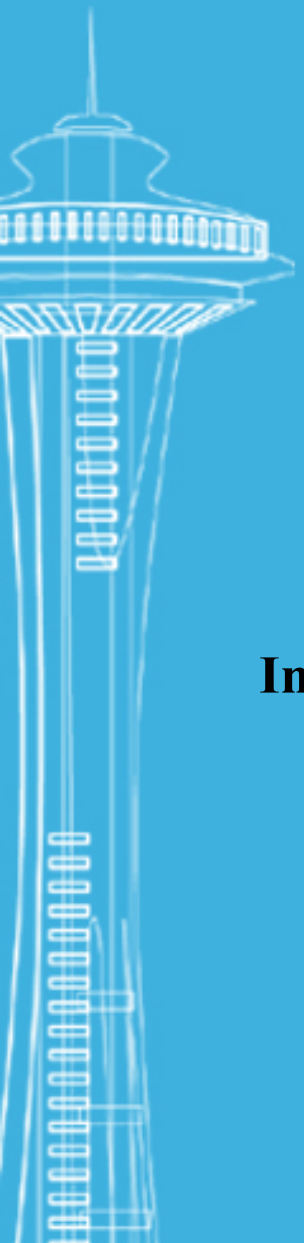
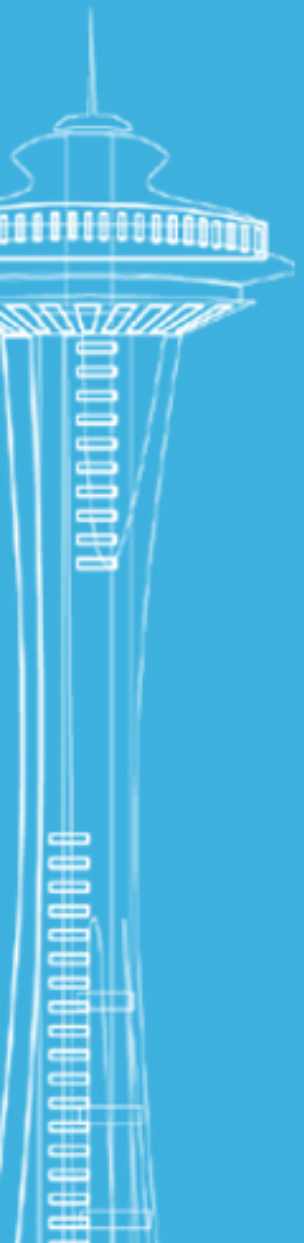


The Low Down on a Node Down:

Improving on NNM's Default Status Event Configuration Settings

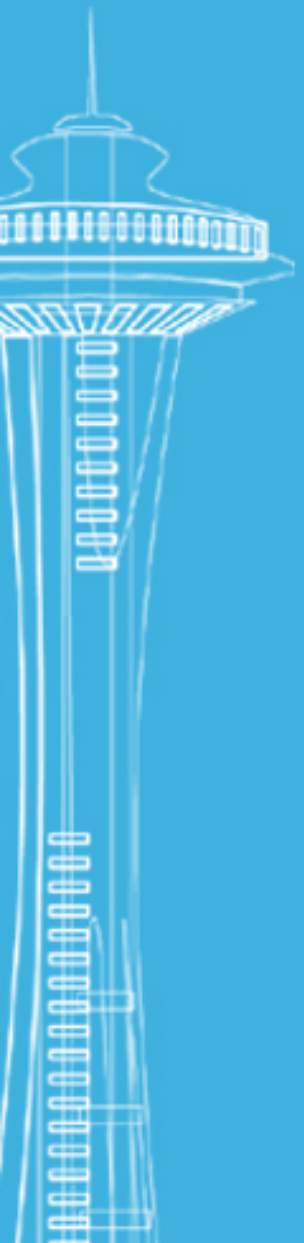
Mike Peckar
Fognet Consulting



Audience:

NNM Users who are…

- Confused by NNM default status events
- New to NNM event customization
- Looking for ways to improve status events



Scope of this session

The Problem:

NNM default status event behaviors confusing

The Causes:

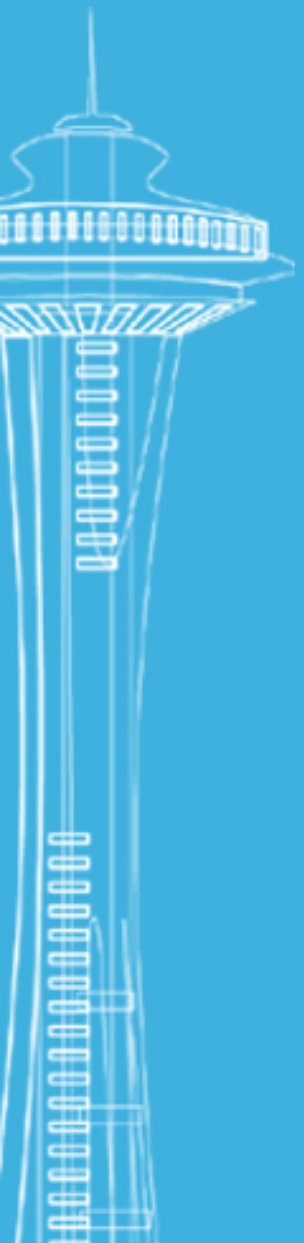
Semantics, default settings

The Solution:

