OPENVIEW 2001: OPENVIEW USER'S CONFERENCE

OpenView Forum International

NNM Status Polling: The Big Uneasy

The Mint Julep

Presented by Mike Peckar
Fognet Consulting

The Hurricane

What’s up with this session?

The “Node Down” Event is the subject of mis-understanding and misinterpretation. This session exposes the whole status polling infrastructure from the perspective of the Node Down event

By the end of this session, you will want to drink heavily. No problem - you are in New Orleans!

Agenda

- **The Node Down**
  - What is a “Node Down”?
- **Node Down Issues**
  - When is a “Node Down” not a “Node Down” and what can I do about it?
- **Improving Status Events**
  - Some steps you can take – with emphasis on those things you can do yourself relatively easily
- **Drink Recipes**
  - Please don’t skip ahead

The Node Down Event

- An event generated by OpenView's *netmon*
  - Node Down is OpenView enterprise event
    - .1.3.6.1.4.1.11.2.17.1.58916865
    - Event generated by NNM server but source is set to target
  - An ovevent sent internally to NNM
    - NNM internally reformats all SNMP traps to ovevents
    - Adds status, logging, actions, and are transported reliably
  - Node down is only generated when all managed interfaces are detected down based on:
    - The Interface Down event
    - NNM topology database known status of other interfaces

The Node Down Event

- *netmon* generates status polls:
  - ICMP mask requests for discovered IP interfaces
  - SNMP or ARP for discovered Level 2 interfaces
  - SNMP status polls for selected sets of devices (6.2)
  - IPX for NT version only
- Interval, timeouts and retries are configurable
  - Default: 5m interval, 0.8s timeout with 2 retries
- Interface Down trap conveys status poll failure
  - Default: a log-only OpenView event

The Node Down Event

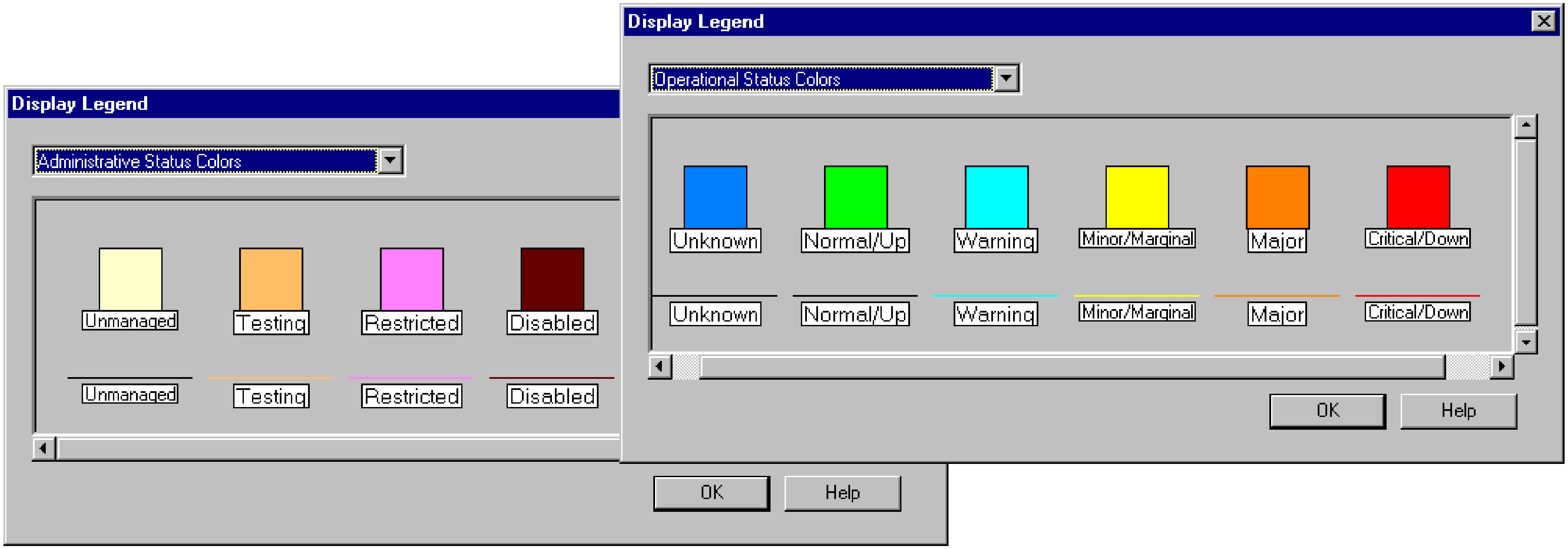
- The morphology of a failed ICMP status poll
  1. *netmon*-scheduled ICMP status poll to interface fails
  2. Interface Down trap generated by *netmon* as log-only
  3. ECS Connector Down logic is applied at this point
  4. Interface status updated in topology by *ovtopmd*
  5. IPMAP changes status at node level (interfaces) - propagated status at segment level (nodes) & above
  6. *ovtopmd* checked for status of node’s other interfaces
  7. If all others down, *netmon* generates Node Down

