HP OpenView

OpenView Forum

The Rocky Road to Advanced Status in NNM

Session #: 282

Speaker: Mike Peckar

Company: Fognet Consulting

Agenda

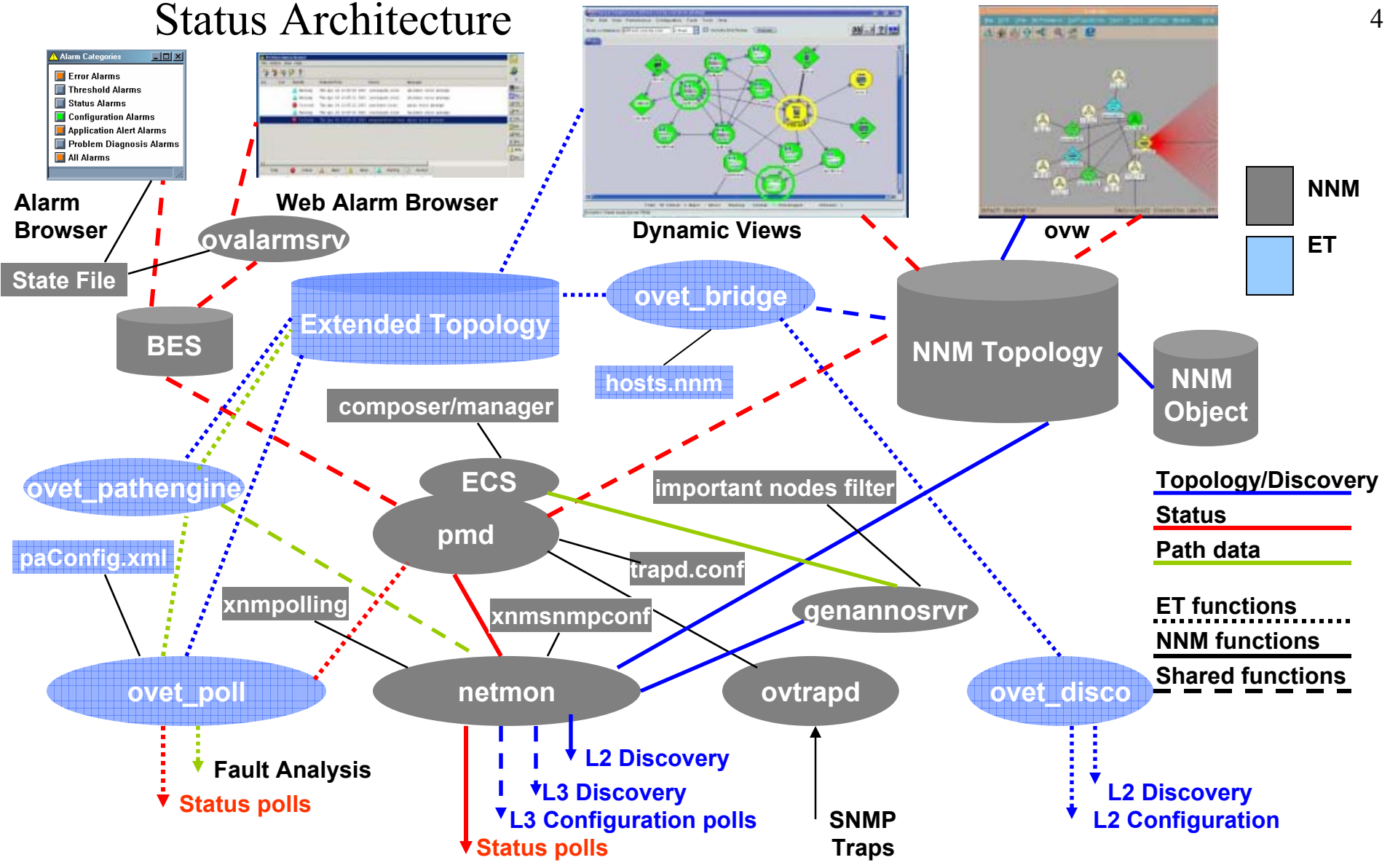
Welcome!

Important Figures

NNM status subsystems in a nutshell

- *netmon* performs discovery and status polling (old poller)
  - 4 areas of polling: Discovery, Configuration, Status, Path
  - *netmon* continues to be responsible for Discovery/Config after switch to APA
  - ICMP only, or in limited cases, SNMP-only.
  - Multiple configuration entry points; limited dynamic reconfiguration
- *Active Problem Analyzer* in 7.01+; *ovet_poll* (new poller)
  - Used for HSRP & OAD by default for ET-discovered devices
  - Issues ICMP *and* SNMP polls; layer two-based status, path analysis
  - Address-aware; Intelligent algorithms for status based on ET knowledge
- *Event Correlation* (ECS embedded runtime; correlation composer)
  - Provides state-based logic that is applied to both poller's status streams
  - Some overlap between APA fault analysis and ECS correlation
  - APA fault analysis is performed before events are generated
  - ECS acts on events after they have been sent into the NNM event subsystem

Status Architecture



netmon vs. APA status polling pros & cons

- *netmon* Cons
  - Single-threaded, single protocol poller, with many legacy issues (IP, DNS)
  - Polls via ICMP *or* SNMP, but never both, SNMP poll limited.
  - Secondary status determination mechanisms complex, layered, and inadequate
  - Rudimentarily dynamic w.r.t intervals and polled object relations
  - Cannot poll into OAD's, cannot handle HSRP, NAT, etc.
  - Cannot separate the concept of an IP address from that of a physical interface
  - All status derived from interface-level via topology propagation rules
  - Is not "connection aware," i.e. can't relate failures detectable from other paths
  - Rudimentary path analysis: Unique path to each interface to determine primary
  - Disparate and legacy configuration switches, files, and dependencies

- *netmon* Pros
  - Behavior characterizable and configurable, less FUD for installed base
  - Due to its lack of in-depth analysis capabilities, still scales OK
  - GUI's available for polling customization and configuration
  - Preservation of investment in netmon-based status

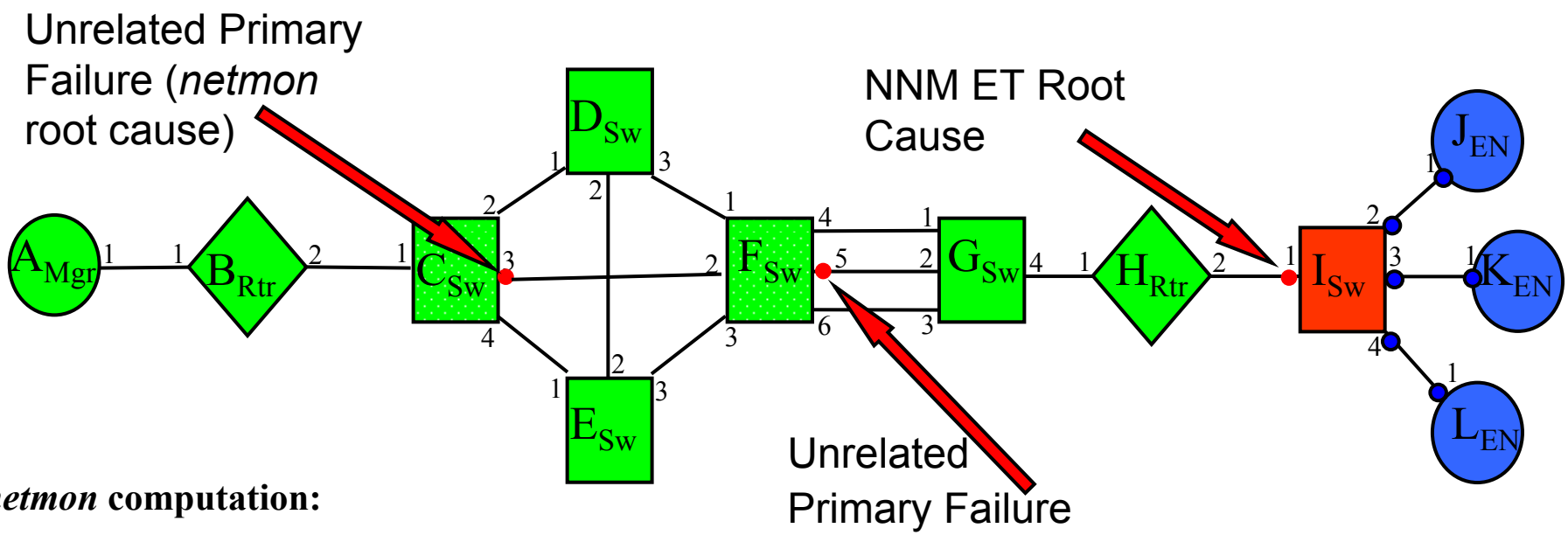
netmon vs. APA status polling pros & cons

- APA Cons
  - Analysis engine is complex and difficult (impossible??) to interpret behaviors
  - Does not do discovery – relies on *netmon* for discovery
  - Polling customization requires modifying XML file; no configuration GUI
  - Many *netmon*-based customizations not inherited by APA, e.g:
    - Poller settings in SNMP Configuration GUI ignored
    - Interfaces unmanaged in NNM topology or by ovautoifmgr still may be polled
    - Object based polling settings based on filters (APA uses ET filters)
    - Separate definition files/filters for Important Nodes (MyHostID.xml)
  - ovet_demandpoll.ovpl, ovet_toposet, available in NNM 7.5+ but not 7.01
  - Cannot be used on management stations in DIDM environments (CS only)
  - Incompatibilities with Lan/Wan Edge SPI 2.0 & MPLS SPI 1.0 (7.01 issue only)
  - IPX and Service Guard polling not available after switching to APA
  - Initial setup can cause message floods in highly-scaled environments
  - Initial configuration steps must be done in proper order with proper timing
  - Status bridge not perfect: IPMAP topology status and APA status mismatches
  - Difficulties in dynamic environments: relies on both ET and *netmon* discovery

netmon vs. APA status polling pros & cons

- APA Pros
  - Multi-threaded, multi-protocol (combines ICMP & SNMP, other protocols)
  - Switched-topology-aware, duplicate IP-aware, neighbor state-aware
  - Event-triggered polling based on NNM and device-generated events
  - Provides status at six different entity levels:
    - Address
    - Interface
    - Node
    - Connection
    - AggPort
    - Board
  - Provides more dynamic polling based on queued status
  - Grouped SNMP queries
  - Special support for OAD, HSRP, IPv6, RAMS
  - Provides advanced connection-oriented and device-oriented status analysis
  - Provides neighbor analysis algorithm
  - Generally provide more accurate & timely status than *netmon*
  - Less reliant on complexities of ECS – more corellation at the source
  - Generates fewer log-only and embedded status events by default
  - Direct migration paths from NNM 6.2, 6.41, 7.01 to 7.5

netmon vs. ET/APA Path Analysis



***netmon* computation:**

A.1 B.1 B.2 C.1 **C.3 F.2** **F.5 G.2** G.4 H.1 H.2 I.1 I.2 J.1

Netmon-based events: Primary: C3; Secondary to C3: F5,I1,J1,K1,L1

ET path engine computation (APA) using connector fault analysis (CFA):

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 D.3 F.1 E.3 F.3**)**

AGGR(F.4 G.1 **F.5 G.2** F.6 G.3**)** - G.4 H.1 H.2 I.1 I.2 J.1

APA-based events: Primary: C3, F5, I1; Secondary to I1: J1, K1, L1

netmon default status polling intervals

- Global default defined in SNMP configuration: 15 minutes
- Dynamically-adjusting polling by *netmon*
  - V6.0 *netmon* enhancement to support ConnectorDown
    - double intervals for polls issued to secondary failure-mode If's Status
  - V6.31 *netmon* enhancement to support 3 new correlations
    - Connector interfaces immediately polled when one's status changes
    - All interfaces changing status re-polled at 2 and 4 minutes
- Object-based polling (V6.2+)
  - Allows different polling intervals for primary vs secondary interfaces
  - Objects defined via NNM standard filters and filter definition language
  - Configure via "Poll Objects" - front-end to configure *netmon.statusIntervals*
  - Tightens default polling intervals for Routers, Bridges, Hubs
  - ***Loosens*** default polling intervals for Nodes to 1 hour (V6.4+)
  - Uses *netmon's* critical path analysis to determine primary interface
- All overridden when using APA polling

netmon object-based polling defaults

- NNM V6.2:
- NNM V6.31:

Object Class	Status Polling Interval (seconds)	Primary Status Polling Interval (seconds)
Routers	180 3 Min	60 1 Min
Bridges	300 5 Min	90 1.5 Min
Hubs	450 7.5 Min	450 7.5 Min

- NNM V6.4:
- NNM V7.01:
- NNM V7.5:

Object Class	Status Polling Interval (seconds)	Primary Status Polling Interval (seconds)
Routers	900 15 Min	60 1 Min
Bridges	14400 4 hours	90 1.5 Min
Hubs	14400 4 hours	450 7.5 Min
Nodes	14400 4 hours	3600 1 Hour

- Use `xnmsnmpconf -resolve target` to determine *netmon*-based intervals
- Use `nmdemandpoll -i target` to force issue status polls (*netmon* status only)

netmon layer 2 status polling defaults

- Support for Bridge, MAU, Repeater MIB; VLANs
- Un-numbered ifs inferred from port table, polled via ARP
  - V5-V6.1: Critical/Normal; V6.2+: Unknown/Normal; V7.0+: Off
- SNMP status mapping fixed from V5 until V6.2
  - Status reflected in maps only; alarms are log-only

ifAdminStatus	ifOperStatus	OV Status	Map Color
down	any	DISABLED	Dark Brown
testing	any	TESTING	Tan
up	up	NORMAL	Green
up	down	CRITICAL	Red
up	testing	TESTING	Tan