Understand and improve (customize) status events

Bonus:

Script to provide improved status event handling



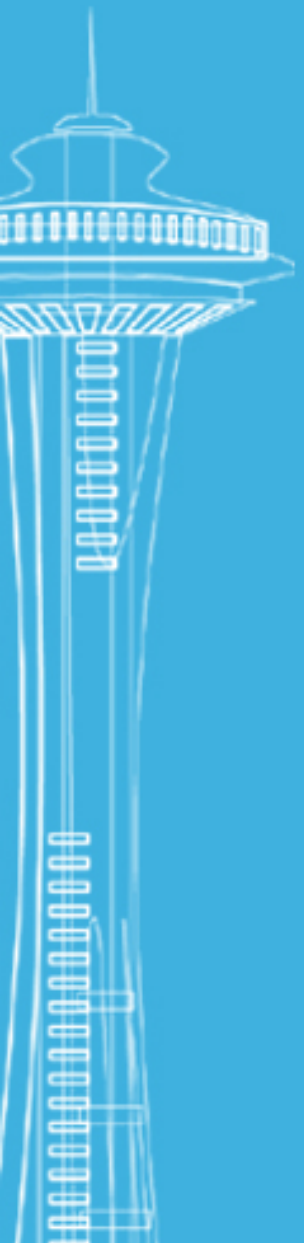
Background

Status Events

- Lines blur between SNMP traps and OV events
- “node down” is a misleading name for this event
- Default logging behaviors are weird
- Behavior differ for singly vs. multi-homed nodes

Examples

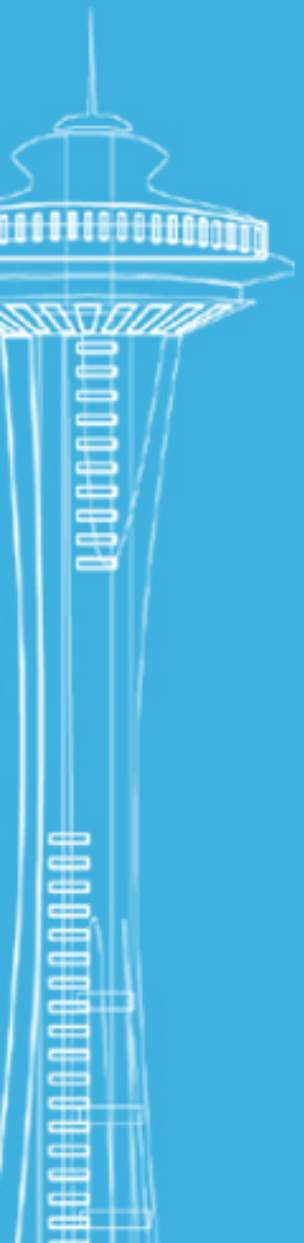
- Node up events but no corresponding node down
- Node down, node recovers, but no node up event
- Device status changes, but no corresponding events received



Status event Morphology

NNM Discovers nodes and places icons on maps
NNM issues ICMP polls to interfaces for status
NNM infers status via SNMP if Bridge MIB supported
NNM nodes status reflected via events & icons in maps

* Status events are internally generated by NNM *



SNMP traps vs. “OV events”