The Node Down Event

- What is a Node **Up** Event?
  - – An event generated by OpenView’s *netmon*
  - – Generated when ***all*** managed interfaces for a node responds to their respective status polls
  - – Status poll success generates Interface_Up Event
  - – Node Up always paired with log-only Interface_Up

The Node Down Event

- Operational & Administrative Status



Display Legend

Administrative Status Colors

Unmanaged	Testing	Restricted	Disabled
Unmanaged	Testing	Restricted	Disabled

Display Legend

Operational Status Colors

🟦	🟩	🟦	🟨	🟧	🟥
Unknown	Normal/Up	Warning	Minor/Marginal	Major	Critical/Down
Unknown	Normal/Up	Warning	Minor/Marginal	Major	Critical/Down

The Node Down Event

Status polling-related OpenView Events

Event	***Logging***	***Status***
OV_IF_Down	*Log-only*	*IF Critical*
OV_IF_AdminDown	*Log-only*	*IF AdminDown*
OV_IF_Unknown	*Log-only*	*IF Unknown*
OV_IF_Testing	*Log-only*	*IF testing*
OV_IF_Up	*Log-only*	*IF Normal*
OV_Node_Down	*Status event*	*Node Critical*
OV_Node_Up	*Status event*	*Node Normal*

The Node Down Event

IF Status for L2-discovered interfaces (all log-only)

– SNMP-Supported defaults per E. Pulsipher, 5.0+

ifAdminStatus	**ifOperStatus**	**OV Status**
down	any	DISABLED
testing	any	TESTING
up	up	NORMAL
up	down	CRITICAL
up	testing	TESTING

– SNMP-Unsupported: inferred from port table maps:

OV Status is CRITICAL for Down and NORMAL for Down

The Node Down Event

- Configurable SNMP status poll (NNM 6.2+)
  - Applies to IP addr's & ranges entered in netmon.snmpStatus
  - Queries ifOperStatus and ifAdminStatus
  - Useful when no route to device (L2-connected)
  - Status conveyed on timeout varies according to
    - netmon –k snmpTimeoutImplies=status lrf setting
      - Unknown, unchanged, critical (default)
    - netmon.statusMapping file triplets: Admin:Oper:Status
      - E.g.: up:down:down
      - sets all adminUp and operDown interfaces to critical
  - Settings do not apply to netmon-discovered L2 interfaces?