– Note APA based SNMP status mapping unexposed

netmon layer 2 status polling

- NNM 6.2+: *netmon.statusMapping* defines customizable SNMP status levels

ifAdminStatus	ifOperStatus	OV Status
up	up	unset
down	down	unknown
testing	testing	normal, up
any	unknown	critical, down
	dormant	disabled
	notpresent	unmanaged
	lowerlayerdown	restricted
	any	testing

- $OV_CONF/*netmon.snmpStatus* - Define L3 IP ranges to poll via SNMP
  - Intended for firewalls: ICMP polling disabled for these devices
  - netmon.lrf –k snmpTimeoutImplies=status[unknown, unchanged, critical (default)

New *netmon* status event varbinds

- Supports NodeIf, ConnectorDown event correlations

IF Status Varbind #	Node Status Varbind #	Description
$2	$2	Hostname of node that caused the event
$5	$5	Timestamp event occured
$7		Interface Name or Label
$8		IP Address of Interface or "0"
$11		Number of bits in the interface subnet mask
✡ $12		Interface ifAlias
✡ $13	$8	Local list of capabilities
✡ $14	$9	Name of primary failure host
✡ $15	$10	Name of primary failure entity
✡ $16	$11	OV OID of primary failure entity
✡ $17	$12	Description of primary failure entity
✡ $18	$13	Primary failure entity list of capabilities

- Event text NNM V6.31+: `IF $7 Down $12, Capabilities: $13 Root Cause $14 $15`
- Event text NNM V6.2-: `IF $7 Down`

✡ = New in NNM 6.31+

Taking the leap

- Read $OV_DOC/whitepapers/Active_Problem_Analyzer.pdf
- Exit GUI sessions
- Run `setupExtTopo.ovpl`, then `etrestart.ovpl`. Wait a while
- Run ovet_apaConfig.ovpl
  - `ovet_apaConfig.ovpl -enable APAPolling`
  - `ovet_apaConfig.ovpl -disable APAPolling`
- What does this script do?
  - Runs xnmpolling with options to switch polling control between netmon/ovet_poll
    - `xnmpolling -statPollOff -ovetPollingOn`
    - `xnmpolling -ovetPollingOff -statPollOn`
  - Makes changes to the $OV_CONF/nnmet/paConfig.xml APA configuration file
  - Populates $OV_DB/nnmet/hosts.nnm from *netmon*-discovered topology
    - *ovet_bridge* uses this file to designate what hosts are polled by APA
  - Restarts appropriate background processes
- Watch ovstatus carefully
  - ovstatus –v *netmon* Should say “Polling 0 interfaces”
  - ovstatus –v *ovet_poll* Should say “Polling devices”

Determining poller control

V7.01
APA off (UNIX):

```
# $OV_BIN/ovet_apaConfig.ovpl -query APAPolling
PollingEngine PollNormalIP Bool false
StatusBridge StatusBridgeEnabled Bool false
```

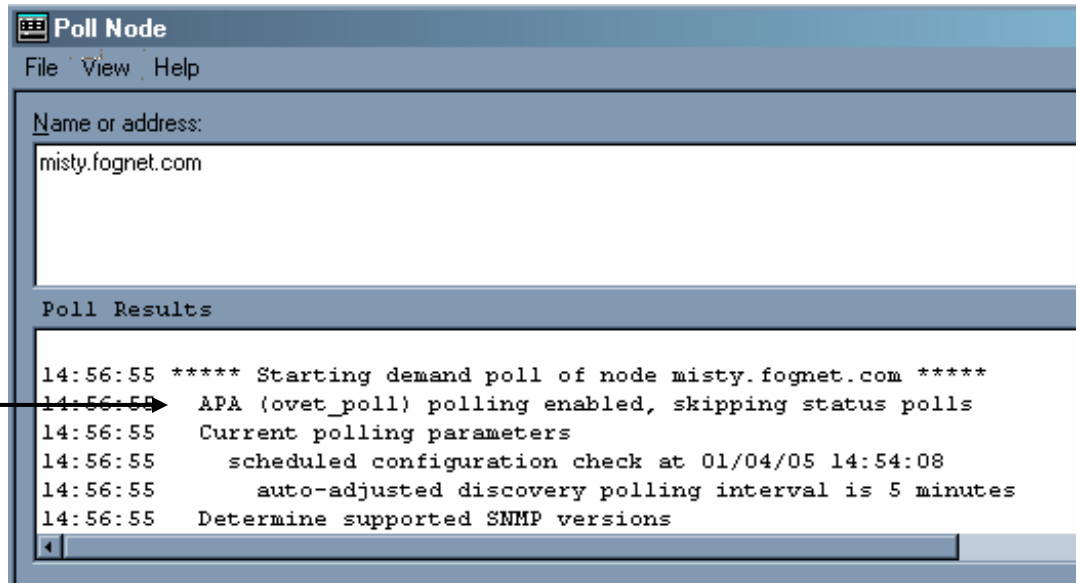
V7.01
APA on (Win):

```
C:\OpenView\NNM\conf>ovet_apaConfig.ovpl -query APAPolling
PollingEngine PollNormalIP Bool true
StatusBridge StatusBridgeEnabled Bool true
```

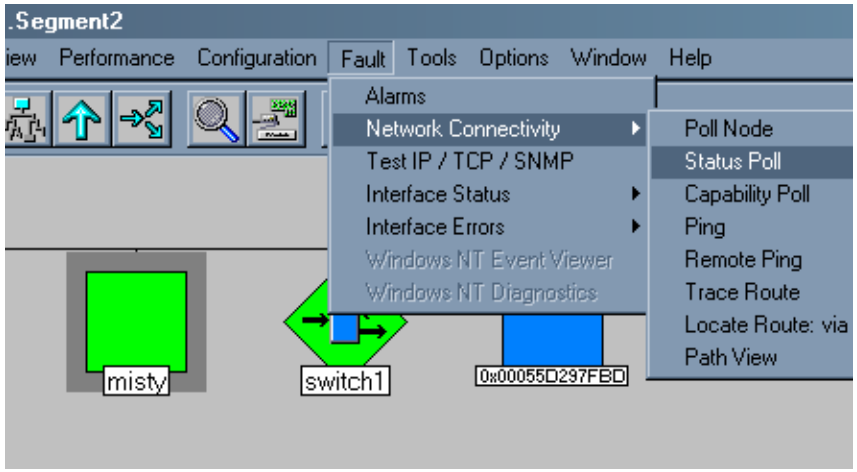
V7.5:
APA on (Win):

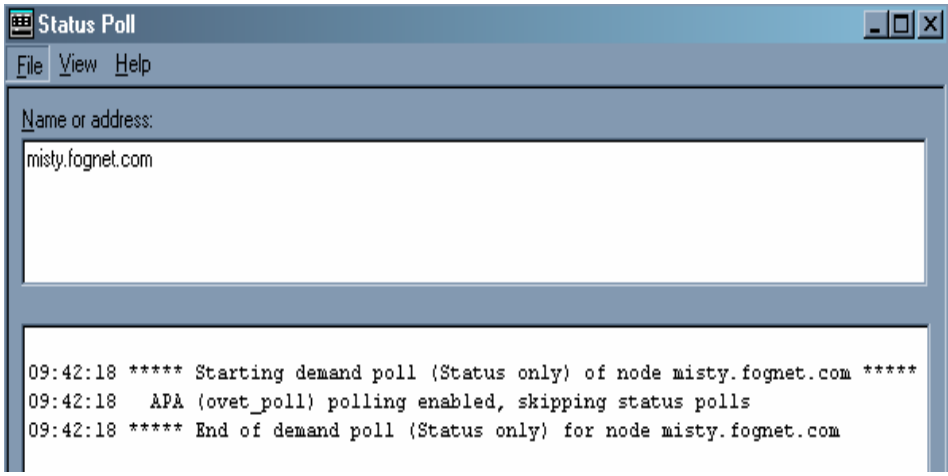
```
C:\>ovet_apaConfig.ovpl -query APAPolling
PollNormalIP true
StatusBridgeEnabled true
```

Fault ->
Network Connectivity ->
Poll Node
(nmdemandpoll)

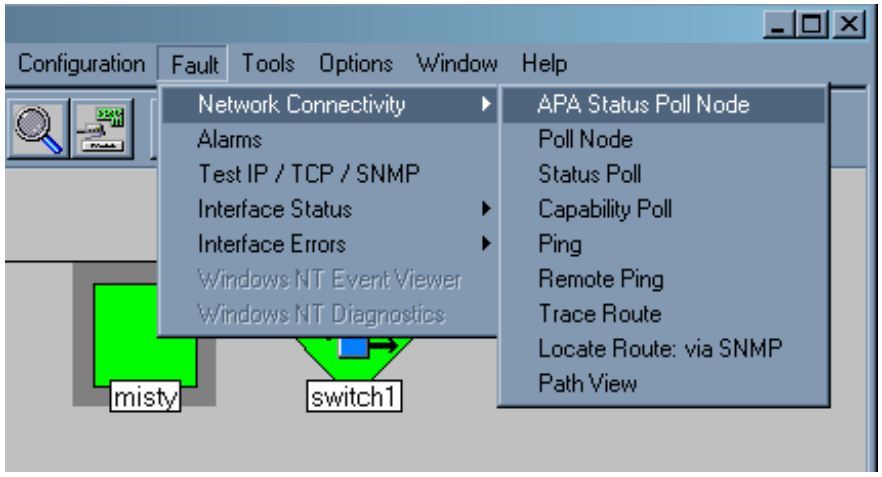


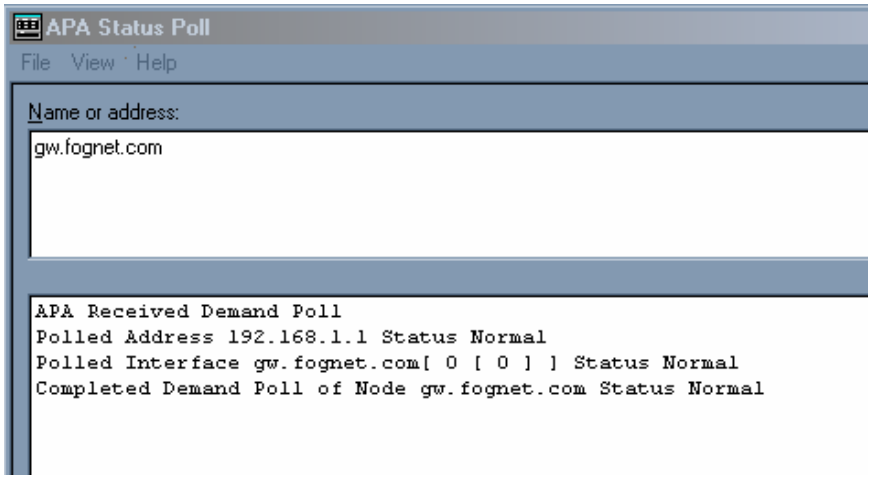
Issuing Ad hoc Status Polls: V7.01:





V7.5: ovet_demandpoll.ovpl





APA status events - overview

- APA-generated status events: `OV_APA_IF_DOWN (58983012)`
  - ICMP and/or SNMP polls related to address, interface, Node, Connection
  - AggPort and Board-level's distinguished in NNM 7.5
  - Polling granularity defined by ET Topology filters (7.01 defaults below)
  - Unreachable refers to secondary entity failure status

Status States:	ADDRESS	CONNECTION	INTERFACE	NODE	BOARD	AGGPORT
Down	Critical	Critical	Critical	Critical	Critical	Critical
Up	Normal	Normal	Normal	Normal	Normal	Normal
Unreachable	Warning	Warning	Warning	Warning	Warning	Warning

- NNM 7.01 filtered polling matrix (NNM 7.5 matrices on slides 30 & 31)

	IsRouter	isSwitch	isEndNode	UncRtrIf	UncSwchIf	UncEndNode	NotConnIf
snmpEnable	true	true	false	true	false	false	false
pingEnable	true	false	true	true	false	true	false

APA status events varbinds (Node, Connector, Address, Interface)

Varbind #	Description
$2	Timestamp event occurred *Note this exception to general rule
$3	Hostname of node that caused the event
$5	Label of the responsible interface
$6	ifAlias of the responsible interface
$8	ifIndex of the responsible interface
$9	ifDescr of the responsible interface
$10	Responsible Level 3 address or port #
$11	Responsible Level 2 address
$12	Number of bits in the Subnet Mask
$13	Route Distinguisher
$15	Capabilities
$16-$28	Varbinds associated with double-object failures if connector failure
$29-$42	Varbinds associated with primary failure if a secondary failure

- Event text: **`IF Down $5 $10 $6 Capabilities: $15`**
- Event text: **`Node Down $10 Capabilities: $15`**
- Event text: **`Address Down $5 $10 $6 Capabilities: $15`**
- Event text: **`Connection Down $5 $10 connected to $16 $18 $23 Capabilities`**

New APA events -Aggregated Port – 7.5

- Multiple Physical ports = an ET trunk virtual port
- Support for Cisco PAgP only
- Pre- NNM Trunk support now called “redundant connection support”
- Fine tuning via the following ET Topology filters

