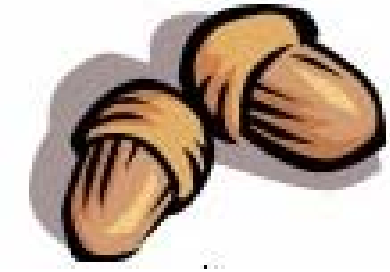
Session Number 80

NNM Status Polling – Soup to Nuts

Tuesday, June 15th, 2004, 4:15 pm

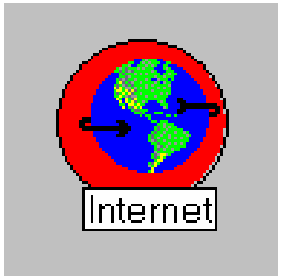
Mike Peckar

Fognet Consulting



Welcome! This session will focus on:

- NNM status polling configuration entry points
- Old poller vs. new poller and how to switch
- Default status alarm behaviors (both pollers)
- Event Correlation affect on status



netmon

APA

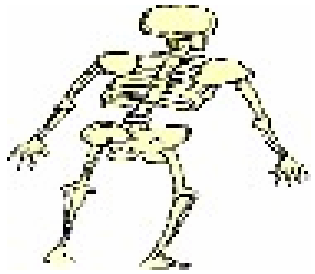
Agenda

- Status event overview and architecture
- *netmon* vs. APA
- APA configuration and polling defaults
- *netmon* configuration defaults
- Event correlations affecting status alarms
  - PairWise
  - De-Dup
  - Repeated
  - IntermittentStatus
  - OV_PollerPlus
  - ConnectorDown
  - NodeIF
- Wrap up

NNM status subsystems in a nutshell

- *netmon* performs discovery and status polling (old poller)
  - 4 areas of polling: Discovery, Configuration, Status, Path
  - ICMP polls issued to ifs with IP addresses *or*
  - SNMP status polls issued to nodes to poll unnumbered ifs
  - Multiple configuration entry points; dynamic reconfiguration
- *Active Problem Analyzer* in 7.01+; *ovet_poll* (new poller)
  - Used for HSRP & OAD by default for ET-discovered devices
  - Can take over status polling for *netmon*, but not all discovery
  - ICMP *and* SNMP polls; layer two-based status, path analysis
- *Event Correlation* (ECS embedded runtime; correlation composer)
  - Increasing affect on status alarms versions 6.0 forward
  - Connector Down for cascade failure
  - Relational DB for alarms == relational alarms display
  - 6.31+: NodeIf, De-Dup added; Pairwise & if events changed

Why is status so complex??? Because dead men tell no tales…



NNM status events - *netmon*

- *netmon*-generated status events: `OV_IF_Down (58916867)`
  - ICMP polls to IP-addressed interfaces. Level 2 polls via SNMP
  - V3.31 –V6.2: Status events "Node centric" - *all IF events "log-only"*
  - V6.31 and after: Status events "Interface-centric"
    - Heavily correlated by ECS, e.g. NodeIF, ConnectorDown, Pairwise, De-dup

• All ode status alarms derived from IF status		**(V3.31-V6.2)**	**V6.31+**
OV_Node_Warning:	One interface down; others up or unknown	**L**	**LO**
OV_Node_Marginal:	>One interface down; others up or unknown	**L**	**LO**
OV_Node_Major:	One interface up	**L**	**LO**
OV_Node_Down:	All interfaces down or unknown.	**L***	**L*+**
OV_Node_Unknown:	All interfaces on the node are unknown	**L***	**L***
OV_Node_Up:	All interfaces up	**LO**	**LO**
OV_IF_Down:	Interface Down	**LO**	**L*+**
OV_IF_Unknown:	Interface Unknown	**LO**	**L***
OV_IF_Up:	Interface Up	**LO**	**LO**

*=Correlated by ConnectorDown *+=Correlated by ConnectorDown and NodeIf

NNM status events - APA

- APA-generated status events: `OV_APA_IF_DOWN (58983012)`
  - ICMP and/or SNMP polls related to address, interface, Node, Connection
  - Polling granularity defined by ET Topology filters (defaults below)
  - Correlated internally by Extended Topology and less heavily by ECS

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

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

NNM topology alarms and topology status

- Major change in NNM V6.31 to topology status alarm logging behaviors
- No changes since NNM V3.31 in default map status propagation rules

	6.31-	6.31+
OV_Segment_: Marginal, Normal, Warning, Unknown	**LO**	**LO**
OV_Segment_Major:	**L**	**LO**
OV_Segment_Critical:	**L**	**LO**
OV_Network_: Marginal, Normal, Warning, Unknown	**LO**	**LO**
OV_Network_Major:	**L**	**LO**
OV_Network_Critical:	**L**	**LO**

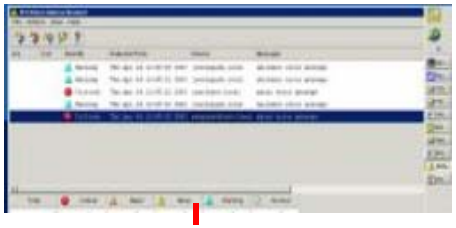
- Other events affecting topology status:

Connection_* (all **LO**),
Station_*
Remote_Mgr_*
IPV6_* (V6.4+)
HSRP_* (V6.4+)

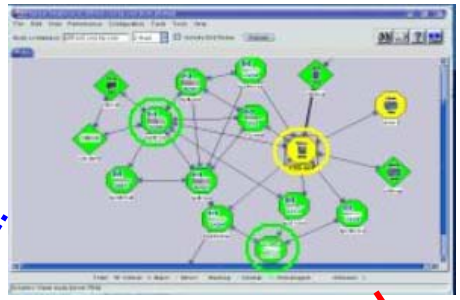
Default map status propagation:

Unknown	No normal or abnormal symbols.
Normal	All symbols are normal.
Warning	One symbol is abnormal; all others are normal.
Minor	Multiple symbols are abnormal; multiple symbols are normal.
Major	One symbol is normal; all other symbols are abnormal.
Critical	All symbols are abnormal.

Status Architecture



Alarm Browser



Dynamic Views

ovw

NNM

ET

ovalarmsrv

BES

Extended Topology

ovet_bridge

hosts.nnm

NNM Topology

NNM Object

composer/manager

ovet_pathengine

ECS

pmd

important nodes filter

baConfig.xml

trapd.conf

xnmpolling

xnmsnmpconf

genannosrvr

ovet_poll

netmon

ovtrapd

ovet_disco

Topology/Discovery

Status

Path data

ET functions

NNM functions

Shared functions

Fault Analysis

Status polls

L2 Discovery

L3 Discovery

L3 Configuration polls

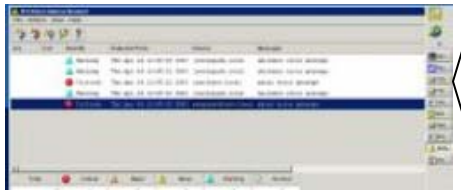
Status polls

SNMP traps

L2 Discovery

L2 Configuration

NNM correlations affecting status alarms



Alarm browsers

eventdb

Binary Event Store

ovalarmsrv

dedup.conf

APA & *netmon*

netmon-only

topology status

ovactiond

genannosrvr

ovtopmd

raw

corr

corr

all

De-Dup

pmd

ConnectorDown

netmon

NodeIf

ECS

Default Output stream

ovet_poll

ovevent

Pairwise

IntermittentStatus

status polls

ovtrapd

others

SNMP traps

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 for a particular device
  - 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
  - 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

- *netmon* pros
  - Behavior characterizable and configurable, less FUD for installed base
  - Due to its simplicity and lack of in-depth analysis capabilities, still scales OK
  - GUI's available for polling customization and configuration
  - ovw-based fault tools keyed to *netmon*, e.g. "status poll"
  - Command line tools to query polling data, e.g. *nmdemandpoll, xnmsnmpconf*

