf, 5-14
SPIs, 1-7
start
view from alarm, A-4
Starter Edition, 1-6
starting
ET discovery, 3-4
starting views, A-2
Static NAT, 7-7
static parameters, 12-16
station view, 1-34
status
aggregated port, G-17
configuring IPv6 polling, H-37
dynamic views, A-5
IPv6, H-18
Status Analyzer, 8-10
status engine, 8-10
stop
ET processes, 3-12
store, 18-14, 18-17
storeStr, 18-14, 18-17
usage, 18-22
storestr
syntax, 18-18
sub, 18-14
submaps
internet, A-6
node, A-7
segment, A-7
subnet mask, G-30
support
function calls, 18-2
support matrix
ET, 2-6
Suppress, 11-20
usage, 14-14, 15-25
Suppress template, 14-2
suppression, 11-9
symbols
segment, A-6
synchronous functions, 13-38
synchronously, 18-3, 18-5
syslog, 21-3
architecture, 21-7
conditions, 21-9
configuration, 21-15
deploy, 21-36
disable, 21-43
enable, 21-35
messages, 21-9
remove, 21-43
template, 21-9
troubleshooting, 21-40
verify, 21-25, 21-37
syslogTrap, 21-7

T

table
tab, A-14
tables, A-14
templates, 13-16
Enahnce, 15-28
Multi-Source, 17-2
Rate, 16-2
Repeated, 16-9
Suppress, 14-2
Transient, 16-15
test, 19-16
test system, 19-2
Test Zone Configuration, 6-10
testing, 19-2, 19-18, 19-20
Threshold
usage, 16-19
threshold
Rate correlator, 16-2
Transient correlator, 16-15
tomcat
security, B-2
tracing, 19-18
turn off, 19-19
Transient
usage, 16-18
Transient template, 16-15
trap
sample, 14-4
trek detail, C-16
troubleshooting, 19-18, 19-20
ECS, 12-48
IPv6, H-34
syslog, 21-40
trunk, G-15

U

unconnected switch ports
disable SNMP polling, G-4
enable SNMP polling, G-4
uninstalling, F-16
problem diagnosis, D-27
unique address, 7-15
unmanage
interface, 5-5
unmonitored, 8-36
unresponsive nodes, 2-4
unsupported templates, 13-17
user
dynamic views, B-8
User Defined template unsupported, 13-17

V

varbind, 18-10
  accessing, 14-20
variable
  automatic, 16-5
  combine, 15-2
    combine
      creating, 15-27
  constant, 15-2
  creating, 15-8
  evaluation, 18-4
  extract, 15-3, 15-15, 15-22, 21-18
  function, 18-2
  scoping, 15-4
variable binding, 14-20, 18-10
Variables, 13-6, 14-7
variables, 15-2
verify
  syslog, 21-25, 21-37
  zones, 3-9
verifying
  filters, E-26
  installation, F-12
versions of Composer files, 19-5
view
  from alarm, A-4
  OAD, 7-34
viewing
  alarms, 1-29
views
  all VLANs, A-11
  dynamic, 1-33, A-5
  ET in DIM, 4-4, 4-5
  index, A-3
  internet, A-8
  lines, A-5
  menu, A-5
  network, A-9
  OSPF, 9-6
  segment, A-10
  starting, A-2
  status, A-5
visualization
  aggregate port, G-20
  board, G-13
VLAN
  in OAD, 7-39
VLAN ID
  duplicate, A-11
VLAN view, 4-5

W

warnings about Composer, 13-38
web
  Alarm Browser, 1-29
web.xml, B-6

X

xnmpolling, 12-29
xnmsnmpconf
  manpage, 3-3

Z

zone
  automatic partitioning, 3-7
  incremental discovery, 3-13
zone discovery, 6-5
zones
  configure, 6-7
  verify, 3-9