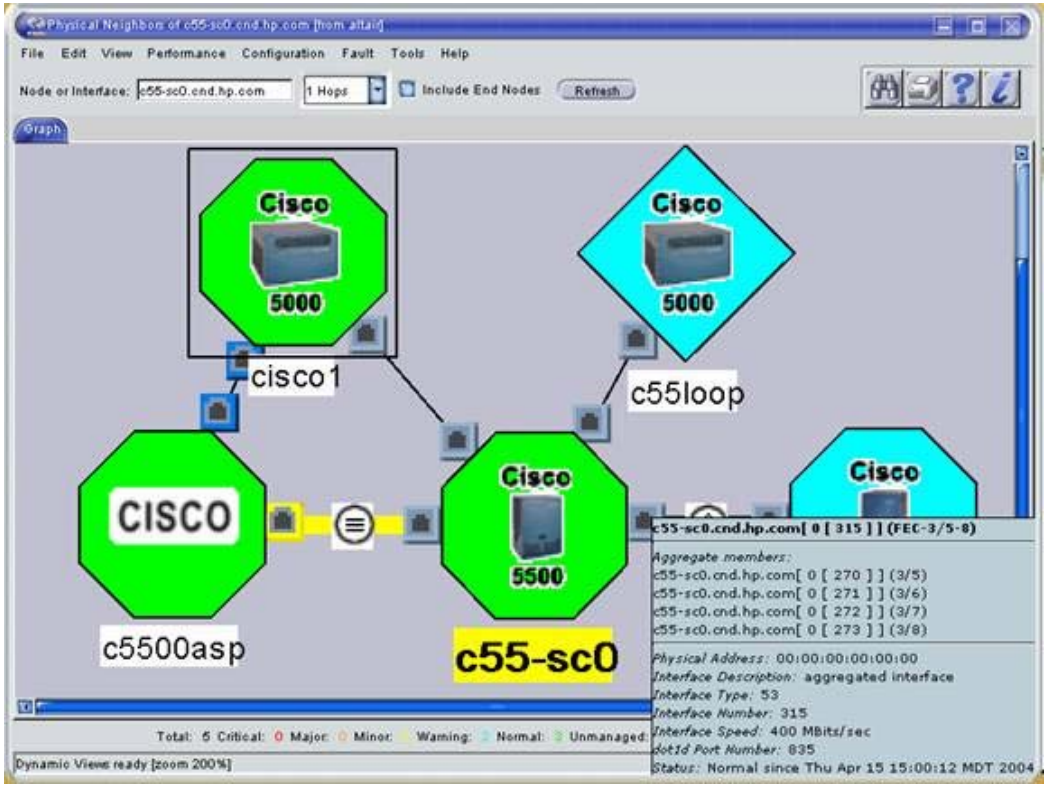
  isAggregatedIF

  virtual interface

  isPartOfAggregatedIF

  physical interface

- Scenario:
  - One physical port goes down on a trunk; TrunkDegraded event issued; Trunk virtual port status changes to Minor in ET; Physical Interface changes to Critical; Interface Down APA event correlated/embedded by ConnectorDown correlation



New APA events -Aggregated Port – 7.5


```
OV_APA_AGGPORT_DEGRADED
OV_APA_AGGPORT_DISABLED
OV_APA_AGGPORT_DOWN
OV_APA_AGGPORT_NOTDEGRADED
OV_APA_AGGPORT_UNREACHABLE
OV_APA_AGGPORT_UP
OV_APA_AGGPORTCONN_DOWN
OV_APA_AGGPORTCONN_UP
```

•Descriptions:

- **Degraded**: The aggregate port connection between two nodes is responding to polls and some of the interfaces are down.
- **Disabled**: the primary aggregated port is not responding to polls in a normal fashion. This could be because all the interfaces' ifAdminStatus are Down|Testing.
- **Down**: the aggregate port connection between two nodes is not responding to polls and all interfaces on this side of the connection may be down.
- **Unreachable**: The aggregate port connection between two nodes is not responding to polls. The problem is due to another entity.
- **Connection Down**: the aggregate port connection between two nodes is not responding to polls and all interfaces may be down on both sides of the connection.

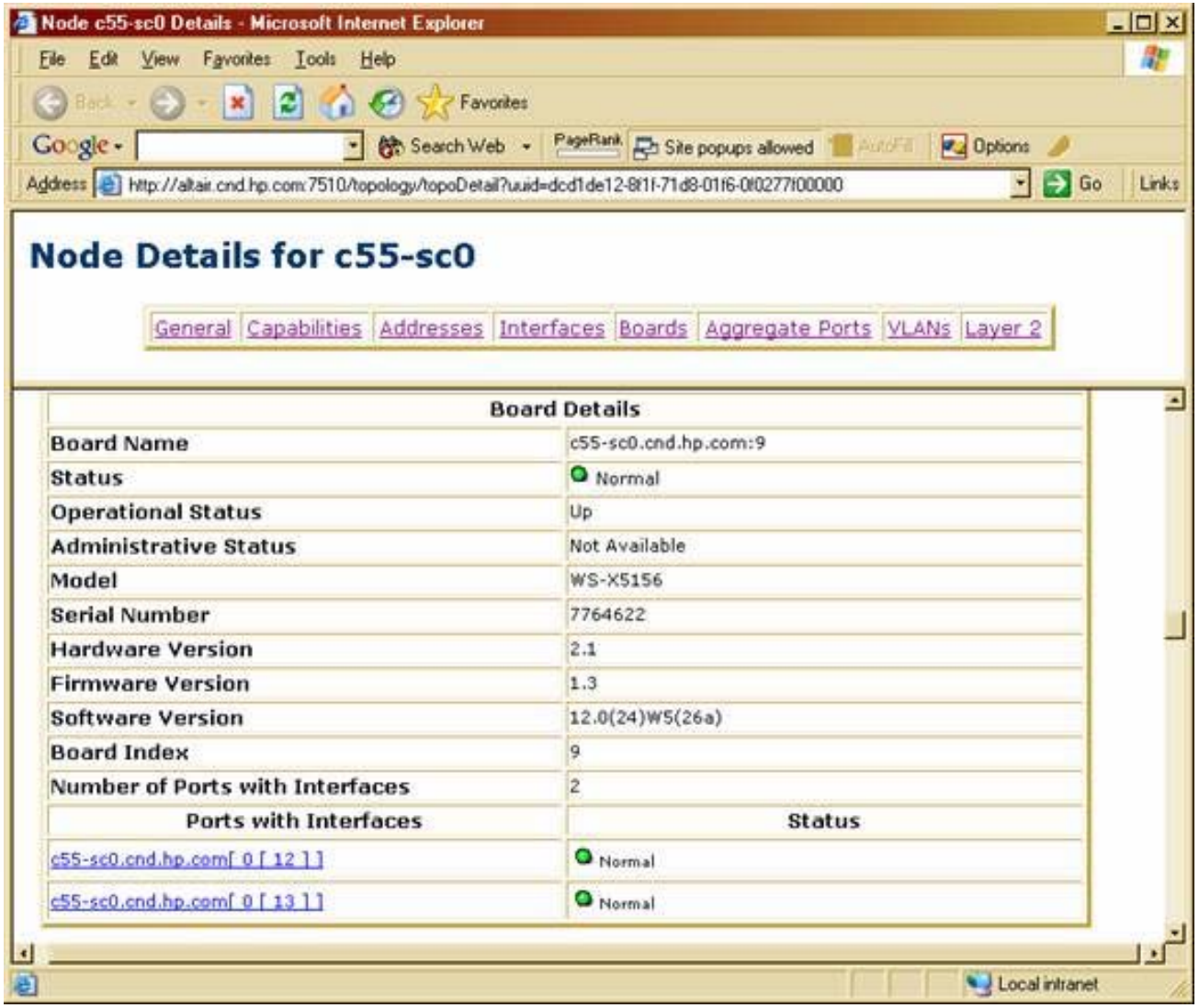
APA Aggregated Port status events varbinds (NNM 7.5)

Description	Entity 1 Varbind #	Entity 2 Varbind #
Timestamp event occurred	$2	$2
Hostname of node that caused the event	$3	$21
Label of the Aggregated interface	$5	$23
ifAlias of the responsible interface	$6	$24
ifIndex of the responsible interface	$8	$26
ifDescr of the responsible interface	$9	$27
Number of contained interfaces down	$10	$28
Number of contained interfaces	$11	$29
Management address of host	$12	$30
Number of bits in the subnet mask	$13	$31
Route Distinguisher	$15	$33
Capabilities	$16	$34

- Event text: **`Aggregate Port Down $5 $6 connected to $21 $23 $24`**

New APA events – Board Entities - 7.5

- Only Cisco Stack, Rhino and C2900 MIBs Supported
- subBoards treated as boards in 7.5
- Look for more in NNM 8.0
- Unreachable means secondary failure



OV_APA_BOARD_DOWN
OV_APA_BOARD_REMOVED
OV_APA_BOARD_UNREACHABLE
OV_APA_BOARD_UP

APA Board status events varbinds (NNM 7.5)

Varbind #	Description
$2	Timestamp event occured
$3	Name of the node that contains the board
$5	Capabilities
$6	Management Address
$7	Route Distinguisher
$9	index of the responsible board
$11	subBoard index of the responsible board
$13	Serial Number
$14	Module Name
$15	Module Description
$16	Hardware Version
$17	Software Version

- Event text: **`Board Down - $14`**

APA status events summary

OV_APA_ADDR_DOWN
OV_APA_ADDR_Intermittent
OV_APA_ADDR_UNREACHABLE
✡ OV_APA_ADDR_UP

OV_APA_CONNECTION_DOWN
OV_APA_CONNECTION_Intermittent
OV_APA_CONNECTION_UNREACHABLE
✡ OV_APA_CONNECTION_UP

OV_APA_IF_DISABLED
OV_APA_IF_DOWN
OV_APA_IF_Intermittent
OV_APA_IF_UNREACHABLE
✡ OV_APA_IF_UP

OV_APA_NODE_DOWN
OV_APA_NODE_Intermittent
OV_APA_NODE_RENUMBERING
✡ OV_APA_NODE_RENUMBERING_FIXED
OV_APA_NODE_SNMP_NOT_RESPONDING
OV_APA_NODE_UNREACHABLE
✡ OV_APA_NODE_UP

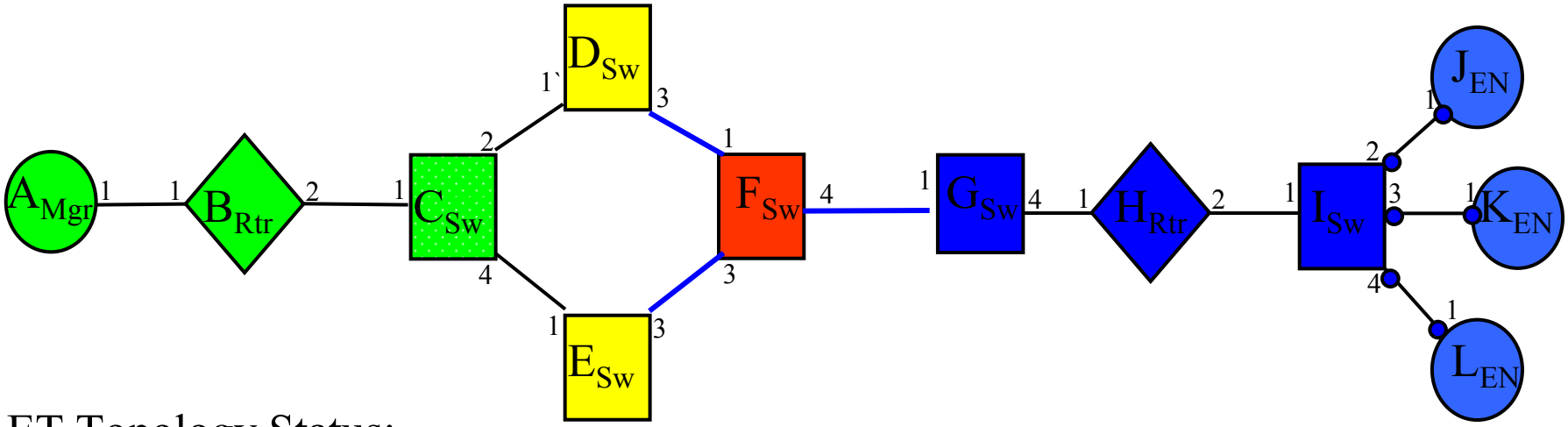
OV_APA_BOARD_DOWN
OV_APA_BOARD_REMOVED
OV_APA_BOARD_UNREACHABLE
✡ OV_APA_BOARD_UP

OV_APA_AGGPORT_DEGRADED
OV_APA_AGGPORT_DISABLED
OV_APA_AGGPORT_DOWN
✡ OV_APA_AGGPORT_NOTDEGRADED
OV_APA_AGGPORT_UNREACHABLE
✡ OV_APA_AGGPORT_UP
OV_APA_AGGPORTCONN_DOWN
✡ OV_APA_AGGPORTCONN_UP

✡ = "Log-Only" Events
This list excludes Informational, OAD, HSRP, & RAMS APA Events

Example of Neighbor Analysis

- Node F Down



- ET Topology Status:
  - Nodes D, E Marginal; F Critical
  - Interfaces D3, E3, F1, F3, F4, Unknown
  - Note ET-based topology status not tied to alarms – very different from IPMAP
  - Alarms correlated by ConnectorDown:
  - OV_APA_Node_Down: F:
    - OV_APA_CONNECTION_UNREACHABLE: D3-F1
    - OV_APA_CONNECTION_UNREACHABLE: E3-F3
    - OV_APA_INTERFACE_UNREACHABLE: F4

APA default status configuration file: paConfig.xml

- set in $OV_CONF/nnmet/paConfig.xml
- Schema defined in paConfigSchema.xsd
- Changes take affect when *ovet_poll* process restarted with ovstart
- Backup pxConfig.xml file before making changes
- Simplified schema with parameterList examples:

```
<paConfig>
  <subSystemConfig> PollingEngine, StatusAnalyzer, Talker, StatusBridge
    <globalParameters> statisticsEnable, statusAnalyzerThreadPoolSize
    <configGroupList>
      <configGroup>  pollingSettings; Traceroute; PingSettings;   ✲configPollSettings
        <generalParameters> cfaDebugLevel, GenerateDegradedEvent
        <classSpecificParameters>
          <defaultParameters> interval; snmpEnable ; timeout
          <classSpecification> isRouter, isSwitch, isEndNode
            <parameterList> interval; snmpEnable ; timeout
```