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
  - Provides more dynamic polling based on status at different entity levels
  - Provides connection-oriented and device-oriented status
  - Defaults generally provide more accurate & timely status than *netmon* defaults

- APA Cons

  - Analysis engine is complex and difficult (impossible??) to interpret behaviors
  - Status algorithms not exposed, so polling behaviors not 100% configurable
  - Intervals configured in SNMP Configuration GUI ignored by APA (FUD)
  - Polling customization requires modifying XML file; no GUI
  - ovw-based fault tools not available, i.e. nmdemandpoll
  - Status polls can't be forced to update known "up" node reporting "down"
  - 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 polling not available after switching to APA

Switching status polling control

- Read $OV_DOC/whitepapers/Active_Problem_Analyzer.pdf
- Exit GUI sessions
- Run `setupExtTopo.ovpl`, then `etrestart.ovpl`
- 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
    - *ovet_bridge* uses this file to designate what hosts are polled by APA
  - Restarts appropriate background processes
- Flookieness happens
  - Watch ovstatus carefully
    - ovstatus –v *netmon*
    - ovstatus –v *ovet_poll*

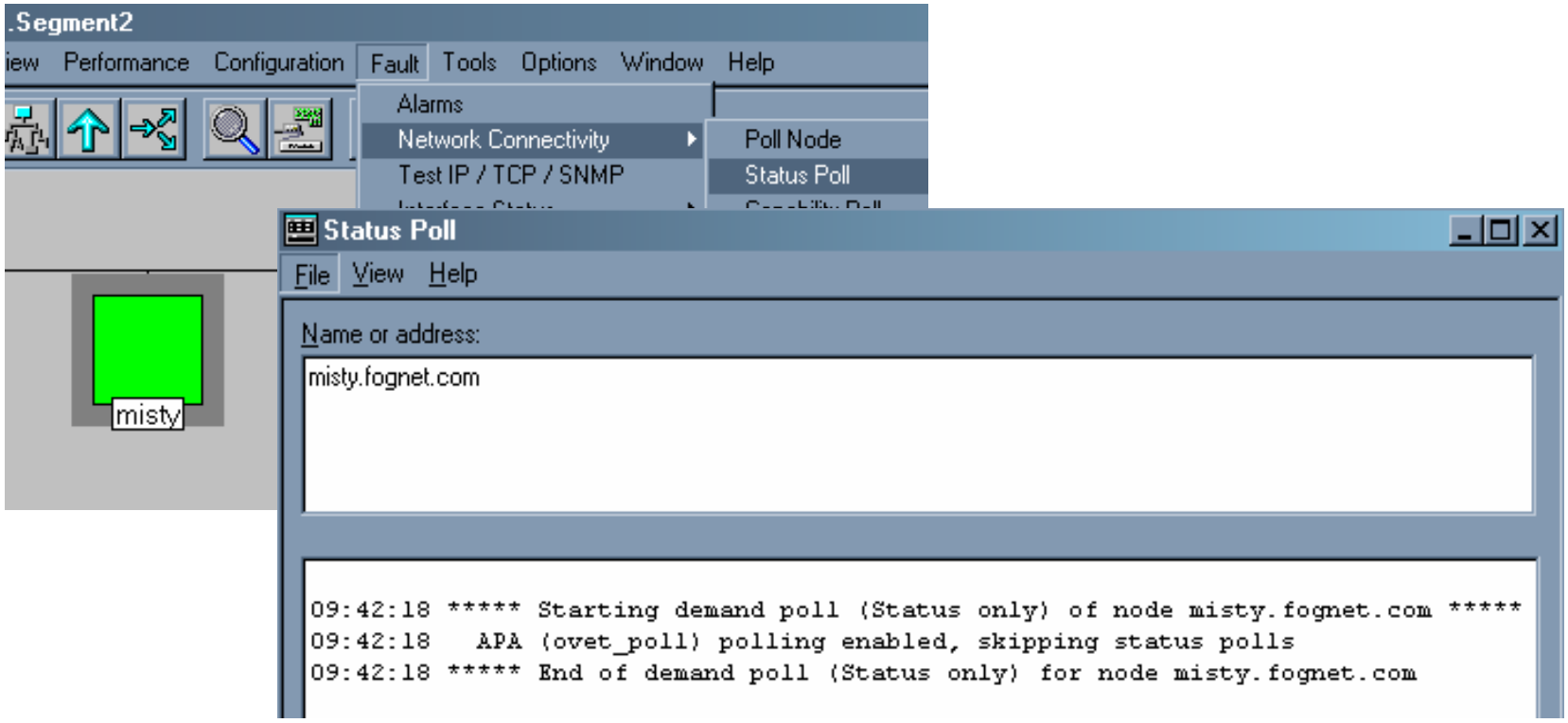
Determining poller control in V7.01

APA off:

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

APA on:

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



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,
        <generalParameters> cfaDebugLevel, GenerateDegradedEvent
        <classSpecificParameters>
          <defaultParameters> interval; snmpEnable ; timeout
          <classSpecification> isRouter, isSwitch, isEndNode
            <parameterList> interval; snmpEnable ; timeout
```

paConfig.xml polling settings – general & global

- SubSystemConfig: PollingEngine
- ConfigGroup: PollingSettings
- General parameters:
  - PrimaryFailureBackoffFactor (1); multiplier for primary failure mode
- Global parameters:
  - BasicPollingEnable (true)
  - PollNormalIP (false|true); Only OAD, HSRP entities polled by default
  - RecieveEvents (false); Receive link down/link up traps
  - HSRPPollingEnable (true)
  - StatisticsEnable (true)
  - StatisiticsInterval (300); seconds
  - ReportBusyObjectsAtStatisticsInterval (false); for tracing
  - ReportBusyObjectsInAlarmBowser (false); for tracing
  - ReportPollingResultsInAlarmBorwser (false); for tracing
  - PollingEngineThreadPoolSize (16); for performance tuning

BLUE indicates changes to paConfig.xml made by `ovet_apaConfig.ovpl -enable APAPolling`

paConfig.xml polling settings – class specific

- SubSystemConfig: PollingEngine
- ConfigGroup: PollingSettings
- ClassSpecific *default* parameters (those not matching ClassSpecifications):
  - Interval (300); snmpEnable (true); pingEnable (true); hsrpEnable (true)
- ClassSpecifications:
  - default; isRouter
    isSwitch
    isEndNode
    UnconnectedAdminUpRouterIf
    UnconnectedAdminUpSwitchIf
    UnconnectedEndNode
    NotConnectedIF

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

paConfig.xml polling settings – topology filters

- ClassSpecifications defined using extended topology 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]`
- ClassSpecification filters are evaluated in xml file order
  - device matching isSwitch *and* isRouter: isRouter rules apply
- Extended Topology Filters
  - Defined in $OV_CONF/nnmet/topology/filter/TopoFilters.xml
    - Similar filter definition logic to NNM filters, only in xml
    - Documentation?

paConfig.xml polling settings – polling engine

- 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

- SubSystemConfig: PollingEngine
- ConfigGroup: PingSettings
- ClassSpecific default parameters:
  - timeout (1000); milliseconds
  - numberOfRetries (2);

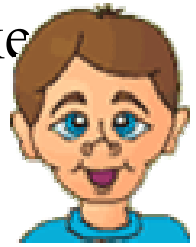
paConfig.xml polling settings – status analyzer

- SubSystemConfig: StatusAnalyzer
- GlobalParameters:
  - statusAnalyzerThreadPoolSize (10)
  - statusAnalyzerQueueSize (65000); input queue: repository of poll results
  - PAStatusAnalyzerDebugLevel (0); logs poller interaction w/ status analyzer (1-4)
  - PAStatusAnalyzerMasterDebugSwitchNode (NUL); specify node or addr to trace
  - PASendStatusAnalyzerSyncEvent (false); sends OV_PESA_Message events

paConfig.xml polling settings – fault analyzer

- ConfigGroup: ConnectivityFaultAnalyzer; Distinguishes primary/secondary failures
- General Parameters:
  - cfaEventFunctionTracingEnabled (false); generate function tracing events
  - cfaDebugLevel (0); quantity of debug events generated (0-4)
  - cfaDoCompositeRoute (false); enable traceroute & findactiveroute for prim/2ndry
  - cfaTraceRouteSeedPattern (NUL); seeded route for relaxing failure analysis path
  - cfaTraceRouteSeedValue (NUL); substitute this value if pattern above matched
  - cfaTraceRouteThroughFirewall (false); sets pathing algorithm when no return path
  - cfaStpConvergenceTimeSecs (50); time to delay polling for STP Convergence
- ClassSpecific default parameters:
  - analysisMaxNumberRetries (1); override snmp config timeouts during analysis
  - importantNodeUpToDown (false); do not suppress/embed 2ndary Node Downs
  - importantNodeDownToUp (false); do not suppress/embed 2ndary Node Ups
- ClassSpecifications:
  - isRouter, uses all default values (example)

paConfig.xml polling settings – HSRP and talke



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

- SubSystemConfig: Talker
  - The ovet_poll executable contains various kinds of talker modules which perform communication tasks external to the ovet_poll process
- ConfigGroup: SnmpTalker
- General Parameters:
  - snmpTalkerSessionCacheSize (32); # of SNMP sessions (uses up file descriptors)
  - NumberOfOIDsPerPDU (50); controls size of SNMP PDUs

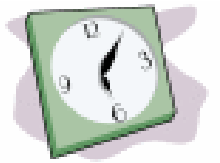
paConfig.xml polling settings – status bridge

- SubSystemConfig: StatusBridge
- Global Parameters:
  - StatusBridgeEnabled (false|true); NNM topology status is owned by ET
  - FullTopoSync (false); Full topology synchronization at initialization
- ConfigGroup: BridgeSettings
- Default Parameters:
  - DisableNNMPolling (true); sets OVETStatusMonitoring flag to true in NNM