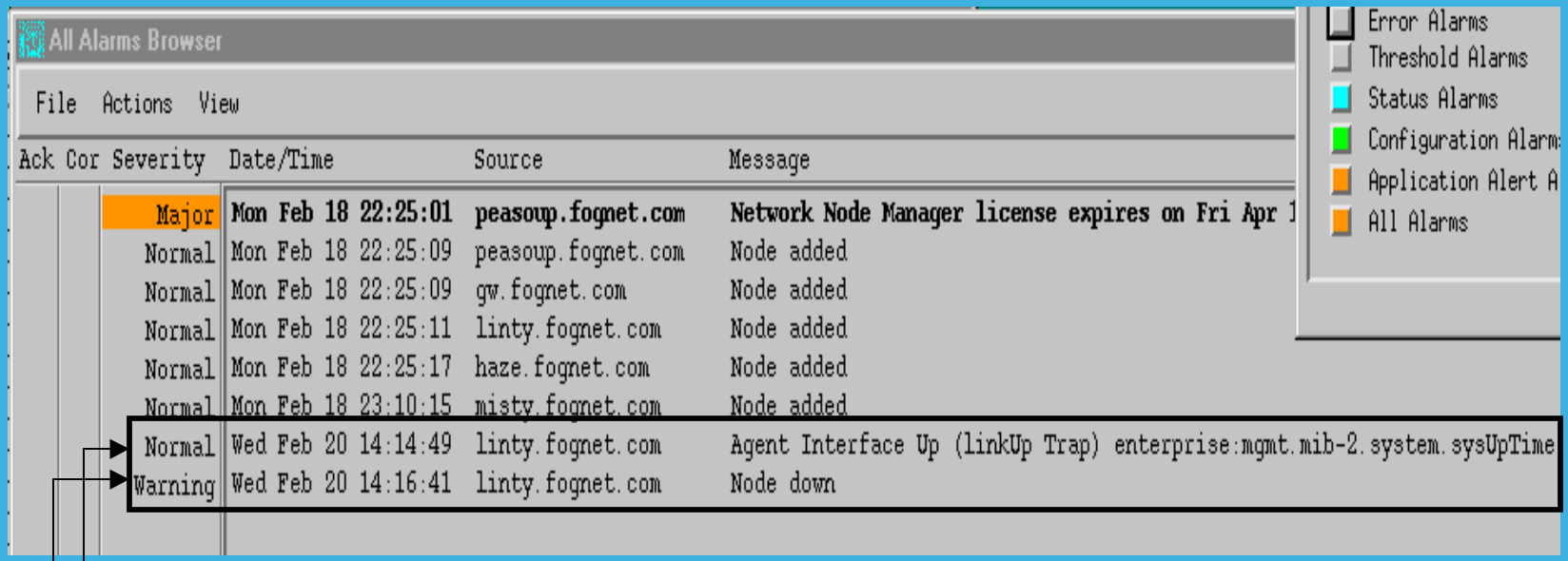
SNMP traps encapsulated as OV events with added value:

- Severity
- Logging behavior
- Actions
- Forwarding behavior: TCP vs. UDP for SNMP

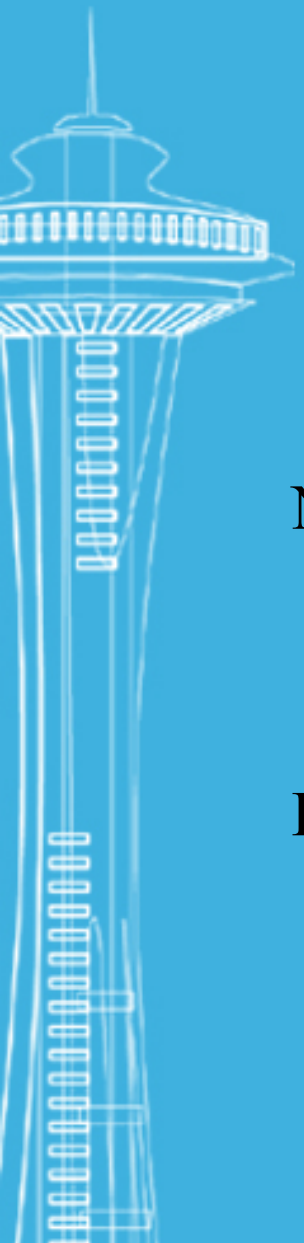
NNM status events are OV events generated by NNM

- Using the NNM SNMP API
- Using SNMP enterprise 1.3.6.1.4.11.2.17.1
- ** NNM Status event source set to node affected **

OV event “source” vs. SNMP Trap “source”



Event is SNMP Trap and source is “linty”
Event is OV Event and source is “peasoup”



Issues relating to status events:

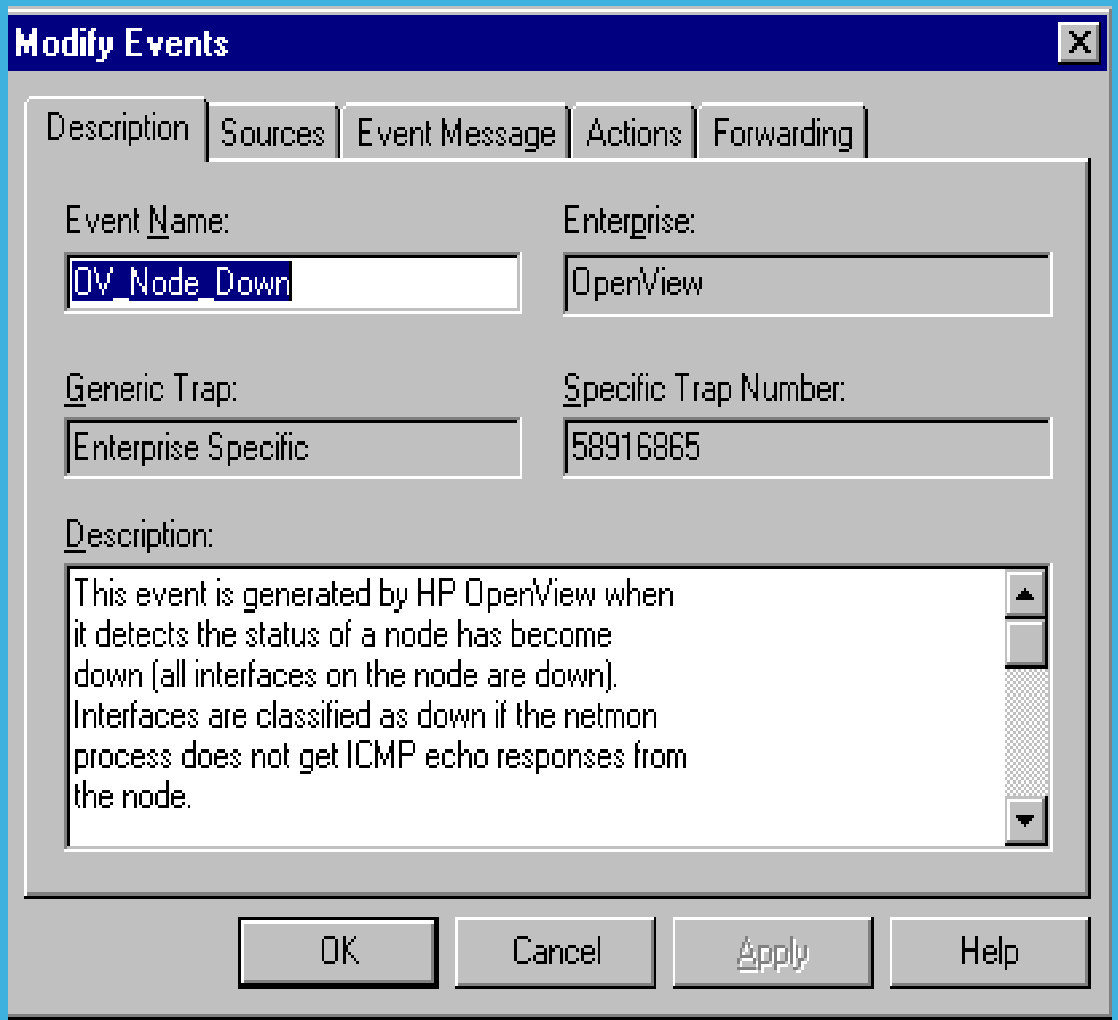
NNM Map status is “interface-driven”

- Status propagated from interface level icon colors
- Status polls are issued to interfaces

Event Browser events are “Node-driven”

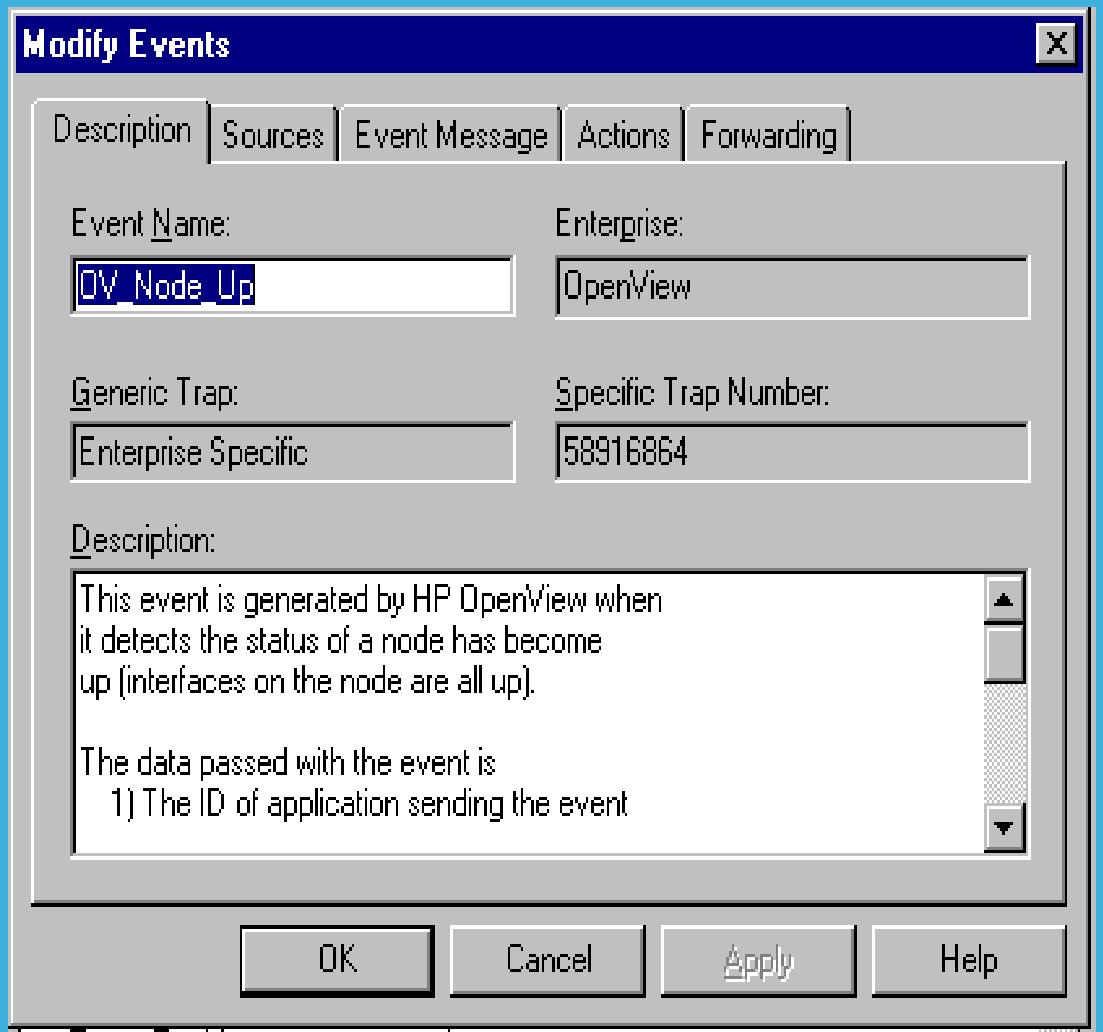
- Events sources are SNMP Agents: one node = one agent
- Interface-related events are set to “Log Only” by default
- Node status determined empirically from interface status