✲ = New in 7.5

APA configuration file subsystems

- PollingEngine
  - Manages APA Tasks
  - Issues ICMP and SNMP polls
- StatusAnalyzer
  - Queues poller results (states)
  - Determines connectivity faults
  - Generates events based on poller results
- StatusBridge
  - Communicates status info to ovw topology DB and IPMAP
- Talker
  - Communicates, via talker modules, to ET device-specific processes

Interesting paConfig.xml polling settings – global

- SubSystemConfig: PollingEngine
- ConfigGroup: PollingSettings
- Global parameters:
  - PollNormalIP (false); Only OAD, HSRP entities polled by default
    - Set to true by running `ovet_apaConfig.ovpl -enable APAPolling`
  - RecieveEvents (false); Receive link down/link up traps
  - StatisticsEnable (true)
  - StatisiticsInterval (300); seconds
  - PollingEngineThreadPoolSize (16); for performance tuning
    - Increasing this consumes more system resources

paConfig.xml polling settings – Class Specific – 7.01

- SubSystemConfig: PollingEngine
- ConfigGroup: PollingSettings
- ClassSpecific *default* parameters
  - Interval (300); snmpEnable (true); pingEnable (true); hsrpEnable (true)
- ClassSpecifications shipped with NNM 7.01

isRouter	isSwitch
isEndNode	UnconnectedAdminUpRouterIf
UnconnectedAdminUpSwitchIf	UnconnectedEndNode
NotConnectedIF	

  - Only customizations/variations are in snmpEnable and pingEnable booleans

7.01 Only	IsRouter	isSwitch	isEndNode	UncRtrIf	UncSwchIf	UncEndNode	NotConnIf
snmpEnable	true	true	false	true	false	false	false
pingEnable	true	false	true	true	false	true	false

APA status events – overview (cont'd)

•NNM 7.5 filtered polling matrix (Page 1 of 2):

Class Specification	snmpEnable	pingEnable	Filter Definitons
APANoPollNodes ✡	false	false	Do not poll entries in APANoPollNodes.xml
isIpPhone	false	false	46xx series Avaya phones w/ IP Telephony SPI
ifsWithAnycastAddrs	n/a	false	Interfaces with duplicate IP addresses (DNA)
isRouter	true	true	Inherited from NNM topodb capability flag
AvayaIptDevices	true	true	Avaya IPT equipment, e.g. s8700CM, etc.
NotConnectedSnmpSwitch	n/a	true	ET finds no L2 connection to a managed device
isSwitch	true	false	Inherited from NNM topodb capability flag
isEndNode	false	true	Not a switch or a router
WanIf ✡	true	false	Defined in filters xml via ifType & ifSpeed
IfTypeFilter ✡	n/a	false	Placeholder for filters based on ifType

DNA = HP Recommends Do Not Adjust

✡ = "commented out" xml file definitions

APA status events – overview (cont'd)

•NNM 7.5 filtered polling matrix (Page 2 of 2):

Class Specification	snmpEnable	pingEnable	Filter Definitons
isPartOfAggregatedIF	true	n/a	Interfaces not port aggregating (DNA)
IFInNotConnectedSwitch	n/a	true	ET finds no connection to mgd device
UnconnectedAdminUpOrTestRouterIf	true	true	Unconnected If is admin up or testing
UnconnectedAdminUpOrTestSwitchIf	false	false	Unconnected If is admin up or testing
UnconnectedAdminDownRouterIf	false	false	Unconnected If is admin down
UnconnectedAdminDownSwitchIf	false	false	Unconnected If is admin down
UnconnectedEndNode	false	true	Unconnected and isEndNode
NotConnectedIF	false	false	Matches any unconnected interface
AllBoards ✡	false	n/a	Currenly only Cisco boards supported
NoPingAddresses ✡	n/a	false	Do not ping addrs matching this filter

DNA = HP recommends Do Not Adjust

✡ = "commented out" xml file definitions

paConfig.xml polling settings – topology filters

- ClassSpecifications defined using extended topology filters
- Extended topology filters are separate entity from NNM traditional filters
- Run `ovet_topodump.ovpl -lfilt` to see a list of all existing filters.
- To see a dump of discovered devices that pass a given filter, run:
  - `ovet_topodump.ovpl -node -filt [filtername]`

```
C:\>ovet_topodump.ovpl -node -filt isSwitch
  Name                          Status    CommAddr       PrivateAddr       OADId     IPVer Index
  switch1.fognet.com            Normal    192.168.1.3    -                 0         4
```

- ClassSpecification filters are evaluated in xml file order
  - device matching isSwitch *and* isRouter: isRouter rules apply
  - ClassSpecifications can refer directly to ET Filter *filterNames* entities
- Extended Topology Filters
  - Defined in $OV_CONF/nnmet/topology/filter/TopoFilters.xml
    - Similar filter definition logic to NNM filters, only in xml

Troubleshooting Class Specification Filter Mappings

- Most common problems relate to mapping of isSwitch and isRouter
  - isSwitch and isRouter flags assigned by *netmon* during discovery
  - Check Firewalls for SNMP/ICMP blocking.
  - Force isRouter with G flag; isSwitch with B flag in oid_to_type
  - Browse SNMP MIBS; check for cut tables that may prevent visbility to:

.1.3.6.1.2.1.1	systemTable	OID for oid_to_type
.1.3.6.1.2.1.17.1	dot1dBaseTable	isSwitch
.1.3.6.1.2.1.4.20.1	ipAdEntTable	
.1.3.6.1.2.1.2.1	ifTable	isRouter/isSwitch
.1.3.6.1.2.1.31.1.1.1.1	ifName	
.1.3.6.1.2.1.31.1.1.1.18	ifAlias	
.1.3.6.1.2.1.4.1.0	ipForwardingTable	isRouter
.1.3.6.1.2.1.4.21	ipRouteTable	isRouter
.1.3.6.1.2.1.4.22.1	ipNetToMediaTable	isSwitch
.1.3.6.1.2.1.3.1.	atTable	

  - Example query:

```
snmpwalk <target> 1.3.6.1.2.1.1
```

paConfig.xml polling settings – Engine Settings

- SubSystemConfig: PollingEngine
- ConfigGroup: TraceRoute
- ClassSpecific default parameters:
  - timeout (3000); milliseconds
  - minTimeToLive (1); initial ttl in first outgoing probe packet
  - maxTimeToLive (30); max ttl (max number of hops)
  - maxTimeOuts (0); Max number of timeouts before ending traceroute
    - 0 = never

- SubSystemConfig: PollingEngine
- ConfigGroup: PingSettings
- ClassSpecific default parameters:
  - timeout (1000); milliseconds
  - numberOfRetries (2);
    - Timeout redoubles on every retry, so default = 7 seconds

paConfig.xml polling settings – Engine Settings (Cont'd)

- SubSystemConfig: PollingEngine
- ConfigGroup: ✡ ConfigPollSettings
- ClassSpecific default parameters:
  - interval (86400); seconds (1 day)
  - enable (true); enable/disable configuration poll of a device
  - interfaceDetailFields (ifAlias,ifName,ifPhysAddress,ifDescr);
    Fields used in interface re-numbering check (default is all possible)
  - boardDetailFields (serialNumber); fields used in board renumbering check (csv)
- Class Specific Parameters:
  - APANoPollNodes filter as placeholder (commented out in topoFilters.xml)

- Uses ovet_demandpoll.ovpl

✡ = New in NNM 7.5

paConfig.xml Status Analyzer settings

- SubSystemConfig: StatusAnalyzer
- GlobalParameters:
  - ✲ validateOnStartup (false); perform full ovet_poll validate and status bridge sync for every node in topology. set to true only in stable and smaller scaled environments: this operation is very cpu intensive
  - statusAnalyzerThreadPoolSize (10)
  - statusAnalyzerQueueSize (65000); input queue: repository of poll results

✲ = New in NNM 7.5

paConfig.xml Fault Analyzer Settings - 7.01

- ConfigGroup: ConnectivityFaultAnalyzer; Distinguishes primary/secondary failures
- General Parameters:
  - cfaStpConvergenceTimeSecs (50); time to delay polling for STP Convergence
- 7.01 ClassSpecific default parameters:
  - analysisMaxNumberRetries (1); override snmp config timeouts during analysis
    - Issues one retry to reduce expected SNMP timeouts during analysis
  - importantNodeUpToDown (false); do not suppress/embed 2ndary Node Downs
  - importantNodeDownToUp (false); do not suppress/embed 2ndary Node Ups
- 7.01 ClassSpecifications:
  - isRouter, uses all default values (example)
- 7.5 ClassSpecific default parameters:
  - isImportantNode (false); suppress/enable secondary failure reporting
    - True: node or connection goes down and is symptomatic: primary alarm generated
    - False: node/connection goes down and is symptomatic: Symptomatic alarm embedded
- 7.5 ClassSpecifications:
  - ImportantNodes: filterName for important nodes entered into MyHostID.xml file

= deprecated in NNM 7.5

= new in NNM 7.5

paConfig.xml HSRP and Talker Settings

- SubSystemConfig: StatusAnalyzer
- ConfigGroup: HSRP
- General Parameters:
  - HSRPTransientWait (60000); Milliseconds to wait for failover to stabilize (1 min)
  - GenerateNoStandbyEvent (true); generate “No Standby” events
  - GenerateDegradedEvent (true); generate “degraded” events
  - GenerateFailoverEvent (true); generate failover events
  - GenerateStandbyChangedEvent (true); generate “standby changed” events

- SubSystemConfig: Talker
  - ovet_poll uses various talker modules to perform communication tasks
- ConfigGroup: SnmpTalker
- General Parameters:
  - snmpTalkerSessionCacheSize (32); # of open SNMP sessions
    - Increase to speed polling performance at expense of system resources
  - NumberOfOIDsPerPDU (50); controls size of SNMP PDUs

paConfig.xml Status Bridge Settings

- SubSystemConfig: StatusBridge
- Global Parameters:
  - StatusBridgeEnabled (false); NNM topology status is owned by ET
    - Set to true by running `ovet_apaConfig.ovpl -enable APAPolling`
  - FullTopoSync (false); Full topology synchronization at initialization
- ConfigGroup: BridgeSettings
- Default Parameters:
  - PrimaryStatusOnly (false); NNM topology changes only reflected for primaries
  - SecondaryFailureStatus (ET); If PrimaryStatusOnly = false, this value controls what status will be reflected into NNM for symptomatic failures. By default, the status will match whatever ET connectivity fault analysis determines the status to be. Allowable values are "ET", "Critical", and "Unknown".
  - CorrelateSecondaryFailures (true); By default, if a symptomatic failure is being reflected into the NNM topology (PrimaryStatusOnly = false), then the status bridge will attempt to correlate the interface failure with a primary interface failure. If set to false, then the status bridge will not attempt to correlate symptomatic failures, with the result that the status message will be a primary failure in NNM

APA XML configuration file best practices

1. Make a backups of the paConfig.xml file, track revisions for reverting
2. Keep offline documentation of *netmon* and APA configuration customizations
3. Use `ovet_topodump.ovpl` to test that the nodes or interfaces pass the class filter
4. Validate your XML syntax using your favorite xml editor or any web browser
5. Test changes using `checkPollCfg` in the support subdirectory
6. Restart the ovet_poll process to begin using the new settings

APA Polling Statistics

- Collected/updated on 5 minute intervals – details in APA white paper
  - OV_APA_Statistics log-only event also reports these (see ovdumpevent output)
  - Statistics available from home base main window

```
1082644801 1 Thu 04 22 10:40:01 2004 patchy.fognet.com  p OV_APA_Statistics APA
stats: Addresses_Polled:5  CfaAddr:0  CfaAnalysisTime:0.000000  CfaIface:0  CfaN
ode:0  CfaSubnet:0  CfaTasks:0  HSRP_AnalysisTime:0.000000  HSRP_Tasks:0  Interf
aces_Polled:2  PAOC_NumBusyObjects:0  PAOC_NumBusyReferences:0  PE_HSRPGroupsPol
led:0  PE_QueueUsage:0  PE_TasksProcessed:6  PE_TimeOnQueue:0.000000  PE_TimeOnQ
ueueAvg:0.000000  SA_ActiveWorkers:0  SA_BlockedEntries:0  SA_QueueSize:65000  S
A_QueueUsage:0  SA_ThreadNum:10  SA_TimeOnQueue:0.000000  SA_TimeOnQueueAvg:0.00
0000  SA_TimeOnQueueOld:0.000000  SA_WorkProcessed:0  SA_WorkersWaiting:10  ;1 1
7.1.0.58983032 0
```

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

Statistic	Current	Max	Min	Max Time
Active Analyzer Tasks	0	0	0	Mar 18, 20
Waiting Poller Tasks	0	0	0	Mar 18, 20
Interfaces Polled (SNMP)	2	2	2	Mar 18, 20
Addresses Polled (ICMP)	5	8	5	Mar 18, 20
HSRP Groups Polled	0	0	0	Mar 18, 20
Waiting Analyzer Tasks	0	0	0	Mar 18, 20

Determining a Device's polling setttings using checkPollCfg

- -o option for specific objects:

```
C:\OpenView\NNM\support\NM>checkPollCfg -o switch1

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

switch1.fognet.com                        -                           -        -     0          -          -0
-                                  IF   [    4]          300          1              0          1     0        -
-                                  IF   [    1]          300          1              0          1     0        -
 192.168.1.3                              -              300          0              0          0     0        -
-                                  IF   [    2]          300          0              0          0     0        -
-                                  IF   [    6]          300          0              0          0     0        -
-                                  IF   [    8]          300          0              0          0     0        -
```

- -A option for all
- -l [ell] option for summary:

```
C:\OpenView\NNM\support\NM>checkPollCfg -l


  Number of unpolled interfaces : 24 ( total  28 )
  Number of unpolled addresses  : 1 ( total  3 )
  Number of unpolled cards      : 0 ( total  0 )
```