  - 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 secondary failures. By default, the status will match whatever ET connectivity fault analysis determines the status to be. Allowable values are "ET", "Critical", and "Unknown". All other values will be treated as "ET". Note that the ET connectivity fault analysis may limit the scope and handling of secondary failures
  - CorrelateSecondaryFailures (true); By default, if a secondary 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. By setting CorrelateSecondaryFailures = false, then the status bridge will not attempt to correlate secondary failures, with the result that the status message will be a primary failure in NNM

APA polling settings – ET discovery & polling

- $OV_CONF/nnmet/DiscoSnmpHelperSchema.cfg
  - Settings here used by ovet_poll (APA) and ET Discovery

  - m_NumThreads (40); ET Query engines threads for SNMP
  - m_TimeOut (4000); 4 seconds
  - m_NumRetries (3); max is 7



Each successive timeout will double the previous time waiting for an SNMP response, up to a retry increase maximum time-out ceiling of 6 seconds. This means for a node that does not support SNMP it will consume a thread for 4+8+14+20 = 46 seconds and send 4 SNMP requests

  - m_Parallel (0); issue getCommunityString requests in parallel

- $OV_BIN/etrestart.ovpl) for changes to take effect.
- These settings override settings in Options – SNMP Configuration !!!
  - Those settings still used for *netmon*-based discovery, configuration polls

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

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 V7.0:
- NNM V7.01:

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
down	any	DISABLED
testing	any	TESTING
up	up	NORMAL
up	down	CRITICAL
up	testing	TESTING

– Note APA based SNMP status mapping unexposed

netmon layer 2 status polling

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

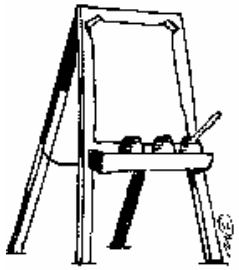
ifAdminStatus	ifOperStatus	OV Status
up	up	unset
down	down	unknown
testing	testing	normal, up
any	unknown	critical, down
	dormant	disabled
	notpresent	unmanaged
	lowerlayerdown	restricted
	any	testing

- $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)

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:



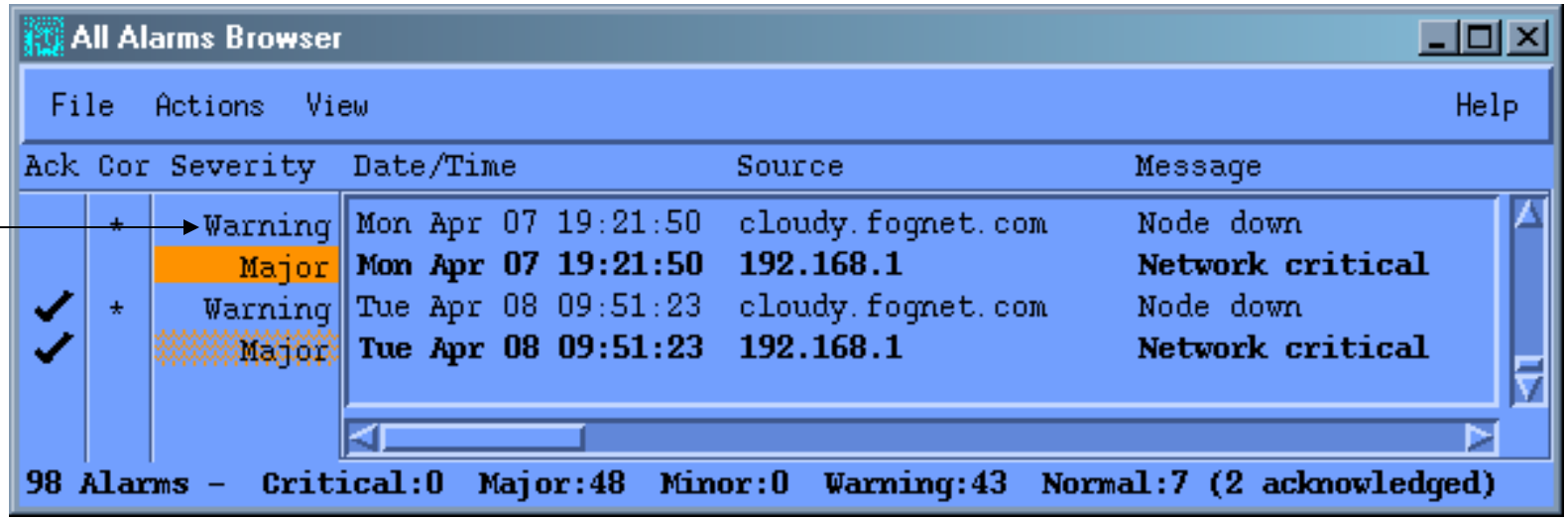
- Correlation Manager:
  - http://host.fqdn/OvDocs/C/ecs/ecscmg.html
  - First Event correlation GUI shipped with NNM's ECS Runtime
  - Available from Options-> Event Configuration -> Edit -> Event Correlation
- Correlation Composer:
  - A "Super Circuit" providing generalized "instances":
    - Suppress, 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

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:
    - 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 behavior – NNM V6.2

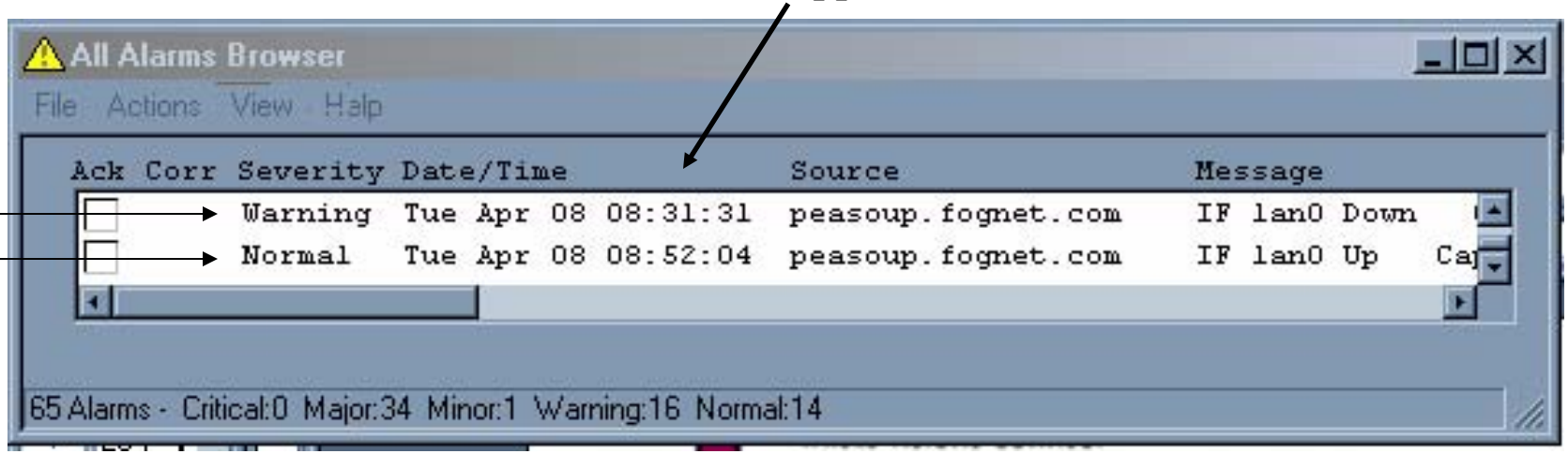
- Topology Status:
  - Status released immediately in all versions
- Configuration:
  - Correlation Manager
- Example: NNM V6.2
  - Node Down at 19:21; Event released immediately; window expires; no changes
  - Node Down at 19:51; Node Up within 10 min; message acknowledged



PairWise correlation behavior – NNM V6.41

- Example: NNM V6.41:
  - IF_Down at 08:31; event held; not seen in alarm browser
  - If set to critical immediately in topology
  - Alarms never seen in alarm browser if If_Up is received within 10 minutes
  - Window expires at 08:41; IF_down event released with *original timestamp*
  - IF_Up is "un-correlated" after time window (set to log in this example)

Note alarm 'appears' in browser at 08:41

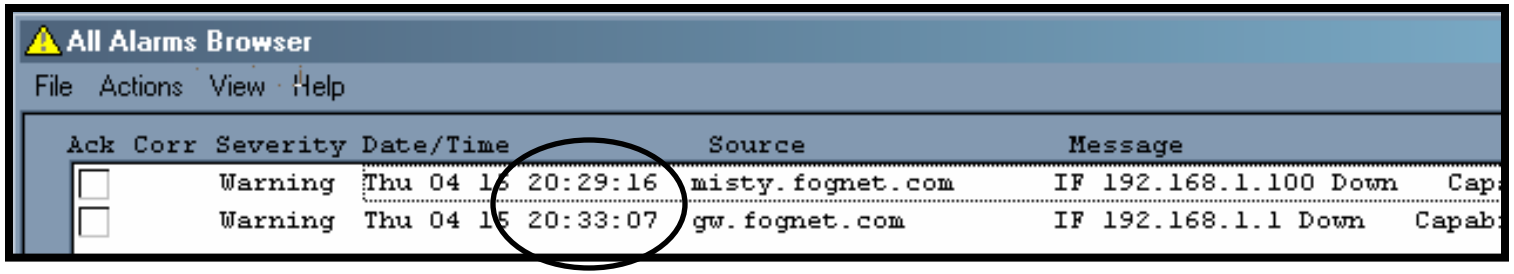


- Note no PairWise—DeDup relationships in this version

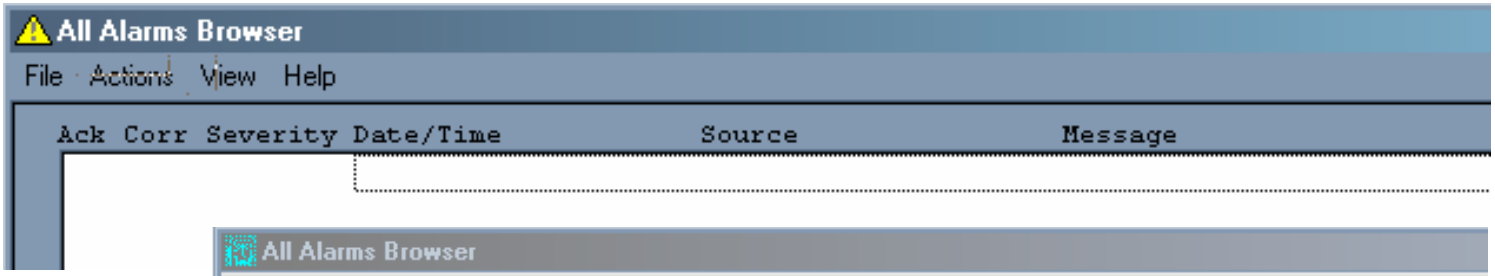
PairWise correlation example – NNM V7.01- *netmon*

- 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

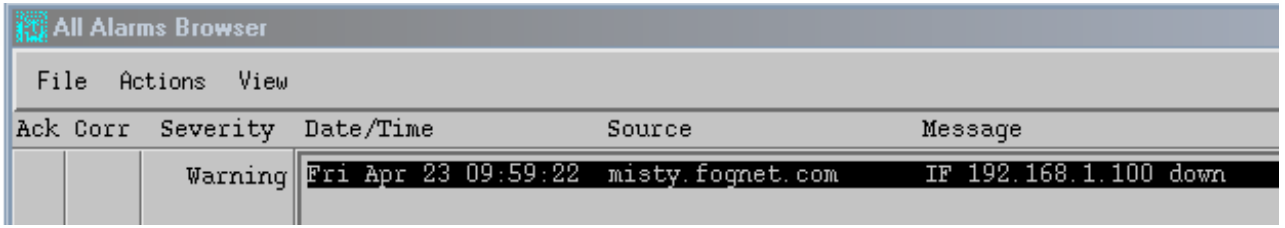
1:



2:



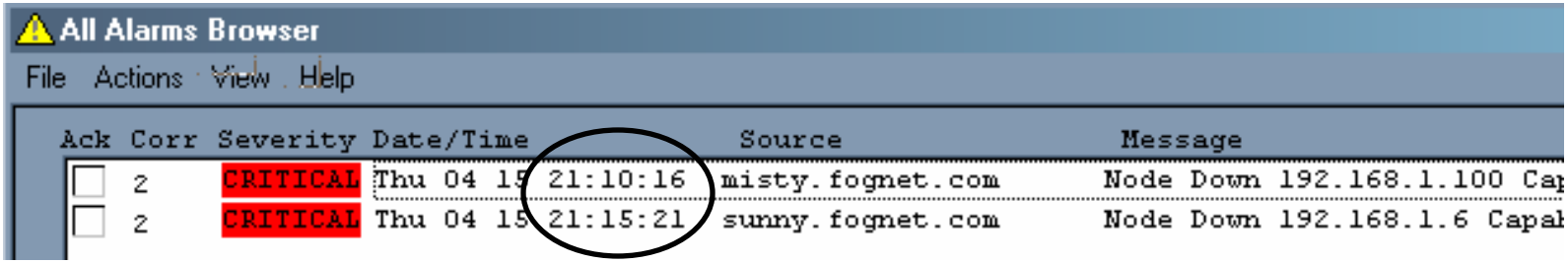
3:



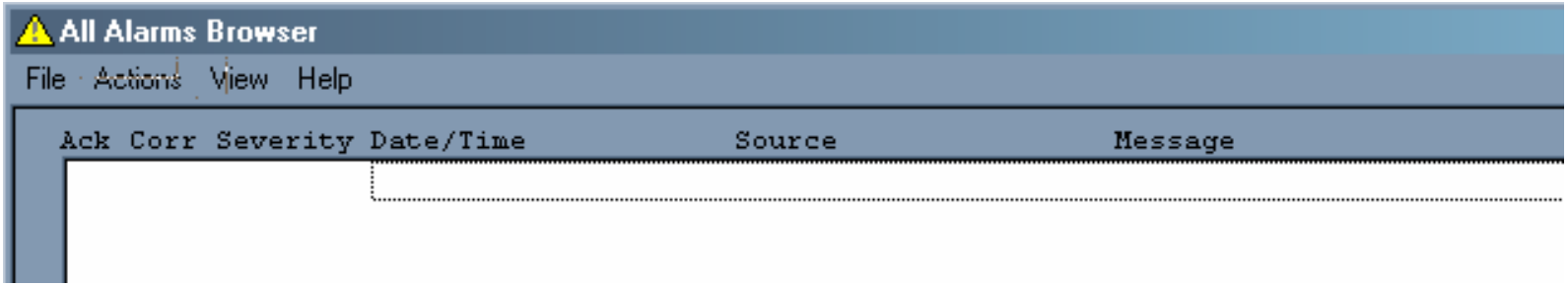
PairWise correlation example - NNM V7.01 - APA

- Example 1: NNM V7.01, APA Polling. IF events released immediately
  - Address; Interface unreachable correlated by ConnectorDown
- Example 2: Address Up events received for both at 20:24
  - Same deal

1:



2:

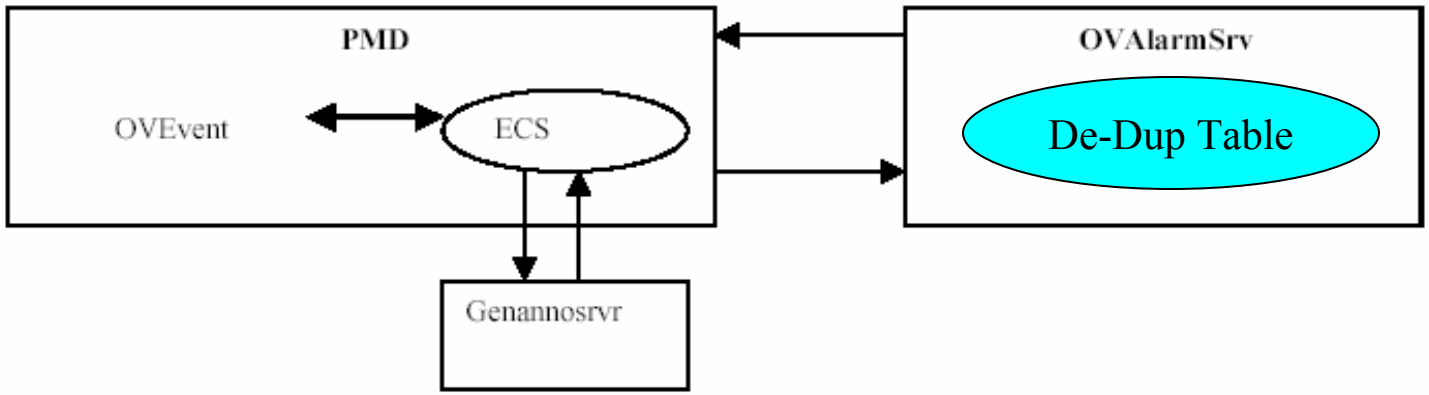


PairWise correlation - status events affected

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 (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!

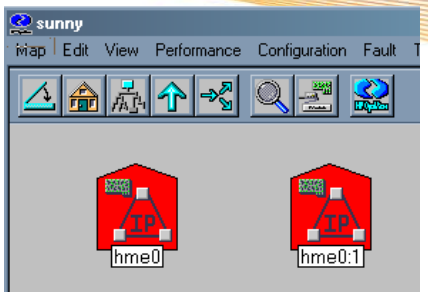


De-Dup correlation configuration – dedup.conf

- 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:

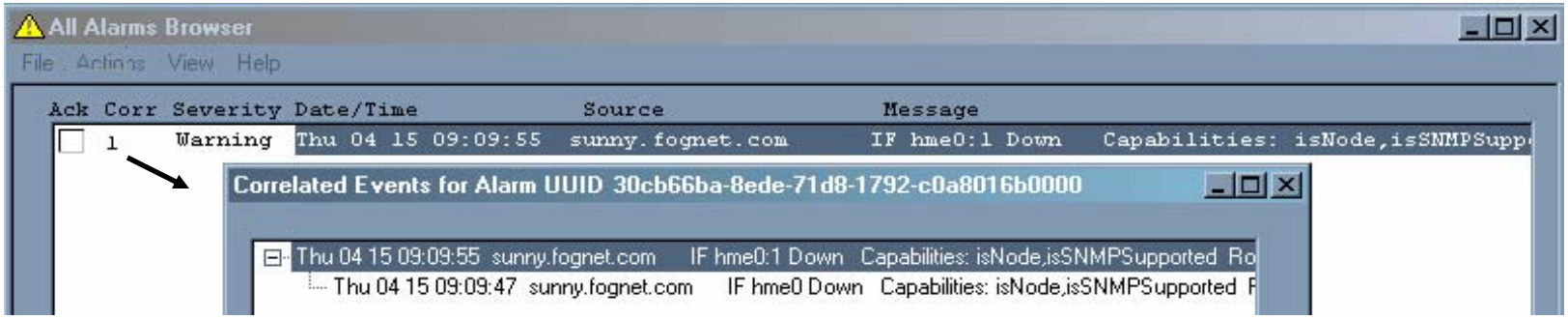
```
# Event De-Duplication Configuration file
# Format  <TrapOid[[, $r][,$NUM][,$*]]>
# Note:
# TrapOid is the oid that identifies the event to be de-duplicated
# $r is the event source
# $NUM is to specify the varbind number. 1<= NUM <=16
# $* is for all varbinds
#
# De-Dup Examples:
# <.1.3.6.1.4.1.11.2.17.1.0.59179225>
# <.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