A Closer Look: OV event "Node Down"

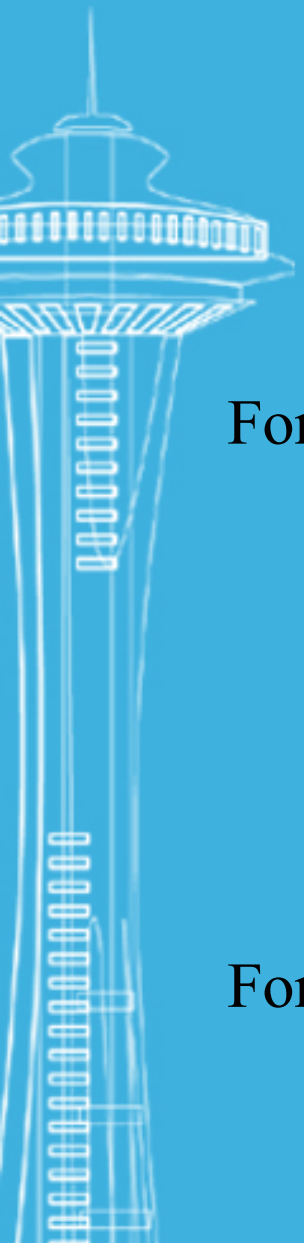


Note:
node down event generated as a result of Interface down events

A Closer Look: OV event "Node Up"



Note:
node up event
Generated as
a result of
interface up
events



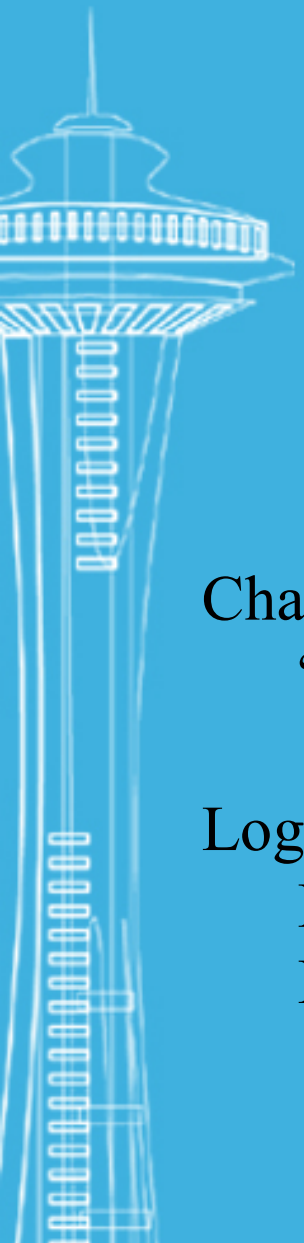
Issues relating to status events

For multiple interface devices:

- Node down will never indicate individual interface status
- Node up unpredictable, for example:
  - Node goes down, most interfaces come back up – no node up event
  - Just one interface comes up, node up event may be generated
- Many monkey wrenches
  - HSRP, backup ISDN interfaces, “flapping” ports on switches.