Fine Tuning a Device's polling setttings using ovet_toposet

- -a option allows APA polling for that entity
- -s option suppresses APA polling for that entity
- Use in conjunction with ovw `ovtopofix -G` to unmanage devices
- Not available in 7.01
- Usage:

```
C:\OpenView\NNM\support\NM>ovet_toposet
Usage: ovet_toposet <command>
 Where <command> can be any of the following :
        [-h]  Prints this help message
        [-s|-a] [-node   <nodeName>]
        [-s|-a] [-nodeif <nodeName>]    [-if <Extended Topology ID|Interface Name>]

        [-s|-a] [-board  <nodeName>]    [-index<index>] [-subindex<subindex>]
        [-s|-a] [-addr ]        [IPv4 IP Address]       [-OADId<OADId>]
```

ovet_toposet example: Stopping APA polling to an interface


```
C:\OpenView\NNM\support\NM>checkPollCfg -o sunny

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

sunny.fognet.com                        -                   -         -      0             -         -0
-                                       IF   [    2]       300        1      0         0      1       -
 192.168.1.6                            -                  300        1      0         0      1       -
-                                       IF   [    3]       300        1      0         0      1       -
 192.168.1.7                            -                  300        1      0         0      1       -
-                                       IF   [    1]       300        0      0         0      1       -
```

```
C:\OpenView\NNM\support\NM>ovet_topodump.ovpl -nodeif sunny
  Name                           Status    CommAddr          PrivateAddr    OADId    IPVer  Index
  sunny.fognet.com               Normal    192.168.1.6       -              0        4
    hme0                         NotMon    -                 -              0        4      2
      -                          Rspd      192.168.1.6       -              0
    hme0:1                       NotMon    -                 -              0        4      3
      -                          Rspd      192.168.1.7       -              0
    lo0                          NotMon    -                 -              0        4      1
```

```
C:\OpenView\NNM\support\NM>ovet_toposet -s -nodeif sunny -if hme0:1

C:\OpenView\NNM\support\NM>checkPollCfg -o sunny

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

sunny.fognet.com                        -                   -         -      0             -         -0
-                                       IF   [    2]       300        1      0         0      1       -
 192.168.1.6                            -                  300        1      0         0      1       -
-                                       IF   [    3]       300        0      1         0      1       -
 192.168.1.7                            -                  300        0      1         0      1       -
-                                       IF   [    1]       300        0      0         0      1       -
```

Forcing ET/APA to Recognize New Objects

- After discovering new objects, or forcing via loadhosts:
  - netmon discovered devices show in IPMAP topology, but not ET topology
  - No command to force an object into ET Topology
- Crude methods: ovstop/ovstart will force update, as will etRestart.ovpl
- Better: from home base, select Discovery Status, then ET Configuration
  - Initiate full discovery now
  - Or change “enable discovery for a specified number of NNM changes”
    - Default threshold is 2500 – restart everything after modifying.

paConfig.xml Example: Disable ICMP polling on a Firewall

- This requires establishing a new node assertion and new class specification.
- Steps:
  - Backup paConfig.xml and TopoFilters.xml
  - Determine the SNMP sysObjectID of the firewall:
    - Neighbor View, Right Click, details, or
    - `snmpget -T <firewallName> system.sysObjectID.0`
  - In TopoFilters.xml, copy and paste an entire OID-based node assertion block
    - Change the name, title, description and OID block to match your firewall device
  - Check xml syntax and confirm filter matches your devices by running:
    - `ovet_topodump.ovpl -node -filt <newNodeAssertionName>`
  - In paConfig.xml, copy entire isRouter ClassSpecification; paste *before* isRouter
    - Change the ClassSpecification Name to match new nodeAssertion filter name
    - Change pingEnable parameter to false
  - Check xml syntax and confirm polling settings updates for firewall by running
    - `$OV_SUPPORT/checkPollCfg -o <firewallName>`
  - Apply your changes by running ovstop/ovstart on ovet_poll

paConfig.xml Example: Filtering by ifType

- Two default filters shipped with paConfig.xml in NNM 7.5, both commented out
  - IfTypeFilter in paConfig.xml sets Ping to false; useful to prevent polls "waking" ifs
  - WanIf stops polling of matching interfaces to suppress *connection* level events
    - WanIf is defined as wanIfTypes filter anded with slowIfSpeeds filter
    - slowIfSpeeds include: 9k, 16k, 56K, 64K
- To Enable:
  - In paConfig.xml remove comments at end and beginning of filter definition
  - Modify default ifTypes in TopoFilters.xml if desired
  - ovstop ovet_poll; ovstart ovet_poll
- IfTypeFilter definition in TopoFilters.xml:

```
<interfaceAssertion name="IfTypeFilter" title="IfTypeFilter" description="Interface are of a
particular ifType">
        <operator oper="OR">
            <attribute>
                <ifType>28</ifType>
            </attribute>
            <attribute>
                <ifType>20</ifType>
            </attribute>
        </operator>
    </interfaceAssertion>
```

ifTypes used to suppress using WanIf or IfTypeFilter

- Complete/Official list at: http://www.iana.org/assignments/ianaiftype-mib

  - ifTypes used by IfTypeFilter:
    - 20 - basicISDN
    - 28 - SLIP (Serial Line IP)
  - ifTypes used by WanIf and wanIfType:
    - 23 - ppp
    - 63 - ISDN and X.25 (basic rate ISDN)
    - 75 - ISDN S/T interface
    - 76 - ISDN U interface
    - 77 - lapd, Link Access Protocol D
    - 81 – ds0
  - Additional ifTypes of interest
    - 18 - ds1 (t1 carrier)
    - 21 – primaryISDN
    - 22 - proprietary Point to Point Serial, found on voip routers
    - 53 - proprietary virtual/internal, like fxp0 on Junipers, etc.

Event correlation circuits related to NNM-derived status

- NNM 6.0:
  - ConnectorDown differentiates primary from secondary failures
  - Repeated Event applied to Node_up
  - Pair-wise applied to many status events
- NNM 6.2:
  - No Changes to ECS
- NNM 6.31:
  - NodeIF supplements Connector Down, AKA “Router/Switch Health”
  - Pair-wise behavior updated ; IntermittentStatus added
- NNM 6.41
  - De-duplication applied to status events
  - Intervals, some ECS circuit parameters changed to reduce status alarms
- NNM 7.01
  - APA-based events added as sources for ConnectorDown, de-dup, and PairWise
  - OV_PollerPlus correlator added as contributed app to supplement APA

Event correlation configuration entry points:

- Correlation Manager:
  - http://host.fqdn/OvDocs/C/ecs/ecscmg.html
  - First Event correlation GUI shipped with NNM's ECS Runtime
  - Available from Options-> Event Configuration -> Edit -> Event Correlation
- Correlation Composer:
  - A "Super Circuit" providing generalized "instances":
    - Suppress, enhance, rate, repeated, transient, multiple source
  - Allows ECS custom logic to be built within composers fixed logic sets
  - Defines 3 "namespaces": OV_NNM_Basic; OV_NodeIf; OV_Poller
  - Operator Mode: **`ovcomposer -m o`** Launch from Correlation Manager
  - Developer Mode: **`ovcomposer -m d`** Launch from command line
- ECS Designer:
  - Separate product add-on; provides full ECS circuit development environment
- Post-processing correlator:
  - $OV_CONF/dedup.conf

Event correlation effects on APA Status

- Pairwise Circuit deletes paired status events mating within 10 minutes
- DeDup Correlation deletes repeating status events, tied to Pairwise
  - Combined Pairwise/Dedup is that "up" events always delete "downs"
- IntermittentStatus catches flapping status otherwise suppressed by above
- NodeIf Correlation affects only *netmon*-based status events
  - AKA Router/Switch health; tied to ConnectorDown only for *netmon* status
- ConnectorDown correlation logic applied to APA poller results
  - *netmon* path analysis not connected to APA poll results in any way
  - Embeds related entities, e.g., Address events embedded under If events, etc.
  - Important node filtering prevents nodes from being considered secondary
    - genannosrv provides *netmon*-based important node filtering
    - MyHostID.xml provides APA-based important node filtering
  - ovet_pathengine provides APA-based critical path analysis
    - ConnectorDown receives these results

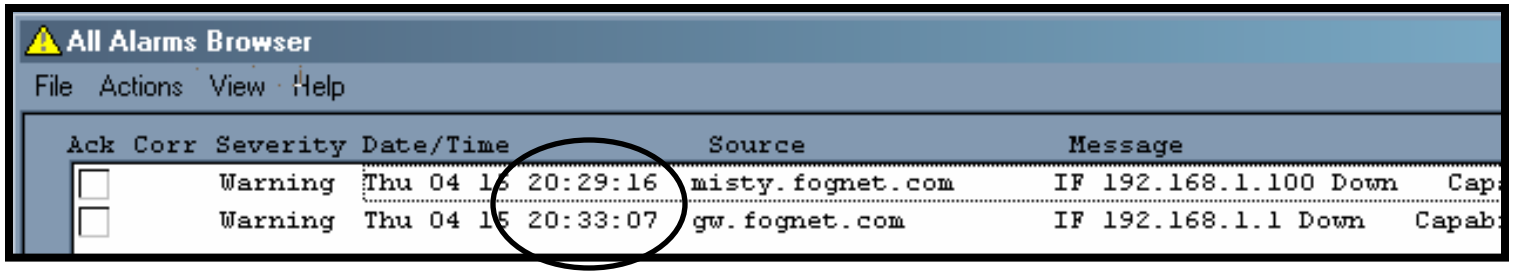
PairWise correlation behavior

- Behavior:
  - Applies to many status events by default, both *netmon* & APA-based
  - Major change to PairWise defaults in V6.31, V6.41, and again in V7.01
  - Parent events not embedded; most parents "log-only" anyway
- Status Alarms:
  - NNM V6.0, V6.1, V6.2:
    - Status alarms ***acknowledged*** if parent rec'd in PairedTimeWindow (10m)
    - Child events released immediately to alarm browser, actions launched
    - No reduction of alarms, no embedding of alarms
  - NNM V6.31, V6.41:
    - Status alarms ***deleted*** if parent rec'd in PairedTimeWindow (10m)
    - Child events ***held;*** actions launched only if and when window expires
    - No alarms seen ***at all*** if parent/child events received within window
    - Child event released with original time stamp to alarm browser after window
  - NNM V7.01, 7.5:
    - Status alarms ***deleted*** if parent rec'd in PairedTimeWindow (10m)
    - Child events ***released*** immediately to alarm browser, actions launched
    - Parent event received after window is sent to alarm browser, not embedded

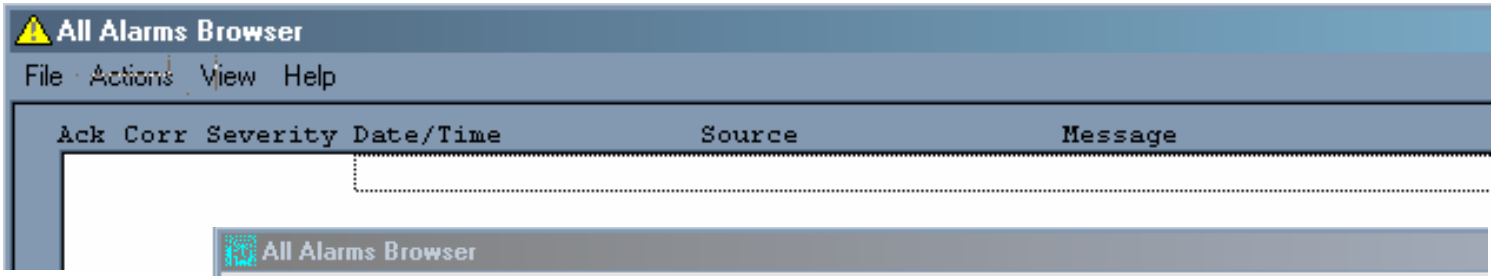
PairWise correlation example – NNM V7.01, 7.5 - *netmon*

- Example 1: NNM V7.01, *netmon* polling. IF_Down events released immediately
  - If_Down events on connector devices may be held by NodeIf, however.
- Example 2: IF_Up events for both received at 20:40
  - Within 10 minutes, alarm is deleted from browser
  - After 10 minutes, alarm deleted anyway if listed in dedup.conf
- Example 3: IF_Up after 10 minutes, alarm remains in browser w/ dedup off

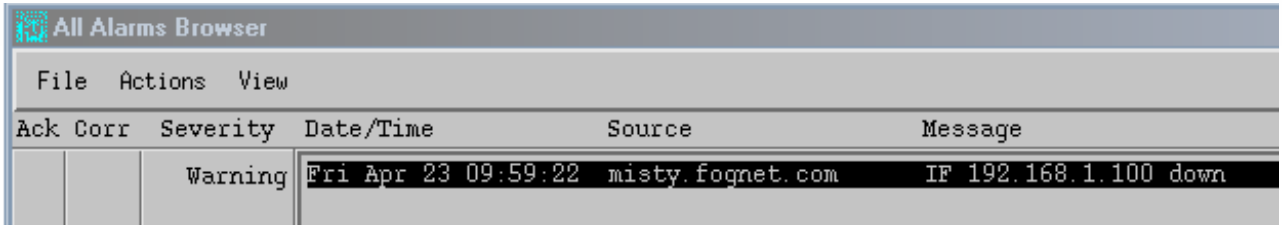
1:



2:



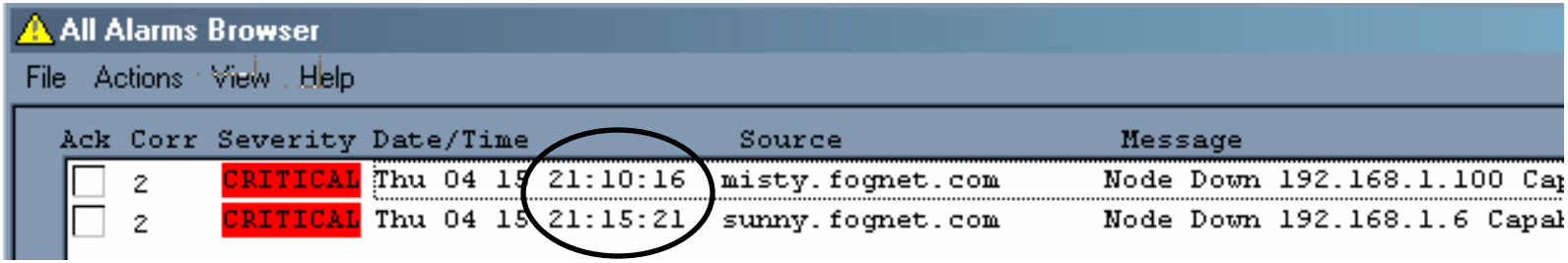
3:



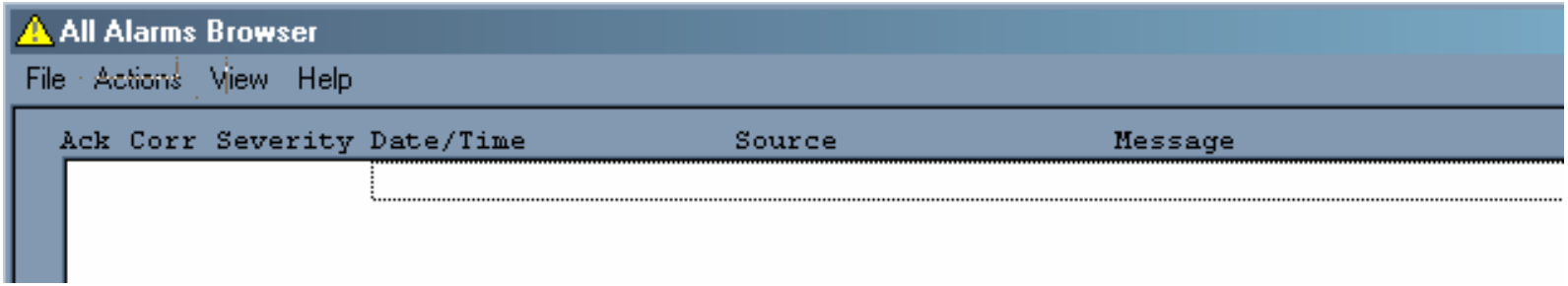
PairWise correlation example - NNM V7.01, 7.5 - APA

- Example 1: NNM V7.01, APA Polling. IF events released immediately
  - Address; Interface unreachable correlated by ConnectorDown
- Example 2: APA_Address_Up events for both received at 21:30
  - Within 10 minutes, alarm is deleted from browser by PairWise
  - After 10 minutes, alarm deleted anyway by dedup.conf

1:



2:

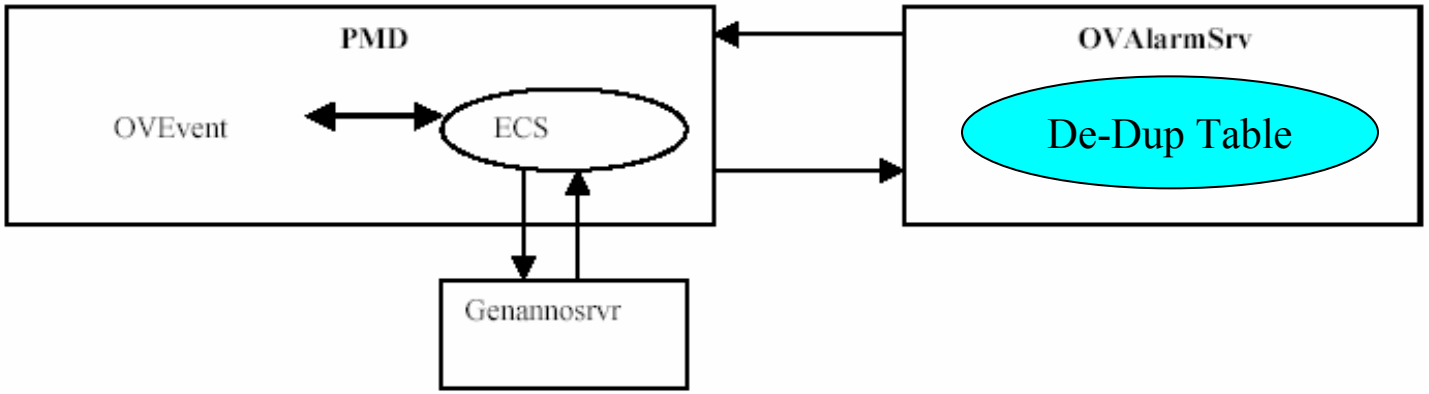


PairWise correlation - status events affected

Parent	Children
V6.0,V6.1,V6.2 (DeleteOrAck: *Acknowledge*; ChildImmediateOutput: *true*):	
Node up	Node_Marginal, _Warning, _Major, _Down
Segment_Normal	Segment_Major, _Critical
Network_Normal	Network_Warning, Network_Critical
Station_Normal	Station_Marginal, _Warning, _Major, _Critical
Remote_Mgr_Up	Remote_Mgr_Down
V6.31,V6.41 (DeleteOrAck: *Delete*; ChildImmediateOutput: *false*):	
Adds: IF_Up	IF_Down
Node_Up	Node_Unknown
V7.0,7.01, 7.5 (DeleteOrAck: *Delete*; ChildImmediateOutput: true):	
Adds: IF_Unknown	
OV_APA_ADDR_UP	_ADDR_DOWN, _ADDR_UNKNOWN
OV_APA_IF_UP	_IF_DOWN, _IF_UNKNOWN
OV_APA_NODE_UP	_NODE_DOWN, _NODE_UNKNOWN
OV_APA_CONNECTION_UP	_CONN_DOWN, _CONN_UNKNOWN

De-Dup correlation behavior

- Behavior:
  - Deletes and embeds existing matching alarm; placing latest alarm in browser
  - Match criteria includes event OID, event sources, and optionally, varbinds
  - Dedup is a post-processing correlation,
    - Fed from *OVAlarmSRV* vs. *pmd* for other ECS circuits
- Status Alarms:
  - Related status alarms may be embedded – not always "exact' match – see example
- Topology Status:
  - Remember: suppressed status alarms aren't suppressed in topology!



De-Dup correlation configuration – dedup.conf

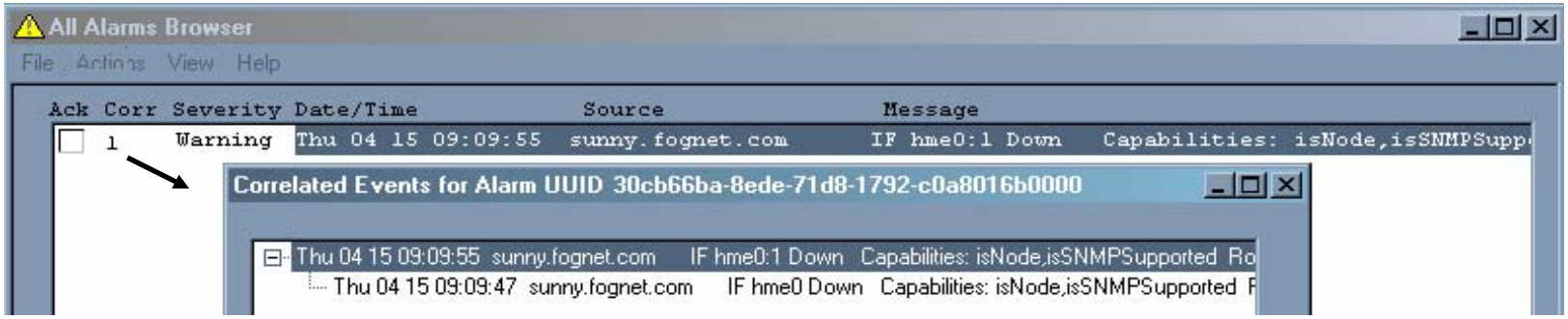
- Configuration:
  - "Internal" correlation
  - $OV_CONF/dedup.conf
  - ovstop/start ovalarmsrv after making changes
- Match event (all sources)
- Match Source
- Match Source & Varbind
- Important note added in V7.0+ here:
- Disable circuit here:

```
# Event De-Duplication Configuration file
# Format  <TrapOid[[, $r][,$NUM][,$*]]>
# Note:
# TrapOid is the oid that identifies the event to be de-duplicated
# $r is the event source
# $NUM is to specify the varbind number. 1<= NUM <=16
# $* is for all varbinds
#
# De-Dup Examples:
# <.1.3.6.1.4.1.11.2.17.1.0.59179225>
# <.1.3.6.1.4.1.11.2.17.1.0.59179225, $r>
# <.1.3.6.1.4.1.11.2.17.1.0.59179225, $r, $2>
# <.1.3.6.1.4.1.11.2.17.1.0.59179233, $r, $1, $2>
# <.1.3.6.1.4.1.11.2.17.1.0.59179225, $r, $*>
#
# PLEASE NOTE:  The dedup.conf configuration is also used to
# determine what events will be deleted by the pattern delete
# feature of the PairWise ECS circuit.  (This feature deletes
# old events from the browser when they have been "cancelled"
# by another event.)
#

# Uncomment out the following line to turn the de-duplication off
#DEDUPLICATION=OFF

# OV_IF_Unknown
<.1.3.6.1.4.1.11.2.17.1.0.40000011, $r>
# OV_IF_Down
<.1.3.6.1.4.1.11.2.17.1.0.58916867, $r>
```

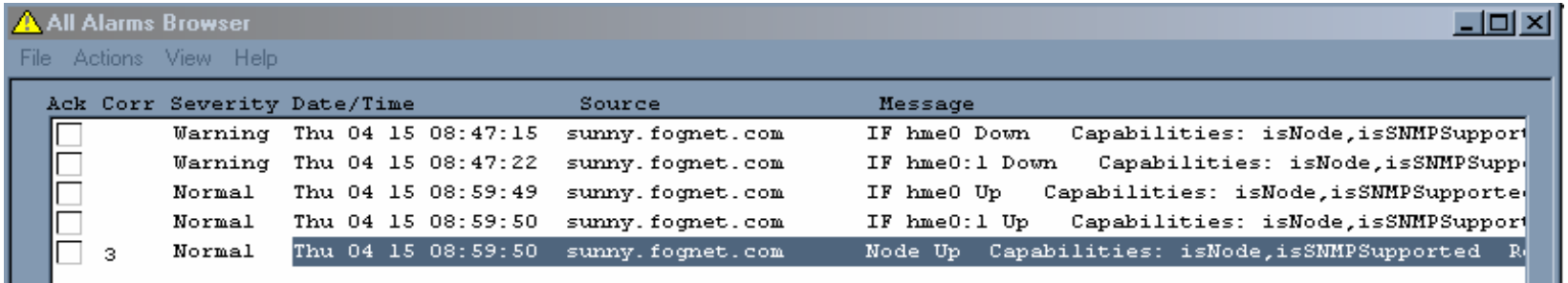
De-Dup correlation example

- Example 1: Multiple Interfaces Down, De-Dup on
  - If_Down events embedded from same source under IF *first* detected down
  - Alarms deleted when If_Up event received, *even after PairWise time window*
- Example 2: De-Dup off
  - If events separate & Pairwise deletes If events before 10 minutes, but *not* after
  - Note "Up" alarms logged here; If alarms correlated to Node by NodeIf

- 1:



- 2:



De-Dup correlation configuration – dedup.conf

- APA events affected by dedup
  - Note dedup keyed to source
  - Note APA source varbind "floats"

```
# Advanced Poller/Analyzer (APA) events
# OV APA down events
<.1.3.6.1.4.1.11.2.17.1.0.58983011, $r, $14>
<.1.3.6.1.4.1.11.2.17.1.0.58983012, $r, $7>
<.1.3.6.1.4.1.11.2.17.1.0.58983013, $r, $4>
<.1.3.6.1.4.1.11.2.17.1.0.58983014, $r, $7>
<.1.3.6.1.4.1.11.2.17.1.0.58983035, $r, $10>
<.1.3.6.1.4.1.11.2.17.1.0.58983037, $r, $10>
<.1.3.6.1.4.1.11.2.17.1.0.58983006, $r, $7>
<.1.3.6.1.4.1.11.2.17.1.0.58983007, $r, $7>
<.1.3.6.1.4.1.11.2.17.1.0.58983010, $r, $7>
<.1.3.6.1.4.1.11.2.17.1.0.58983025, $r, $7>
<.1.3.6.1.4.1.11.2.17.1.0.58983026, $r, $7>
<.1.3.6.1.4.1.11.2.17.1.0.58983027, $r, $4>
<.1.3.6.1.4.1.11.2.17.1.0.58983028, $r, $7>
# OV APA up events
<.1.3.6.1.4.1.11.2.17.1.0.58983001, $r, $14>
<.1.3.6.1.4.1.11.2.17.1.0.58983002, $r, $7>
<.1.3.6.1.4.1.11.2.17.1.0.58983003, $r, $4>
<.1.3.6.1.4.1.11.2.17.1.0.58983004, $r, $7>
<.1.3.6.1.4.1.11.2.17.1.0.58983034, $r, $10>
<.1.3.6.1.4.1.11.2.17.1.0.58983005, $r, $7>
<.1.3.6.1.4.1.11.2.17.1.0.58983009, $r, $7>
```

```
# OV APA unreachable events
<.1.3.6.1.4.1.11.2.17.1.0.58983021, $r, $14>
<.1.3.6.1.4.1.11.2.17.1.0.58983022, $r, $7>
<.1.3.6.1.4.1.11.2.17.1.0.58983023, $r, $4>
<.1.3.6.1.4.1.11.2.17.1.0.58983024, $r, $7>
<.1.3.6.1.4.1.11.2.17.1.0.58983036, $r, $10>
<.1.3.6.1.4.1.11.2.17.1.0.58983008, $r, $7>
<.1.3.6.1.4.1.11.2.17.1.0.58983020, $r, $7>
# OV APA intermittent events
<.1.3.6.1.4.1.11.2.17.1.0.58983015, $r, $7>
<.1.3.6.1.4.1.11.2.17.1.0.58983016, $r, $14>
<.1.3.6.1.4.1.11.2.17.1.0.58983017, $r, $7>
<.1.3.6.1.4.1.11.2.17.1.0.58983018, $r, $4>
#       (Link Intermittent - hostname & ifIndex)
<.1.3.6.1.4.1.11.2.17.1.0.58983019, $r, $1, $2>
# OV HSRP events (only check virtual IP address)
<.1.3.6.1.4.1.11.2.17.1.0.60001409, $3>
<.1.3.6.1.4.1.11.2.17.1.0.60001410, $3>
<.1.3.6.1.4.1.11.2.17.1.0.60001411, $3>
<.1.3.6.1.4.1.11.2.17.1.0.60001412, $3>
<.1.3.6.1.4.1.11.2.17.1.0.60001416, $3>
# OV APA detection of IF Renumbering
<.1.3.6.1.4.1.11.2.17.1.0.58983033, $r, $6>
<.1.3.6.1.4.1.11.2.17.1.0.58983040, $r, $6>
```