Node Down Issues

1. Node is down but no Node Down generated

2. Node comes up but no Node Up generated

3. Node is up but false Node Down generated

Node Down Issues

1. Node is down but no Node Down generated
   - Node is unmanaged or an interface is unmanged
   - Node unreachable from node other than NNM
   - Target node is multi-homed: Node Down generated only when **all** managed interfaces are reported down
   - Status poll hasn’t occurred yet (behind schedule?)
   - Node down suppressed for ECS Connector Down: a node in the network path to the target node is down.
   - SNMP-polled and snmpTimeoutImplies=unchanged

Node Down Issues

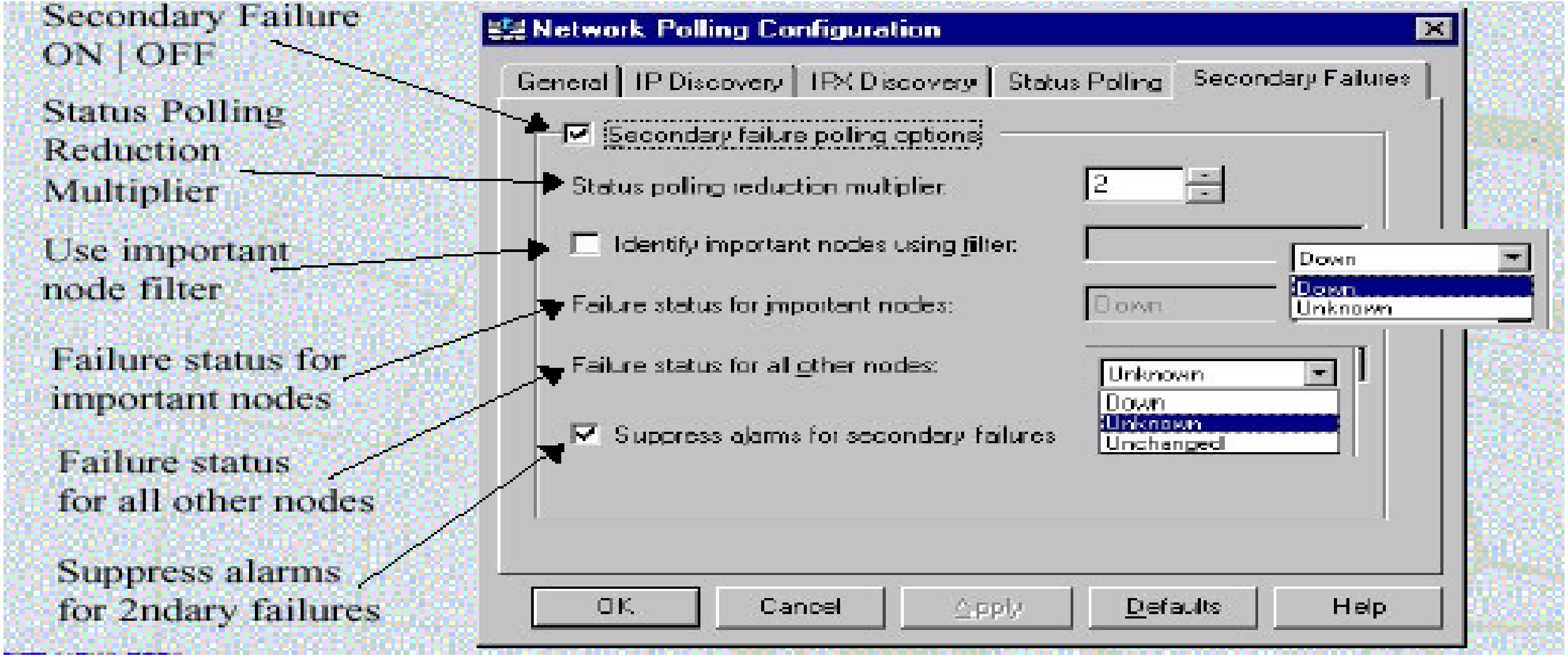
1. Node is down but no Node Down generated (Cont’d)

- ECS Connector Down, Node Down, and status polling
  - Purpose: prevents “cascade failures” and throttles status polls
  - ECS correlates with network path as maintained by netmon to:
    - Distinguish Primary failures: those interfaces closest in path
    - Distinguish Secondary failures: those interfaces that are “downstream”
    - Filter Important Nodes: defines nodes not to include as secondary
  - Default: Set secondary as unknown and suppress alarms
  - Utilizes drill-down capabilities of event browser via eventdb

Node Down Issues

1. Node is down but no Node Down generated (Cont'd)

- ECS Connector Down configured via Polling Config:



Secondary Failure ON | OFF
Status Polling Reduction Multiplier
Use important node filter
Failure status for important nodes
Failure status for all other nodes
Suppress alarms for 2ndary failures

Network Polling Configuration

General
IP Discovery
IPX Discovery
Status Polling
Secondary Failures
OK
Cancel
Apply
Defaults
Help

☑ Secondary failure poling options

Status poling reduction multiplier: 2

□ Identify important nodes using filter:

Failure status for important nodes: Down (Down / Unknown)

Failure status for all other nodes: Unknown (Down / Unknown / Unchanged)

☑ Suppress alarms for secondary failures

Node Down Issues

1. Node is down but no Node Down generated (Cont’d)

- What to do
  - Check polling config and ECS settings
  - Check managed/unmanaged states of interfaces
    - Use ovtopodump
  - See Node Down Issue #3 regarding polling falling behind
  - See Section 3: Improving Status Polling

Node Down Issues

2. Node comes up but no Node Up generated
   - Node multi-homed and at least one interface is still down when node becomes reachable from NNM.
   - Example: Remote router with a backup ISDN
     - Reachable interface goes down, far end interfaces also go down, NNM sees all interfaces down: Node Down
     - Reachable interface goes up, far end interfaces can be polled and come up, ISDN interface down, NO Node Up
   - What to do:
     - See Section 3: Improving Status Polling

Node Down Issues

3. Node is up but false Node Down generated

The infamous “False Node Down”

Case A:

Issues external to NNM Server

Case B:

Issues internal to NNM Server

Node Down Issues

3. Node is up but false Node Down generated (Cont’d)

Case A: External “False” Node Downs

ICMP PINGS don’t have much of a chance these days:

- ICMP protocol dropped by busy router due to priority
- ICMP protocol blocked by constantly changing security tools/firewalls/policy management
- ICMP timeouts due to lower level latencies: net congestion
- ICMP timeouts due to higher level latencies: VPN Gateways
- ICMP reply timeouts due to target system resources (NT, cheap NICs, etc..)

Node Down Issues

3. Node is up but false Node Down generated (Cont’d)

Case B: Internal “False” Node Downs

NNM tuning, server OS tuning, or sizing issues:

- Interval/Timeouts/Retries improperly configured
- ICMP receive buffer overruns due to:
  - Excessive ICMP Queue length
  - Excessive number of unpingable interfaces
- DNS configuration and lookup latency
- Excessive number of managed interfaces

Node Down Issues

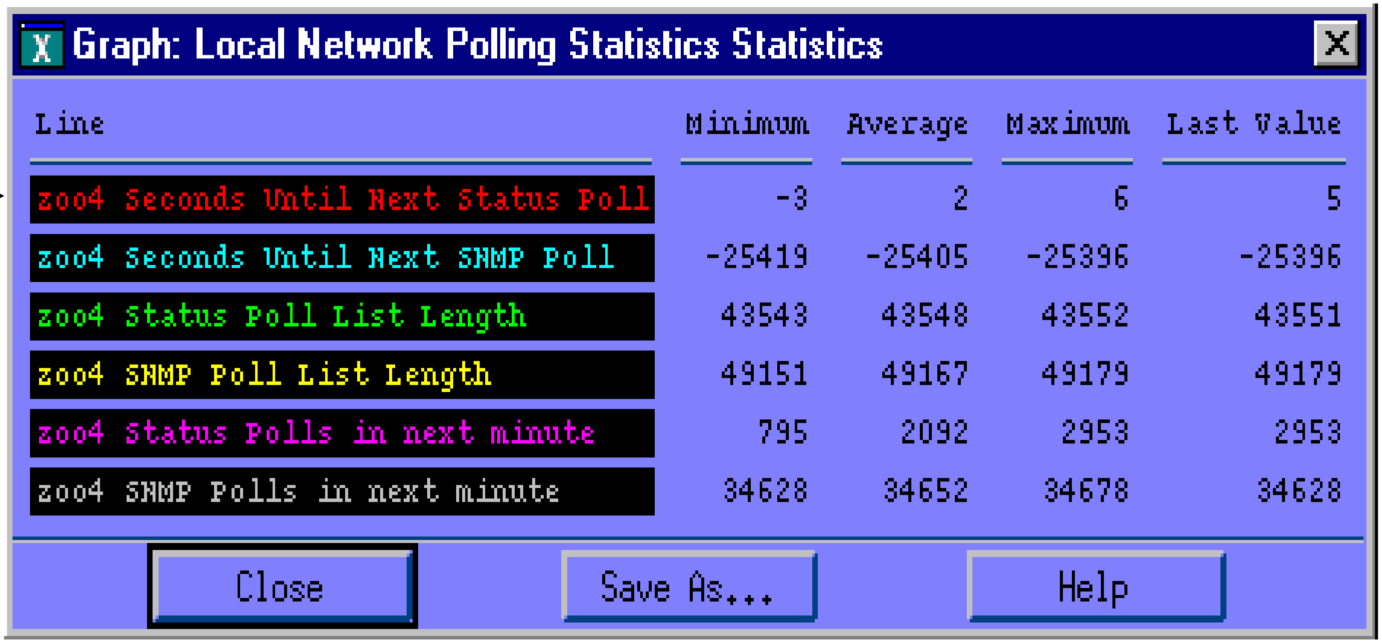
3. Node is up but false Node Down generated (Cont'd)