For single interface devices:

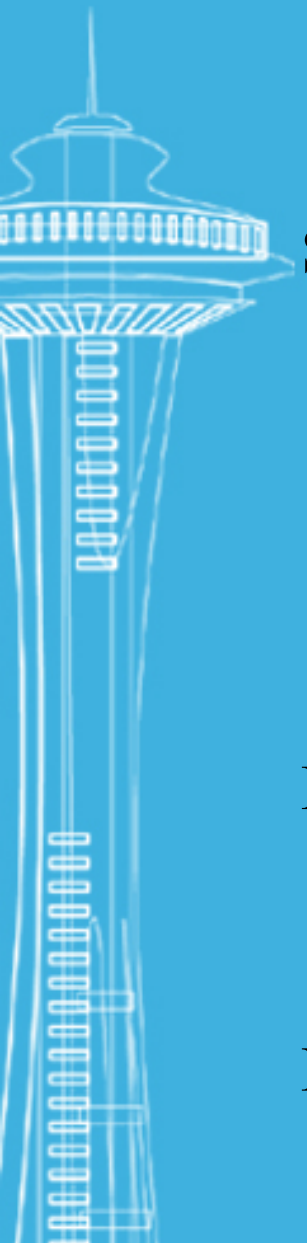
- Interface status always directly maps to node status
- Two events if interface status events are logged



Suggested event customizations:

Change Event texts to more clearly reflect real meaning
- "node down" means "all interfaces unreachable from NNM"

Log interface up and interface down events
- But consider topology and device characteristics (more below)
- De-duplicate node/interface status pairs (more below)



Topology matters!

Switched topologies: Many interfaces mapping to switch ports

- Consider NOT logging interface status events (the default)
  - Note switch port status doesn't reflect end-node status
- Study and test impacts of Layer 2 and SNMP polling options
  - netmon "discover layer 2 objects" switch discover option
  - netmon.statusMapping and netmon.snmpStatus
  - netmon –k nonIPStatusPolls, -k bridgeMIB –k segRedux options

Routed topologies: Interfaces more critical vis-à-vis status

- Logging interface status events becomes more critical
- WAN interfaces particularly critical

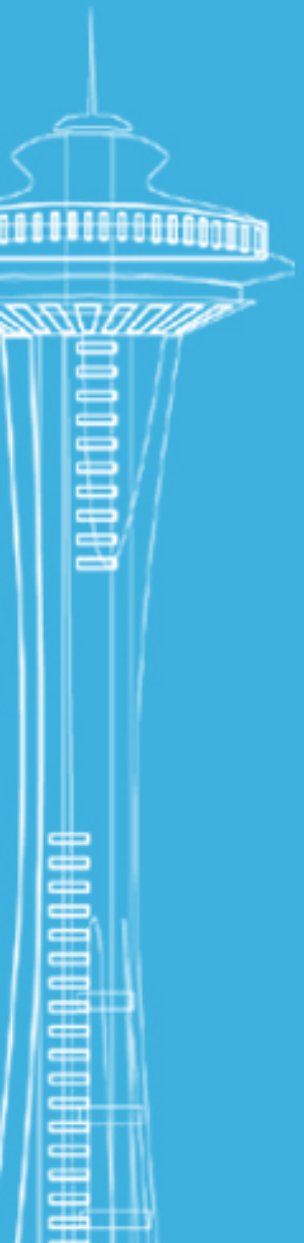
Mixed topologies – No simple solutions, must get creative

- Consider logging interface status events only for specific devices
- More below

Event Customizations – text and logging

Suggested Changes to default status events

Default Event text	New Event Text	Default Logging	Suggested Logging
Node down	Node Unreachable	Status Alarm	Status Alarm
Node Up	Node Reachable	Status Alarm	Status Alarm
IF Down	If $7 Unreachable	Log-Only	Log Some!
IF Up	If $7 Reachable	Log-Only	Log Some!



Event customizations – Log Some

Log Interface status events, but only for multi-homed nodes

1. Make copies of interface down/up events and set:
   - Sources as all single-homed nodes
   - Status as “log-only”
2. Set interface down/up events from log-only to log as status events
3. Use handy script included (See 23 easy steps below)
4. Or, manually select list of sources to exclude for interface status

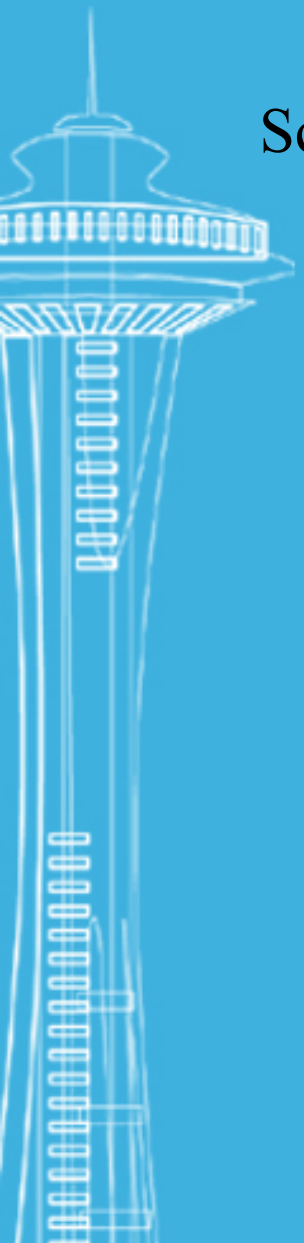
Why is this so hard? Would event correlation help?