RepeatedEvent Correlation

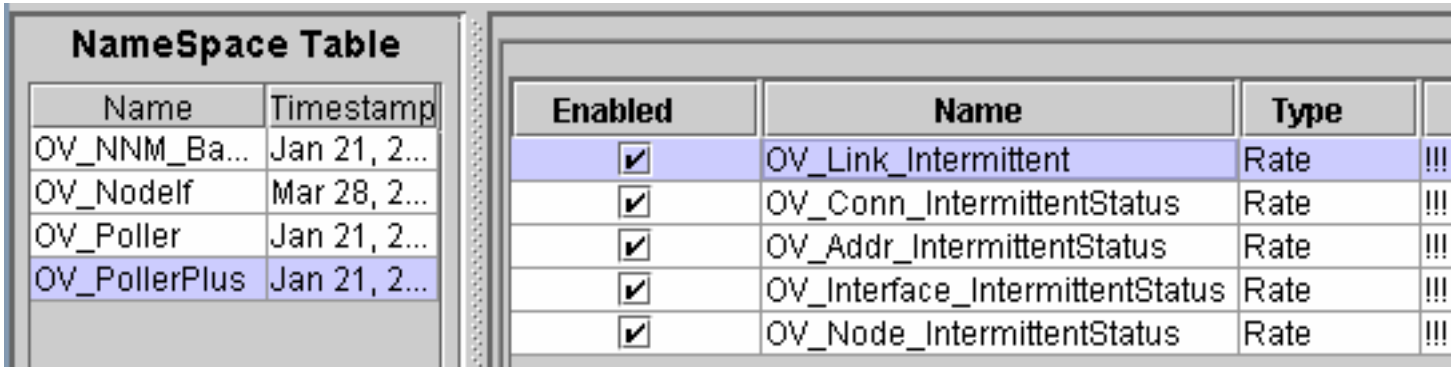
- Repeated event correlation becomes a “legacy” correlation with De-dup
- Embeds subsequent matches under original event in alarm browser
- Only indication to users is incrementing of correlated message count
- Default Time Window = 10 minutes
- Repeated event default correlations affecting status:
  - OV_Interface_Unknown
  - OV_Interface_Down
  - OV_Node_Up in V6.x, disabled in NNM 7.x
- Several other events still configured to use Repeated Event:
  - OV_IF_Intermittent
  - OV_BadSubnetMask
  - OV_PhysAddrMismatch

IntermittentStatus Correlation

- IntermittentStatus correlation behavior
  - Detects flapping interfaces/nodes otherwise "hidden" by PairWise
  - AKA "Router/Switch Intermittent Status," in NNM Docs and "OV_Connector_IntermittentStatus" in correlation composer
  - Applies only to connector interfaces polled by *netmon*
  - Irrelevant when using APA Polling
  - New Alarm in V6.31:
    - OV_IF_Intermittent – OpenView enterprise 58982423
  - RATE_COUNT
    - Default is 4 in V6.31, 5 in V6.4
    - Default is 4 in V7.0, 7.01, 7.5
    - RATE_PERIOD Default is 30 minutes
  - Configure via Correlation Composer OV_NNM_Basic namespace
  - netmon.lrf: –k shortPollTime=120; netmon.lrf:-k shortPollDownCount=2

OV_PollerPlus correlations

- Behavior:
  - Same function as IntermittentStatus for APA status events affected by PairWise
  - Contributed, unsupported. NOT enabled by default, must be manually loaded
  - Four individual circuits for APA connection, interface, address, node events
  - Fifth addititional circuit which is for Link Down traps (Generic 2)
- Status Alarms:
  - Four Alarms in V7.x: OV_APA_[INTERFACE|NODE|ADDR|CONN]_Intermittent
  - RATE_COUNT
    - Default is 2
    - RATE_PERIOD Default is 30 minutes
- Configuration:
  - See note in APA white paper for configuration instructions



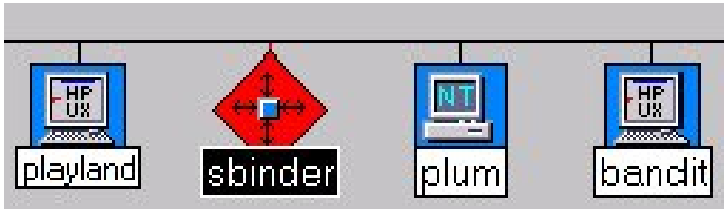
ConnectorDown correlation behavior

- General behavior:
  - NNM's "first" built-in correlation, introduced in NNM V6.0
  - Circuit embeds interface-related events under node-related events
  - Circuit reads path data encoded in varbinds to distinguish primary/secondary
  - Circuit embeds secondary failures under primary failures (*netmon*-only)
  - Important Node filter defines list of nodes always considered primary (*netmon*)
    - *genannosrvr* feeds important node filter data to ECS
- *netmon*-based status:
  - *netmon* builds path data in memory on startup to determine primary/secondary
  - Topology status set to "unknown" for subsequent secondary failures
  - Scheduled polls to "downstream" secondary interfaces doubled
- *ovet_poll*-based status:
  - APA failure analysis done on adjacent nodes to correlate connector state
  - *ovet_pathengine* passes primary/secondary path analysis data to *netmon*
  - By default, ET never passes secondary device status to topology/alarm browser

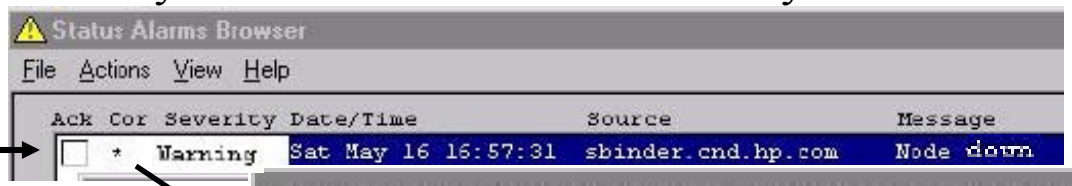
ConnectorDown correlation configuration

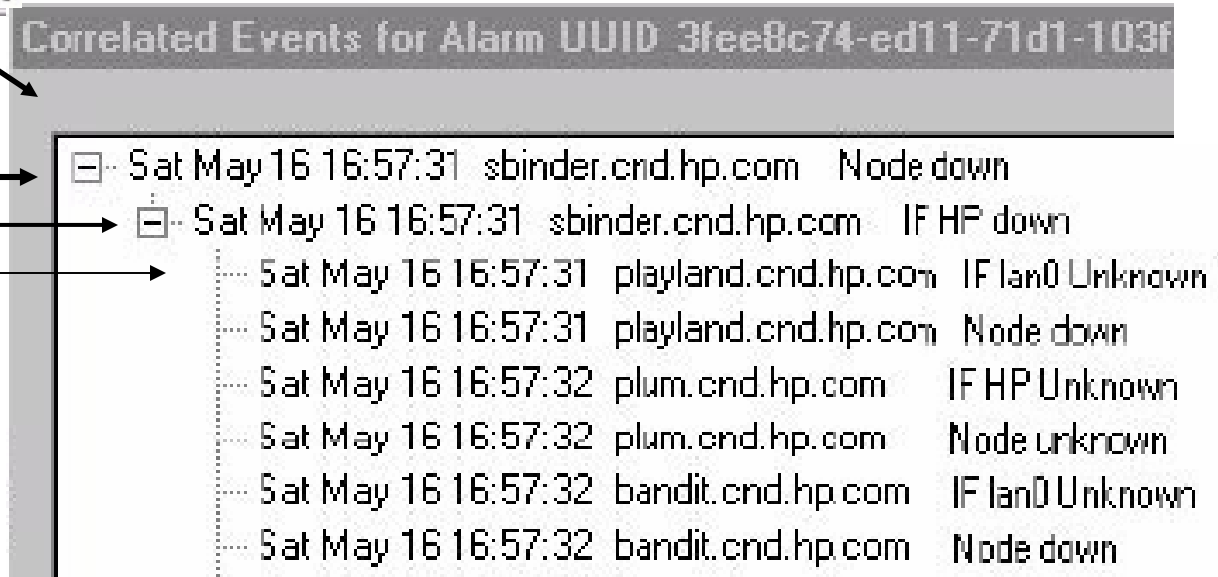
- Status Alarms:
  - Interface events embedded into node events for *netmon*-based status
  - Interface/Address/Connection/Node events embedded for APA-based status
  - Secondary alarms from "downstream" nodes embedded (*netmon*-based polling)
- Topology Status:
  - Primary Interfaces updated immediately
  - Secondary failures set to selected failure status when polled
    - Down, Unknown (default), Unchanged (*netmon* via xnmpolling)
    - Off, ET, Critical, Unknown (APA via the following StatusBridge settings:
      - PrimaryStatusOnly (false); SecondaryFailureStatus (ET); CorrelateSecondaryFailures (true)
  - Connector Node topology status other than Up/Down held 4 minutes
- Configuration:
  - Correlation Manager
  - Network Polling Configuration:
- Scope: Most if not all netmon and APA-based status events

APA/*netmon* ConnectorDown correlation differences

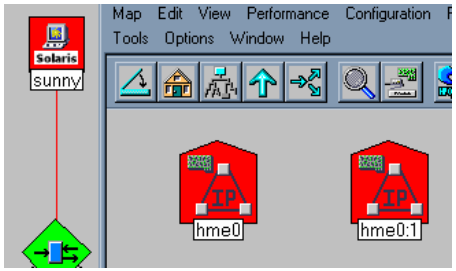


- ConnectorDown with *netmon* polling:
  - Note "triple" layers of embedding
    - Primary Parent Event – Node Down
    - Primary Child Event -- Interface Down
    - Secondary Grandchild Event – Secondary Nodes down or unknown

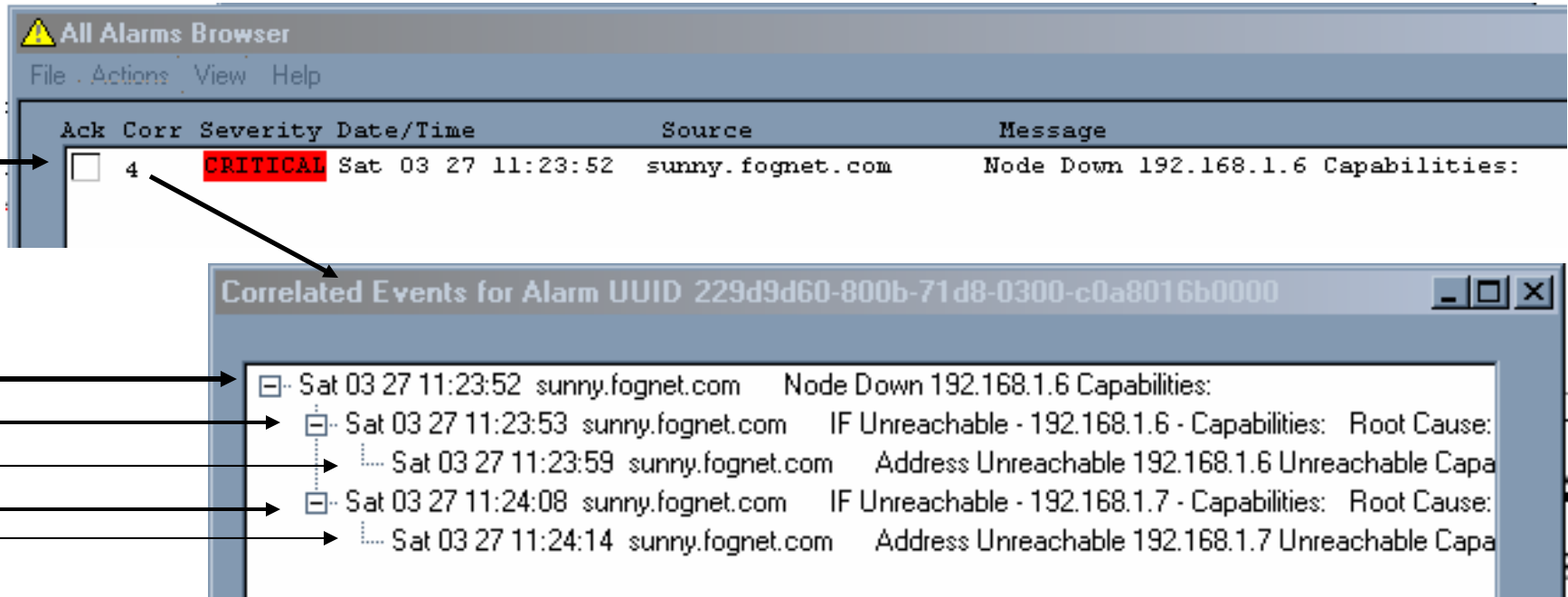




APA/*netmon* ConnectorDown correlation differences



- ConnectorDown with V7.01 using APA polling:
  - Note "triple" layers of embedding
    - Primary Parent Event – APA Node Down
    - Primary Child Events -- Interface unreachable
    - Secondary Grandchild Events – Address unreachable
  - Secondary failures never indicated by default