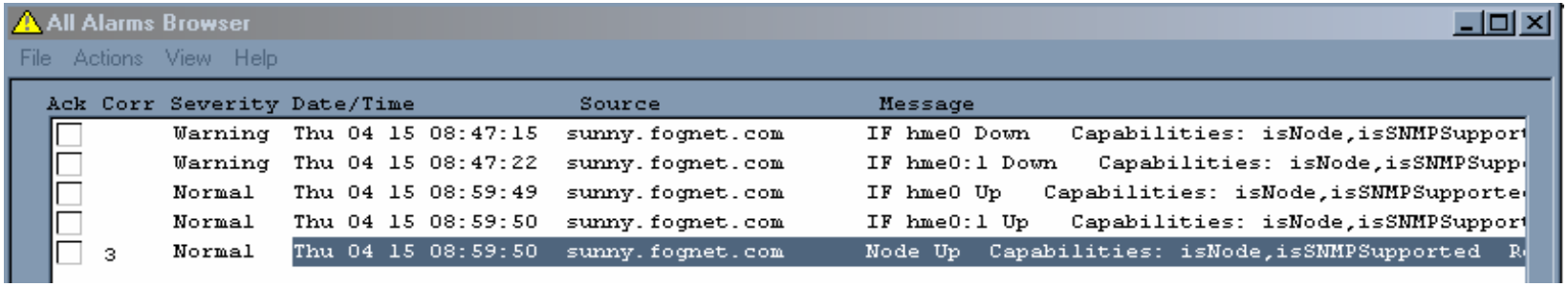


- Example 1: Multiple Interfaces Down, De-Dup on
  - If_Down events embedded from same source under IF *first* detected down
  - Alarms deleted when If_Up event received, *even after PairWise time window*
- Example 2: De-Dup off
  - If events separate & Pairwise deletes If events before 10 minutes, but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