Nope: Default ECS “Pairwise” circuit is “source unaware”

Don’t forget:

Un-manage all nodes/interfaces if you are uninterested in their status

Use netmon.noDiscover to prevent discovery of unwanted nodes



Script automates interface status events for multi-homed nodes

Disclaimer: Script is unsupported and serves as an example of potential automated actions based on NNM events. Script is unable to handle race conditions, has no error handling, and is not intended for use in highly distributed or scalable implementations.

OneIfHosts.ovpl – page 1 of 3

Cut here

```
#!/opt/OV/bin/Perl/bin/perl
#
# Written by Mike Peckar, Fognet Consulting, ov@fognet.com, unsupported
#
# 1. Create or recreate an external list of all NNM nodes that have
# only one interface. Use this list as the set of node sources for a
# copy of the OpenView interface down/up events. Invoke this script
# with the single argument "ALL" for this behavior.
# 2. Append a single node to the above mentioned list. Intended
# to be used as an automatic action with the node added event
# to keep the list updated when new nodes are discovered.
# For example, on NT/2000, syntax for the automatic action:
#    OVHIDESHELL C:\\OpenView\\bin\\OneIfHosts.ovpl $2
# Note this requires proper ovactiond trust. Create trusted command
# under $OV_CONF/TrustedCmds.conf directory or touch ALLOW_ALL
#
# Set the name of the file holding names of singly-homed hosts here:
```

Script automates interface status events for multi-homed nodes

OneIfHosts.ovpl – Page 2 of 3

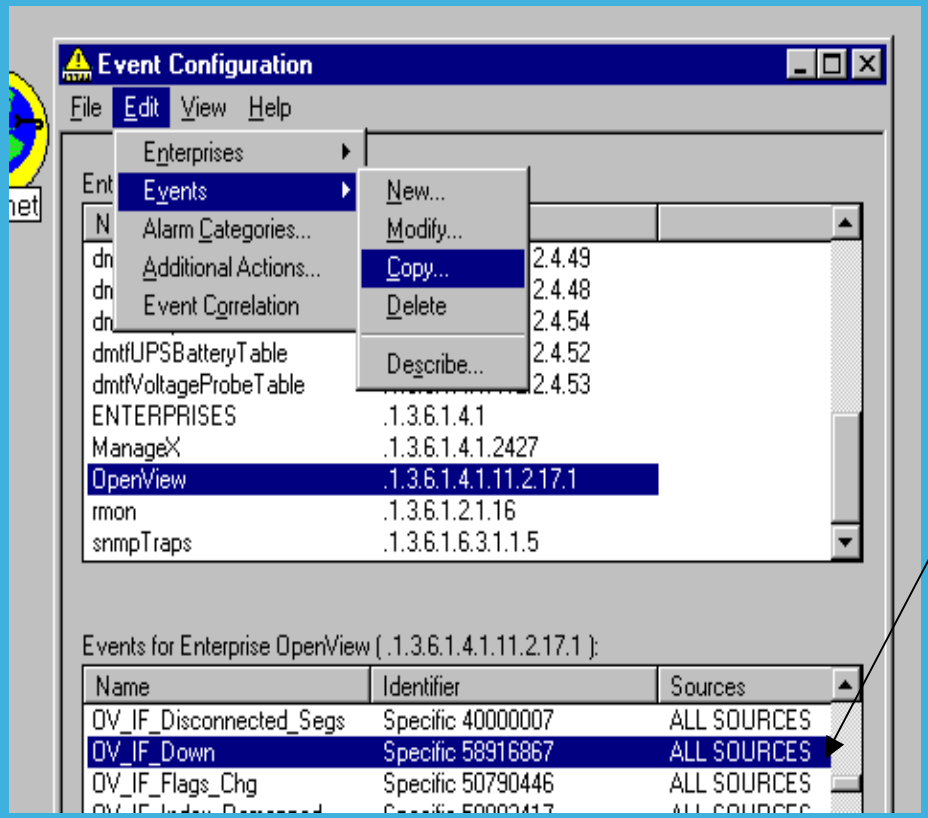
```
$hfile="$OV_CONF/OneIfHosts";
# For pre-6.2 NNM versions, define OV envvars, otherwise this works:
use OVvars;
#usage:
$NAME=$0; $NAME =~ s/.*\///; $NAME =~ s/.*\\//;
if ( scalar(@ARGV) != 1 ) {
    printf ("\nUsage 1: $NAME ALL\n");
    printf ("         Dump all singly-homed node names to $hfile\n");
    printf ("\nUsage 2: $NAME <Node Name>\n");
    printf ("         If singly-homed, append node name to $hfile\n");
    exit 10
}
#list all nodes with just one interface:
if ( $ARGV[0] =~ /^[aA][lL][lL]$/ ) {
   $cmd="$OV_BIN/ovobjprint -a \"Selection Name\" \"TopM Interface
     Count\"=1";
   open( IN, "$cmd |");
   while (<IN>) {
      $matches .= "$1\n" if m/^\s*\d*\s+["](.*)["]$/;
   }
   $matches =~ s/^(.*).$/$1/s;
   open ( OUT, ">$hfile");
   print OUT $matches;
```