What to do first: Determine if issues external or internal

– View Network Polling Stats under Fault Menu:

Seconds until Next ICMP poll > 0 = OK!

False Node Down issues are external



Line	Minimum	Average	Maximum	Last Value
zoo4 Seconds Until Next Status Poll	-3	2	6	5
zoo4 Seconds Until Next SNMP Poll	-25419	-25405	-25396	-25396
zoo4 Status Poll List Length	43543	43548	43552	43551
zoo4 SNMP Poll List Length	49151	49167	49179	49179
zoo4 Status Polls in next minute	795	2092	2953	2953
zoo4 SNMP Polls in next minute	34628	34652	34678	34628

Node Down Issues

3. Node is up but false Node Down generated (Cont’d)

What to do: First: See if issues are external or internal

- ovstatus –v netmon
  - Provides *netmon* performance troubleshooting info
- netmon tracing and logging
  - Provides several layers of detail
  - See man/ref pages
- Start data collection on network polling statistics
  - Catch trends in *netmon* performance

Node Down Issues

3. Node is up but false Node Down generated (Cont’d)

What to do: View log-only events:

- Method 1
  - Log events to *trapd.log* (legacy, performance hit)
  - In *pmd.lrf*: OVs_ YES_ START**::-** ***SOV_ EVENT;*** **t:** OVs_ WELL_ BEHAVED: 15: PAUSE
- Method 2
  - *ovdumpevents* –f mytraplog.txt
  - *ovdwevent* exports the data from the event database to the default (SOLID) data warehouse or other configured/supported ODBC store, but Log-only data not exported by *ovdwevent* by default:
    - *touch* $OV_ANALYSIS_CONF/NO_EXTREME_EVENT_FILTERING

Node Down Issues

3. Node is up but false Node Down generated (Cont’d)

What to do: Case A - external issues:

- Check ICMP protocol priority in router configs
- Implement operator procedures
  - Manually re-poll devices reported down by NNM
  - Poll nodes from alternative sources, *traceroute*, SNMP, etc
- Implement automated actions to re-poll nodes
  - Using automatic action scripts on node/interface down
  - Using event correlation engines
  - See Section 3: Improving Status Polling for examples!