- 1:



- 2:



De-Dup correlation status events affected

- Status events configured for de-dup by default in V6.41:
  - OV_IF_Down, OV_IF_Unknown, OV_IF_Intermittent
- Additional status events added for de-dup by default in V7.0:
  - OV_IF_Down, OV_IF_Unknown, OV_IF_Intermittent
  - OV_Bad_Subnet_Mask, OV_Duplicate_IP_address
  - OV_DuplicateIfAlias, OV_Node_Added
  - OV_Lic[All], RMON_Rise_Alarm
- Additional status events added for de-dup by default in V7.0:
  - OV_APA[ADDR|IF|NODE|CONNECTION]_Down
  - OV_APA[ADDR|IF|NODE|CONNECTION]_Up
  - OV_APA[ADDR|IF|NODE|CONNECTION]_Unreachable
  - OV_HSRP_[All]

RepeatedEvent correlation behavior

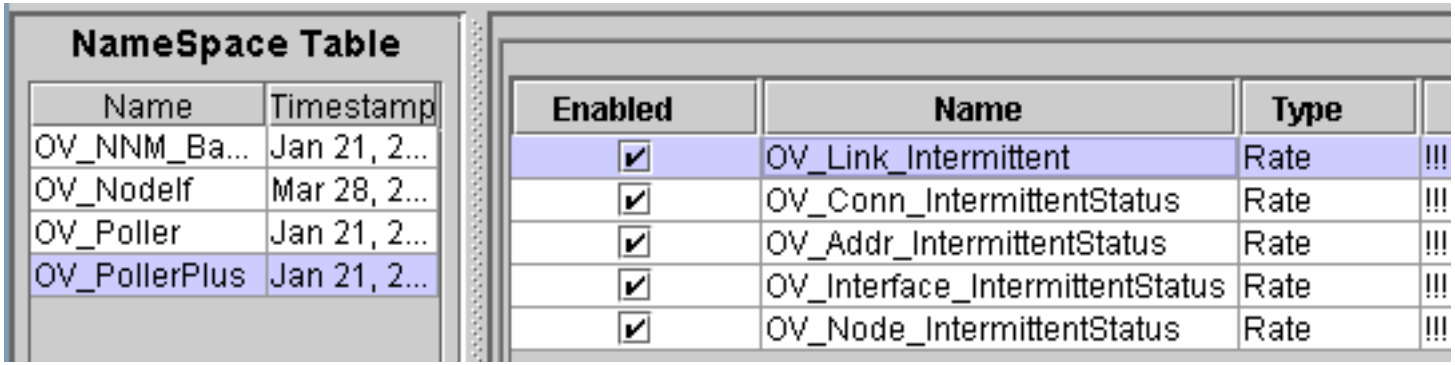
- Behavior:
  - Repeated event correlation becomes a “legacy” correlation with De-dup
  - Embeds subsequent matches under original event in alarm browser
  - Only indication to users to increment in correlated message count
  - Repeated event default correlations affecting status:
    - OV_Node_Up in V6.0+ (RepeatedTimeWidow = 1h)
- Status Alarms:
  - No logged status alarms affected by this correlation in any version.
- Topology Status:
  - No impact
- Configuration:
  - Correlation Manager

IntermittentStatus correlation behavior

- Behavior:
  - Detects flapping interfaces/nodes that would be “hidden” by PairWise
  - Also called “Router/Switch Intermittent Status” correlation
  - Also called “OV_Connector_IntermittentStatus”
  - Applies only to connector interfaces
- Status Alarms:
  - New Alarm in V6.31: OV_IF_Intermittent – OpenView enterprise 58982423
  - RATE_COUNT
    - Default is 4 in V6.31
    - Default is 5 in V6.4
    - Default is 4 in V7.0, 7.01
    - RATE_PERIOD Default is 30 minutes
- Topology Status:
  - Remember: suppressed status alarms aren’t suppressed in topology!
- Configuration:
  - Correlation Composer, OV_NNM_Basic and OV_Poller namespaces
  - netmon.lrf: –k shortPollTime=120; netmon.lrf:-k shortPollDownCount=2
- Irrelevant when using APA Polling

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: OV_APA_[INTERFACE|NODE|ADDR|CONN]_Intermittent
  - RATE_COUNT
    - Default is 2 in V7.01
    - 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

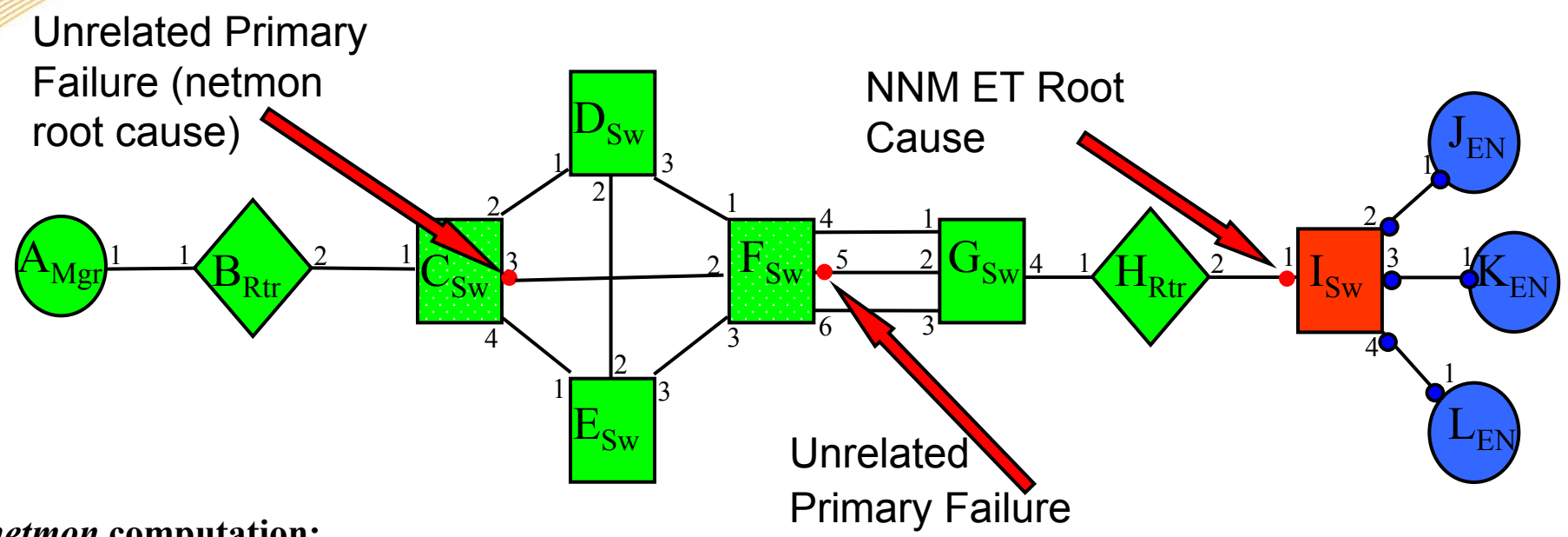
- Status Alarms:
  - Interface events embedded into node events for *netmon*-based status
  - Interface/Address/Connection/Node events embedded for *ovet_poll*-based status
  - Secondary alarms from "downstream" nodes embedded (*netmon*-based polling)
- Topology Status:
  - Primary Interfaces updated immediately, secondaries set as Unknown (*netmon*)
  - Primary entities updated immediately, no secondary status at all (*ovet_poll*)
  - Connector Node topology status other than Up/Down held 4 minutes
- Configuration:
  - Correlation Manager
  - Network Polling Configuration:
- Scope: ConnectorDown inputEventList *(*V7.01 default):
  - *netmon*:

If_Up, If_Down, ifUnknown, nodeUp, nodeDown, nodeUnknown, nodePrimary, nodeWarning, nodeMajor, nodeMarginal

  - *ovet_poll*:

APA_ADDR_UP, APA_ADDR_Unreachable, APA_IF_UP, APA_IF_Unreachable, APA_NODE_UP, APA_NODE_Unreachable, APA_CONNECTION_UP,

netmon vs. ET/APA path analysis for ConnectorDown



***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

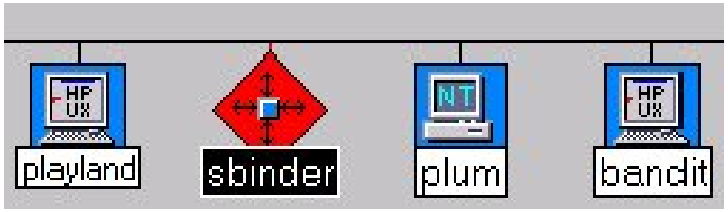
ET path engine computation (APA) using connector fault analysis:

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**)**

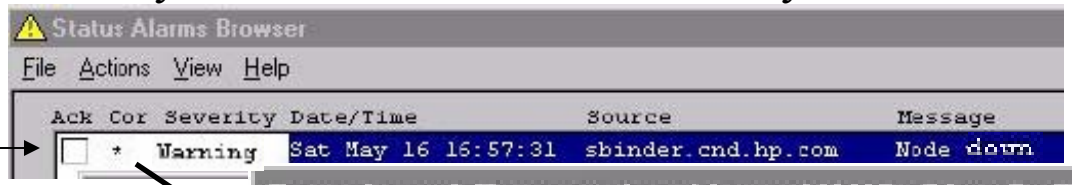
AGGR(F.4 G.1 **F.5 G.2** F.6 G.3**)** - G.4 H.1 H.2 I.1 I.2 J.1

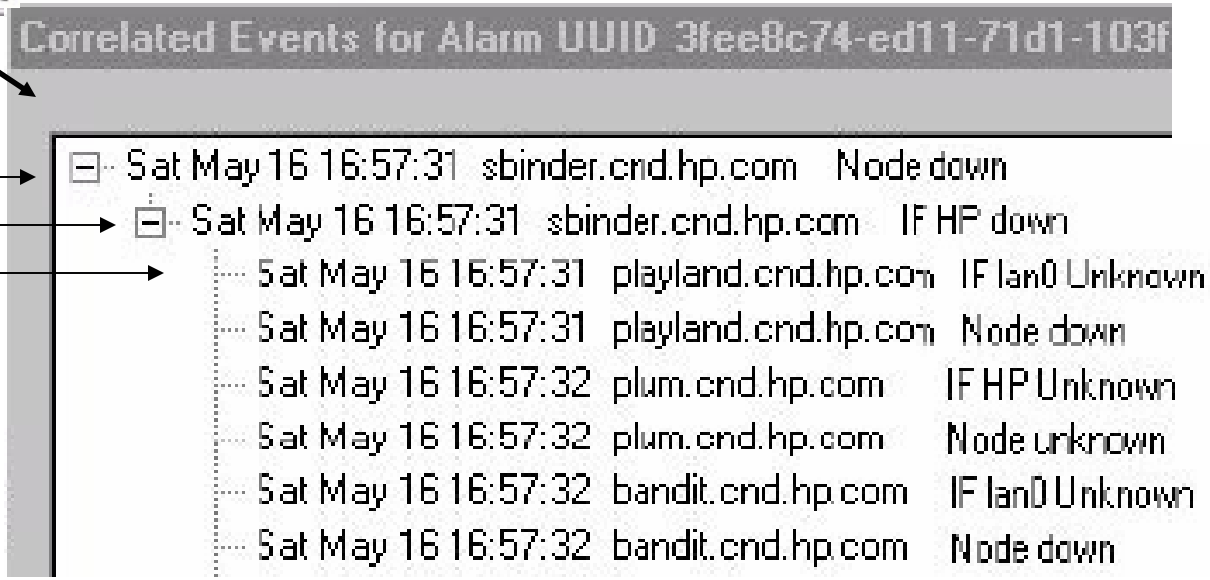
APA-based events: Primary: C3, F5, I1; Secondary to I1: J1, K1, L1

ConnectorDown correlation example 1

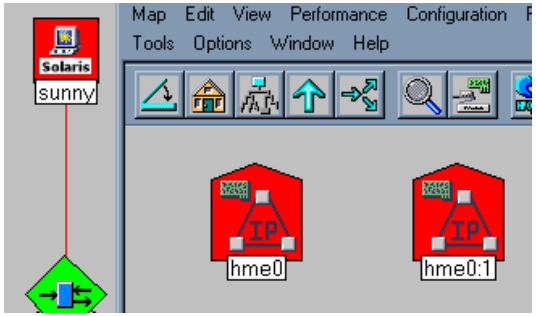


- ConnectorDown with *netmon* polling:
  - Note "triple" layers of embedding
    - Primary Parent Event – Node Down
    - Primary Child Event -- Interface Down
    - Secondary Grandchild Event – Secondary Nodes down or unknown