NodeIf correlation behavior

Behavior:

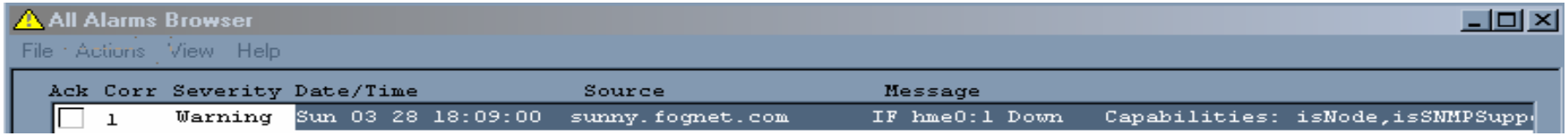
- AKA "Router/Switch Health"
- Supplements ConnectorDown after major re-work of *netmon* status alarms V6.31
- Takes advantage of V6.31 *netmon* dynamic polls to connector interfaces
- NodeIf correlates *netmon*-based status of interfaces on the same device
- Suppresses interface status alarms from non-connector devices (e.g. systems)
- Suppresses interface alarms from unconnected ports

- Status Alarms:
  - Simple device: send interface events immediately, always suppress node events
  - Node alarms for simple devices *not* suppressed when using APA
  - Connector: hold interface event from alarm browser until either:
    - Major node status event occurs (all if's down, all up, all unknown), or
    - PairedTimeWindow (10 Minutes).
- Topology Status:
  - Interface & Node status released immediately
- Configuration:
  - Correlation Composer, OV_NodeIf namespace
  - netmon.lrf: –k scheduleChassisIfsImmediate=false

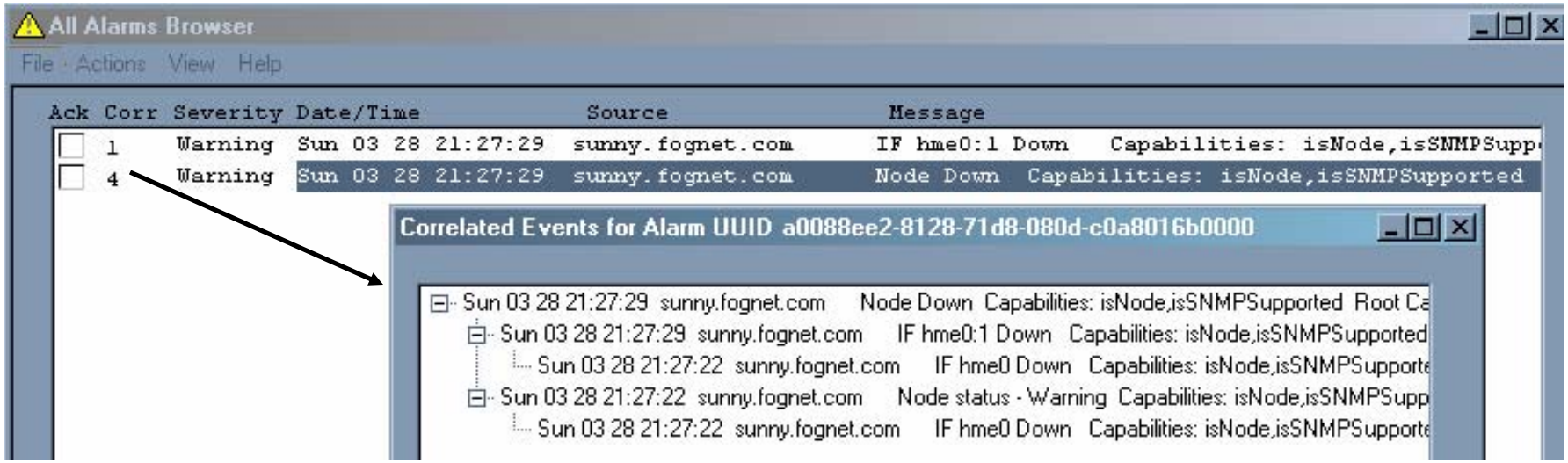
NodeIf correlation behavior – example

- Example 1: NNM V7.x, *netmon*-based status, NodeIf on (default)
  - Node Down never seen for simple devices – only Interface level events
- Example 2: : NNM V7.x, *netmon*-based status, NodeIf disabled
  - Node event and embedded ConnectorDown correlations deleted for simple device
  - For connector, IF events combined into single IF event (or node event if all down)
  - Same behavior for NNM V6.31, V6.41, V7.0

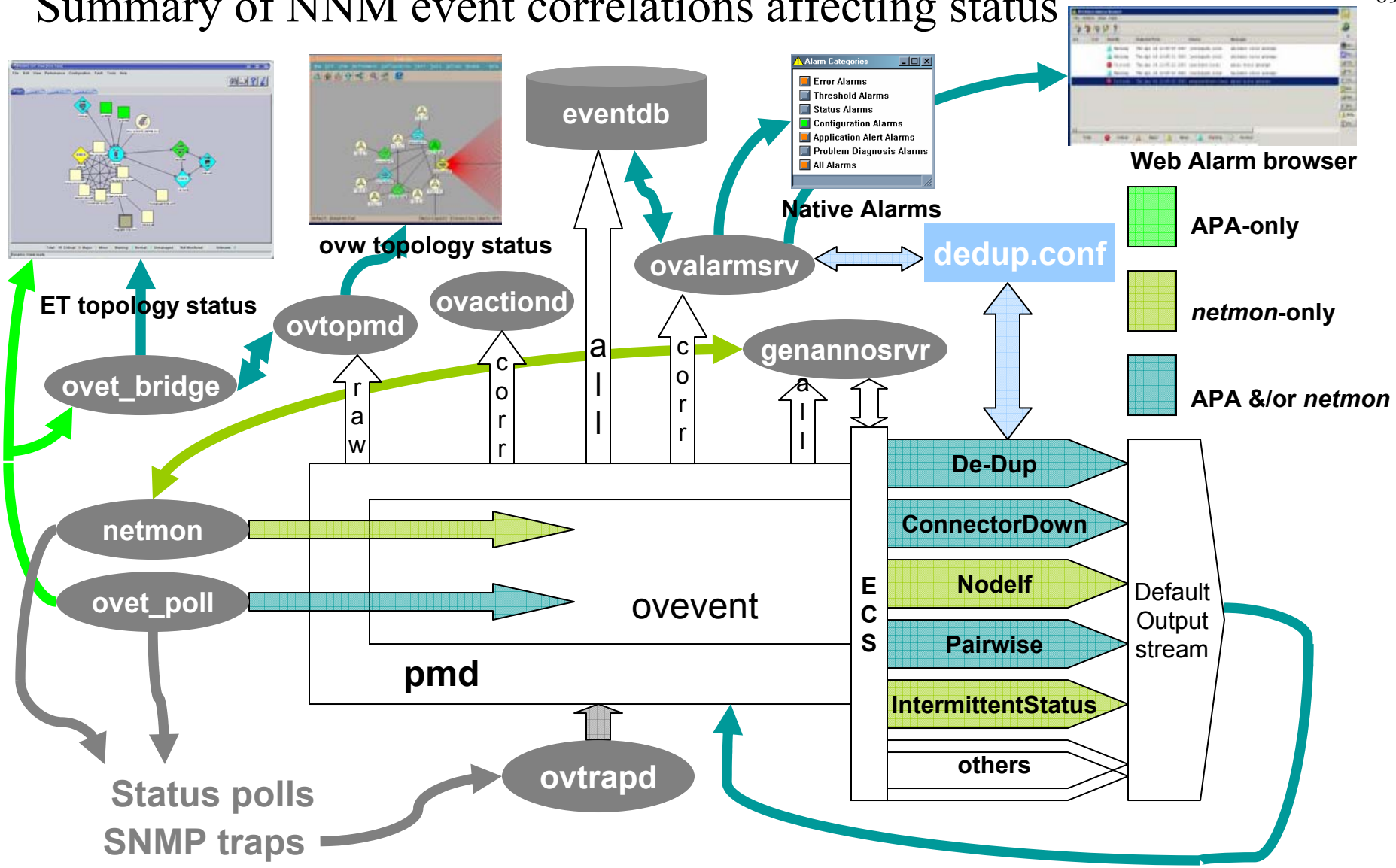
1:



2:



Summary of NNM event correlations affecting status



Characterizing status behaviors

- *netmon* logging
  - Runtime statistics:
    - `ovstatus -v netmon`
  - Object-based data:
    - `netmon -n <target>` dumps verbose information about the target node to $OV_LOG/netmon.trace, including the critical path.
    - `netmon -i <IP>` dumps verbose information about the target Interface to $OV_LOG/netmon.trace including the critical path.

- Stack tracing via pmdmgr
  - To turn on the tracing of the OV_EVENT stack:
    - `pmdmgr -SOV_EVENT\;T0xffffffff`
  - This should produce entries in $OV_LOG/pmd.trc0
  - To turn on pmd debugging to trace all stacks (SNMP, etc):
    - `pmdmgr -D0xffffffff`
  - Turn tracing off, simply restart OV, or run:
    - `pmdmgr -SOV_EVENT\;T0x0`

Characterizing status behaviors

- *netmon* tracing
  - `netmon -M <Mask>` sends an event to the running *netmon* to turn on tracing.
  - Trace output is written $OV_LOG/netmon.trace.
  - Run `netmon -M 0` to turn off tracing
  - Tracemasks are listed in the *netmon* man/ref page
  - netmon.trace grows without bounds and can easily fill up a disk
  - Tracemasks are additive, so to trace both SNMP requests and SNMP replies and timeouts, use a tracemask of 12 (4+8).

- *netmon* stack dumping
  - `netmon -a <action-number>` dumps netmon internal data structures s to $OV_LOG/netmon.trace. Use netmon –a ? to list action-numbers
  - To dump ping list: netmon –a 12
  - To dump SNMP list: netmon –a 16

Characterizing status behaviors

- APA logfiles
  - UNIX:
    - /var/opt/OV/log/ovet_poll.log.txt
    - /var/opt/OV/log/ovet_poll_err.log
  - Windows:
    - <install_dir>data/log/ovet_poll.log.bin
      - Use <install_dir>\bin\ovlogdump.exe to view log
    - <install_dir>data/log/ovet_poll_err.log

- APA Tracing
  - Mostly XPL based tracing (See XPL tracing guide)
  - See ovtrcadm, ovtrccfg, and ovtrcgui (windows only) in $OV_SUPPORT
  - Also Some limited APA tracing via log-only events (e.g., CFA):
    - See `ovdumpevents` command

Characterizing status behaviors

•ECS Logging

- Logging All Incoming ECS Events
  - To turn on logging `ecsmgr -log_events input on`
  - To turn off logging: `ecsmgr -log_events input off`
  - To change log size: `ecsmgr -max_log_size event <Kbytes>`
  - The log file name is *ecsin.evt0*, is a rolling log, is 512k by default and is 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 ecsin.evt1

•Logging Output and Correlated Events

- `ecsmgr -log_events stream on`
- `ecsmgr -log_events stream off`
- `ecsmgr -max_log_size event <Kbytes>`

•The log file is default_xxx.evt0, is a rolling log, 512K by default, and is located in:

- UNIX: $OV_LOG/ecs/1/default_sout.evt0 & $OV_LOG/ecs/1/default_sout.evt1
- Windows:<install_dir>\log\ecs\1\default_sout.evt0 & default_sout.evt1

•Logging discarded or suppressed events:

- UNIX: $OV_LOG/ecs/1/default_sdis.evt0 & $OV_LOG/ecs/1/default_sdis.evt1
- Windows:<install_dir>\log\ecs\1\default_sdis.evt0 & default_sdis.evt1

Characterizing status behaviors

•ECS Tracing and Simulation

- Simulate events for testing ECS logic
  - •*ecsevgen* and *ecsevout* in $OV_CONTRIB/ECS can be used to replay logs for testing the effects of ECS circuit changes. For more info, see TroubleshootingEventCorrelation.txt in that directory.

- ECS Support tools
  - $OV_SUPPORT/processEvents – Summarizes output of ovdumpevents (UNIX Only)
  - $OV_SUPPORT/processCorrEvents – Summarizes output of correlation log (UNIX Only)
  - $OV_SUPPORT/ECSTracing.ovpl – Front-end to pmdmgr & ecsmgr tracing/logging commands

•ECS Tracing

- •ECS tracing can be extremely verbose, but certain development activities can't be sufficiently troubleshot with logging. To enable full ECS tracing:
  - •`ecsmgr -i 1 -trace 65536`
  - •`pmdmgr -SECSS\;T0xffffffff`
- •The traces are then written to $OV_LOG /pmd.trc0.

Characterizing status behaviors

•ECS Tracing and Simulation

- Simulate events for testing ECS logic
  - •*ecsevgen* and *ecsevout* in $OV_CONTRIB/ECS can be used to replay logs for testing the effects of ECS circuit changes. For more info, see TroubleshootingEventCorrelation.txt in that directory.

•ECS Tracing

- •ECS tracing can be extremely verbose, but certain development activities can't be sufficiently troubleshot with logging. To enable full ECS tracing:
  - •`ecsmgr -i 1 -trace 65536`
  - •`pmdmgr -SECSS\;T0xffffffff`
- •The traces are then written to $OV_LOG /pmd.trc0.

Questions?

Important! Please complete the session survey

Thank You

The Rocky Road to Advanced Status in NNM

Session #: 282

Prepared by

Mike Peckar
Fognet Consulting

www.fognet.com