Node Down Issues

3. Node is up but false Node Down generated (Cont'd)

What to do: Case B - internal issues:

- Improve status polling performance
  - Turn off superfluous polling (http, level-2, SNMP v2)
  - Consider netmon switch for burst mode (-b 20)
  - Check impact of ICMP polls w.r.t traffic near NNM LAN
  - Increase status poll reduction multiplier in ECS Conn. Down
- Don't forget OS and Hardware issues
  - Like kernel parameters (see NNM Installation Guide)
  - Or Network performance settings (nettune, ndd, etc)
- Address scalability issues by distributing NNM

Node Down Issues

3. Node is up but false Node Down generated (Cont’d)

What to do: Case B - internal issues:

- Examine polling timeouts, retries & intervals
  - Remember: timeout doubled for every retry…
    - Default: t: 0.8 r: 2 Status poll fails after 5.6 seconds
      t: 2.0 r: 4 Status poll fails after 30 seconds
      t: 5.0 r: 5 Status poll fails after 5 min, 15 sec

    - Default Interval: 5 minutes

      Balance timeout/retries with polling intervals
    - Set status polling under Options – SNMP Configuration

Node Down Issues

3. Node is up but false Node Down generated (Cont’d)

What to do: Case B - internal issues:

- ICMP receive buffer overruns on NNM system
  - See –q switch in netmon man/reference page:

-q ICMP-queue-length Indicates to netmon that it should allow up to ICMP-queue-length outstanding ICMP requests at a time for status polling of interfaces). (default: UNIX = 20, Windows NT operating system = 3) PLEASE NOTE: If you increase the ICMP-queue-length too much, you may find that ***interfaces may be declared Critical when they are actually up***. This is because there is a limited system buffer for incoming ICMP responses, and having too many arriving at the same time can cause some to be lost. If you find false Critical indications after increasing the ICMP-queue-length, you should reduce the length to the point where the problem goes away.

Node Down Issues

3. Node is up but false Node Down generated (Cont’d)

What to do: Case B - internal issues:

- Do not muck with queue lengths haphazardly – remember:
- longer timeouts/retries = fewer simultaneous polls
  - Queue may fill up and cause status polling to fall behind:
  - nnmICMPSecsUntilNextPoll values will be less than 0
  - False Node Downs not always generated by this condition, but status polls are failing to occur on schedule
- Increasing ICMP Queue length risks receive buffer overruns
  - Polling may “catch up, but False Node Downs reported
  - Increase interval, manage fewer interfaces, scale up.
- Shorter timeouts/retries = greater risk of latency timeout

Node Down Issues

3. Node is up but false Node Down generated (Cont’d)

What to do: Case B - internal issues:

- DNS configuration and lookup latency
  - See Doug Stevenson’s excellent session on this topic: *Name Resolution in Network & Systems Management*
    - Which took place today at 1:00 ☹
  - Note each ICMP status poll executes a lookup! What to do:
    - Implement consistent and ubiquitous naming conventions
    - Test name resolution latency
    - Use loopback addresses on your routers and switches
    - Check netmon performance statistics
    - Implement cache-only DNS server on NNM to conserve resources

Node Down Issues

3. Node is up but false Node Down generated (Cont’d)

What to do: Case B - internal issues:

- DNS configuration and lookup latency for NNM 6.2+
  - Use *netmon*’s -U minNameServerAlertAvgMsecs lrf switch
  - After netmon makes 100 Name Service lookups, it will continuously monitor the total average time for name server responses to validate that there is adequate performance of the Name Service system. If the running average exceeds minNameServerAlertAvgMsecs milliseconds, *netmon* will generate an OV_NS_PerformErr event. This event indicates that netmon is unnecessarily slowed by slow Name Service requests. To turn off this feature, set minNameServerAlertAvgMsecs to 0. Run netmon -? to see the default value of this option.