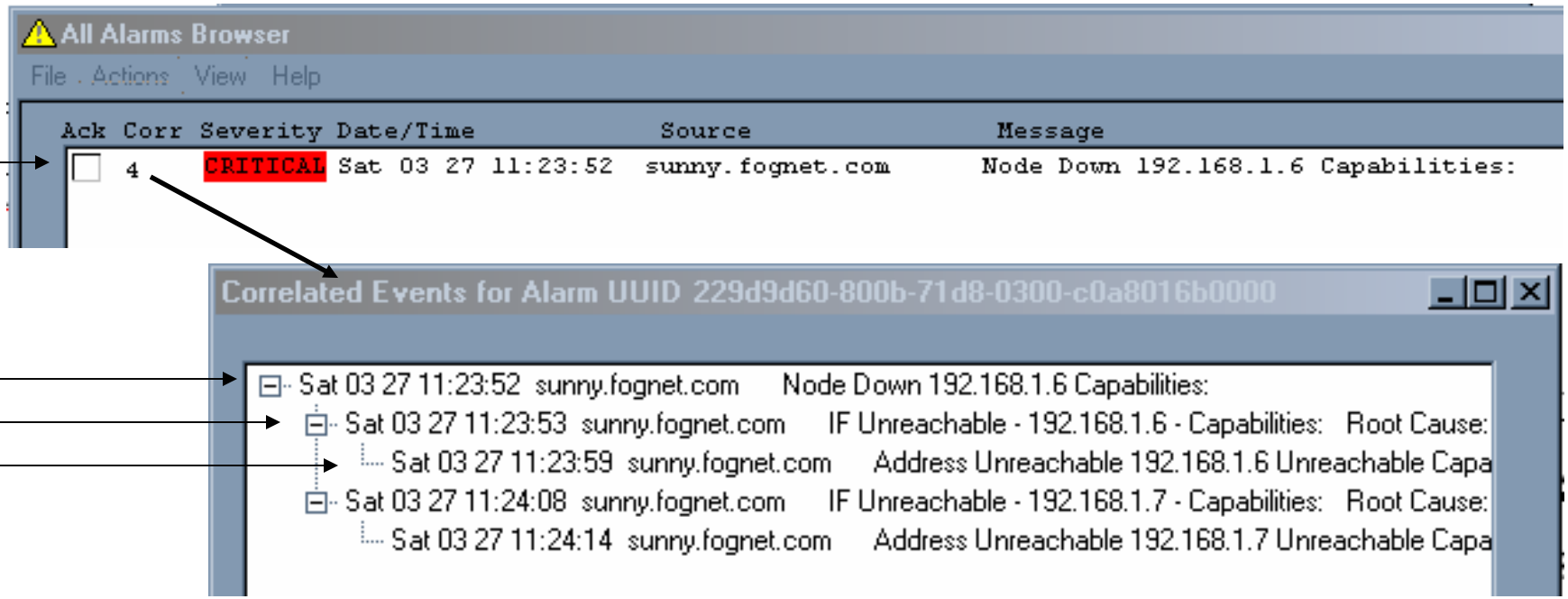




ConnectorDown correlation example 2



- 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

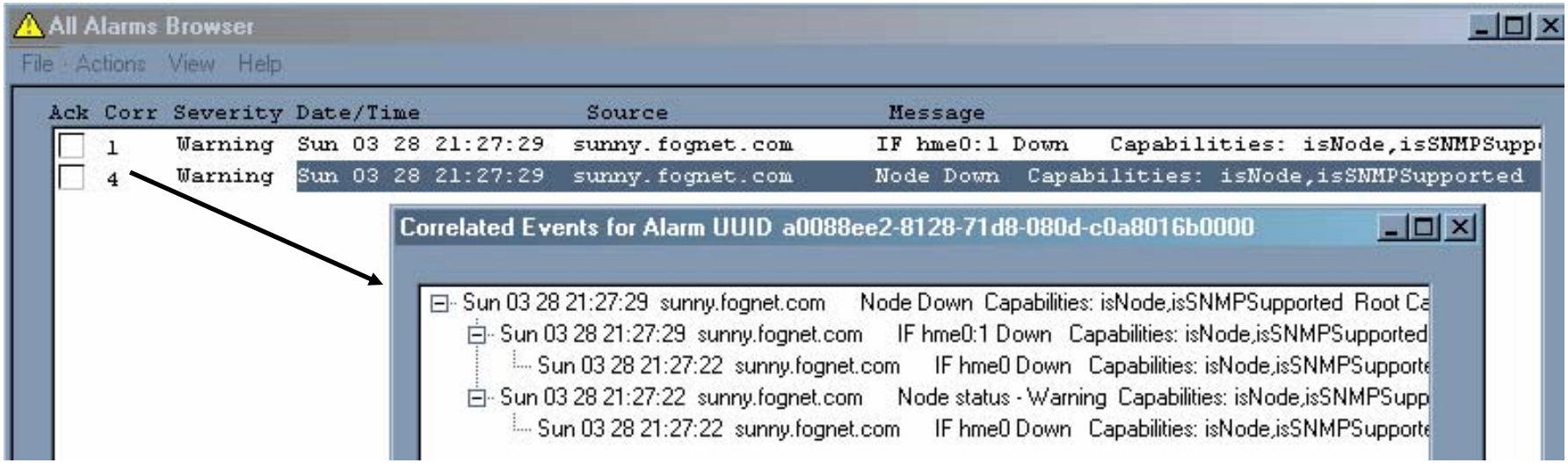
- Behavior:
  - AKA “Router/Switch Health”
  - Supplements ConnectorDown after major re-work of *netmon* status alarms V6.31
  - Takes advantage of V6.31 *netmon* dynamic polls to connector interfaces
  - NodeIf correlates *netmon*-based status of interfaces on the same device
  - Suppresses interface 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 V7.01, *netmon*-based status, NodeIf on (default)
  - Node Down never seen for simple devices – only Interface level events
- Example 2: : NNM V7.01, *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:



New status event varbinds supporting NodeIf, ConnectorDown

- *netmon*-based status events – selected varbinds of interest (New since V6.31)

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 APA status events varbinds

- Selected APA event varbinds of interest :

Varbind #	Description
$2	Timestamp event occured
$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	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 this is a secondary failure

- Event text: **`IF Down $5 $10 $6 Capabilities: $15`**

New APA events – NNM 7+

- These ship with NNM V7.0 but are never generated

All Status Alarms:

(Except:)

Error messages:

Statistics

OpenView	.1.3.6.1.4.1.11.2.17.1
rmon	.1.3.6.1.2.1.16
snmpTraps	.1.3.6.1.6.3.1.1.5

Events for Enterprise OpenView (.1.3.6.1.4.1.11.2.17.1):

Name	Identifier	Sources
OV_APA_ADDR_DOWN	Specific 58983011	ALL SOURCES
OV_APA_ADDR_Intermittent	Specific 58983016	ALL SOURCES
OV_APA_ADDR_UNREAC...	Specific 58983021	ALL SOURCES
OV_APA_ADDR_UP	Specific 58983001	ALL SOURCES
OV_APA_CONN_Intermittent	Specific 58983017	ALL SOURCES
OV_APA_CONNECTION_...	Specific 58983014	ALL SOURCES
OV_APA_CONNECTION_...	Specific 58983024	ALL SOURCES
OV_APA_CONNECTION_UP	Specific 58983004	ALL SOURCES
OV_APA_IF_DOWN	Specific 58983012	ALL SOURCES
OV_APA_IF_Intermittent	Specific 58983015	ALL SOURCES
OV_APA_IF_UNREACHABLE	Specific 58983022	ALL SOURCES
OV_APA_IF_UP	Specific 58983002	ALL SOURCES
OV_APA_Message	Specific 58983000	ALL SOURCES
OV_APA_NODE_DOWN	Specific 58983013	ALL SOURCES
OV_APA_NODE_Intermittent	Specific 58983018	ALL SOURCES
OV_APA_NODE_UNREAC...	Specific 58983023	ALL SOURCES
OV_APA_NODE_UP	Specific 58983003	ALL SOURCES
OV_APA_Statistics	Specific 58983032	ALL SOURCES

Other Correlations related to status

- Multiple reboot
  - Monitors coldStart & warmStart traps – external vs. internally-generated status
- Scheduled Maintenance
  - Disable status during backup windows, configuration windows, etc.
- Extended Topology SPI's
  - HSRP
  - Frame Relay SPI
  - Correlated event provides the following information:-
    - IF Index that went down
    - List of DLCIs that went from active to inactive
    - No of times they toggled between these states before the interface went down
    - Remote destination(s) affected
    - Correlated list of DLCI status change events and IF down event

General Considerations

- All
  - Monitor ovdumpevents: change flooding “log-only” events to “don’t log”
- Running V6.31or previous – “interim version”
  - Upgrade to V6.41 or V7.01
- Running V6.41
  - Pairwise: Disable or change DeleteOrAcknowledge & ChildImmediateOutput
  - Intermittent Status: Disable if changing PairWise
  - NodeIf: Change status event logging behaviors to show simple node status
  - Log topology status events previously logged (e.g. network|segment critical)
  - Log interface up status if desired
- Running V7.0 – “interim version”
  - Upgrade immediately to 7.01 and switch to APA poller
- Running V7.01
  - Switch to APA-based polling
  - Pairwise: Disable or change DeleteOrAcknowledge to Acknowledge
  - Intermittent Status: Disable if changing PairWise
  - Closely examine dedup.conf entries and consider additions/deletions
  - Disable ConnectorDown if using APA polling
  - Log APA up status alarms if desired

Tracing and logging

- Event logging
  - Dump BES, including "log-only" events (old trapd.log format): `ovdumpevents`
  - "Re-create" trapd.log: add "`-SOV_EVENT;t`" flag to pmd.lrf; see man/ref ov_event
  - Dump then tail: `ovdumpevents -t`
  - Dump BES (show corrlation log): `ovdumpevents -c "default"`
- Support Utilities
  - $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 cmds
- ECS Logging
  - ecsmgr
    - Log ECS input events to $OV_LOG/ecs/1/ecsin.evt# : `ecsmgr -log_events input on`
    - Log output events : `ecsmgr -log_events stream on`
      - Events output by ECS are logged to $OV_LOG/ecs/1/default_sout.evt#
      - Events suppressed by ECS are logged to $OV_LOG/ecs/1/default_sdis.evt#
    - More info in:
      - White Paper: Developing NNM Event Reduction
      - User Guide: Correlation Composer Guide

fin

Thank You

Prepared by

Mike Peckar
Fognet Consulting

business partner

hp invent