Improving Status Events

- In General:
  - Rename events in Options – Event Configuration:

Event	Text	Rename to:
OV_Node_Down	Node Down	Node unreachable from NNM
OV_Node_Up	Node Up	Node reachable from NNM
OV_If_Down	If Down	If $7 fails status poll
OV_IF_Up	If Up	If $7 responds to status poll

$7 is SNMP event variable binding for Interface Name, e.g. Lan0

Improving Status Events

- Ping Reports – 6.2 *netmon-snmpCollect* hook
  - Round Trip Time - msec between send and receive
  - Ping Retry % - proportion of configured retries used
  - Netmon collects, but does not store by default
  - Define filters to name node sources & configure in Options – Data Collection & Thresholds
  - **`$OV_CONF/snmpRep.conf`** shows MIB Expressions used for these collections
  - Or, Turn off to improve netmon performance

Improving Status Events

- If down and If Up Events:
  - Copy event and change from log-only to status event
  - Set Multi-homed devices as sources for new event
- Get fancy if you wish…
  - Find multi-homed devices using:
    - `ovobjprint -a "TopM Interface List”`
  - Use external list as source for event (supported)
- This buys you:
  - Status alarms for important previously log-only events
  - No duplicate alarms for single-homed nodes

Improving Status Events

- Simple re-polling
  - Write a simple script that runs as automatic action to the Interface Down event that:
    - Exits if too many instances are running (message flood)
    - Runs NNM’s *netcheck* utility to re-poll interface
    - Report results into a log, or as email, or uses:
      - SendMsg.ovpl in $OV_CONTRIB to send an *ovevent*

Get Fancy if you wish:

- Query NNM topo or object databases for node capabilities
- E.g. If SNMP Supported, check SNMP UpTime

Improving Status Events

- More capability-based scripted actions
  - If SNMP reports multiple interfaces report status of other interfaces via rnetstat – compare this to status of other interfaces as reported in NNM topology database, contrast this with SNMP ifTable status queries.
  - If SNMP reports agent matches your list of TCP-capable agents, attempt a TCP port connect and report results.
  - Write capability logging scripts as auto-action to the Node Added (discovery) event – and key re-polling scripts (auto-actions to If Down) to results. Rediscover your network to populate lists.

Improving Status Events

- Capability-based re-polling choices
  - Scripted auto actions on Interface Down trap
    - Using NNM utilities – using PERL with NET:PING
  - Event Correlation engines
    - ECS – Netcool – NerveCenter – Taave – Many others
  - Freeware and simple polling and status tools
    - MTRG - What’s up Gold - PERL w/ NET:PING
  - Developer’s tools
    - NNM DevKit - CSOV Perl Mod for direct manipulation of NNM Icons based on your poller

Improving Status Events

- Recap
  - Node Down is logged, but log-only Interface Down is of much greater importance
  - Interface Down doesn't mean the interface is down
  - A standard status poll in NNM is a single ICMP ping
  - Issues arise from the strangest of places (like DNS)
  - You can do simple things to mitigate confusion
  - You can do simple things to improve polling integrity
  - Things can get very complex very quickly, too, so…

Mint Julep

- The perfect julep comes of infusing the bourbon with the mint and letting it have a night's rest.
- 1 bottle Kentucky Bourbon
- 3 cups Fresh Mint Leaves
- 24 Fresh Mint Sprigs
- 6 Lemon Twists
- Bring a half-cup of water to a boil. Remove from heat and add 1 cup of sugar. Refrigerate until cool. Pour bourbon into a 1-1/2 quart jar and add mint leaves. Cover and refrigerate overnight. Strain liquor into a pitcher and discard the mint. Sweeten to taste with sugar and water mixture.
- Fill a julep cup or a Collins glass with cracked ice. Add infused bourbon and stir until glass frosts. Take a twist of lemon and rub it on the rim of the glass and toss it into the drink. Garnish with mint sprigs. Serves 6.

Hurricane

Ingredients:

- dark rum
- passion fruit juice
- Hawaiian Punch

Fill a Blender ½ full with ice. Add 8 oz Dark Rum and equal parts passion fruit juice. Fill the rest with the punch and blend. Add squeeze of lime. Make approximately 3 16-oz servings

Hand Grenade

- Ameretto
- 2 shots Melon Liquer
- 2 shots Everclear (190 proof)
- Pineapple juice

Mixing instructions:

Begin by filling the glass halfway with Amaretto, add shots of melon, add shots of everclear, add rest of glass with pineapple juice, stir, then pour into glass with ice.

Fin

Laissez les bon temps rouler..