Script automates interface status events for multi-homed nodes

OneIfHosts.ovpl – Page 3 of 3

```
} else {
#find number of interfaces for $ARGV[0] and if 1, write it to $hfile:
   $node = $ARGV[0];
   $cmd="$OV_BIN/ovobjprint -a \"TopM Interface Count\" \"Selection
     Name\"=$node";
   open( IN, "$cmd |");
   while (<IN>) {
      ($f1,$f2) = split(' ', $_);
      if ( $f1 =~ /^\d*$/ && $f2 =~ "1" ) {
         open ( OUT, ">>$hfile");
         print OUT "\n$node";
      }
   }
}
#in either case, signal NNM event subsystem to re-read $hfile..
$cmd="$OV_BIN/xnmevents -event" or die "Problem signaling xnmevents";
open( IN, "$cmd |");
#end of file
```

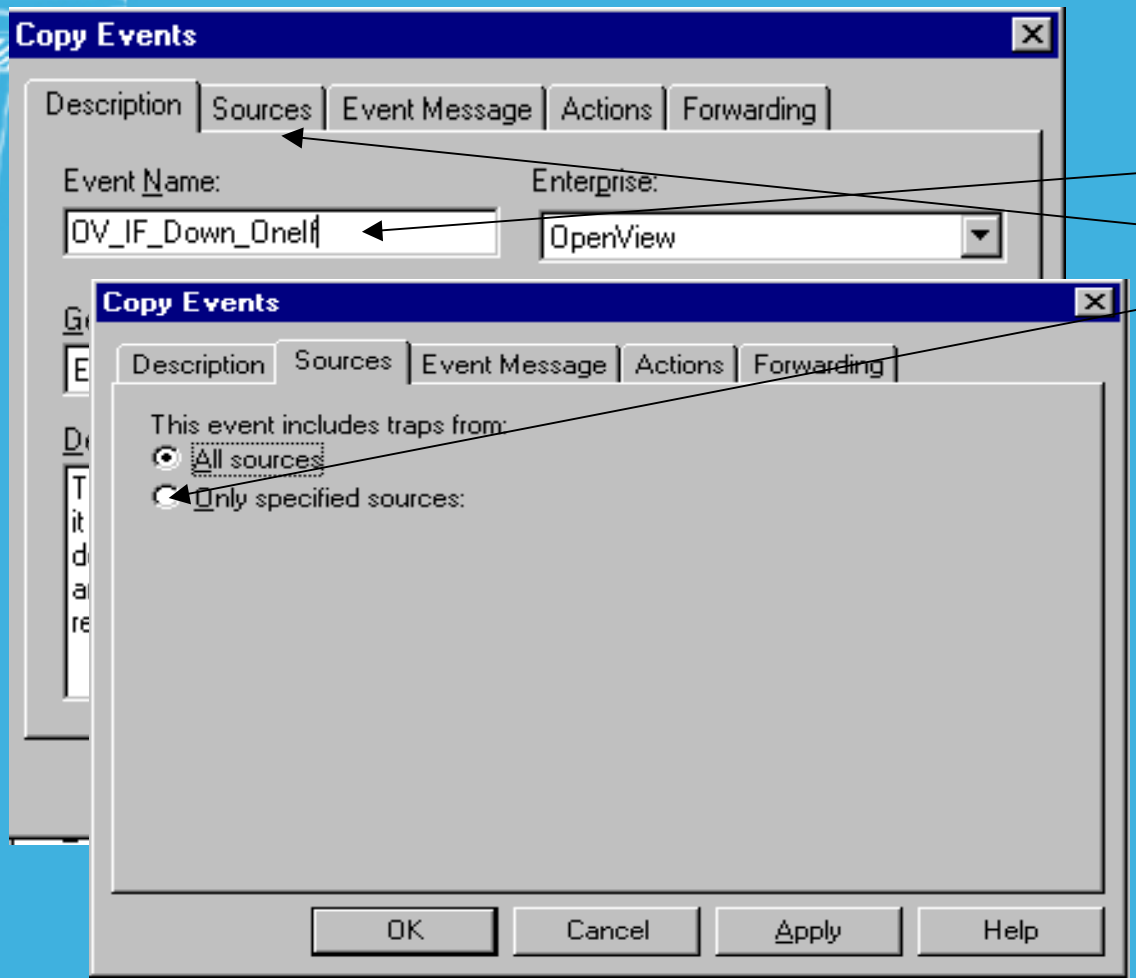
Script automates interface status events for multi-homed nodes



Steps:

1. Open event configuration
2. Select OV_IF_Down
3. Edit – Events – Copy

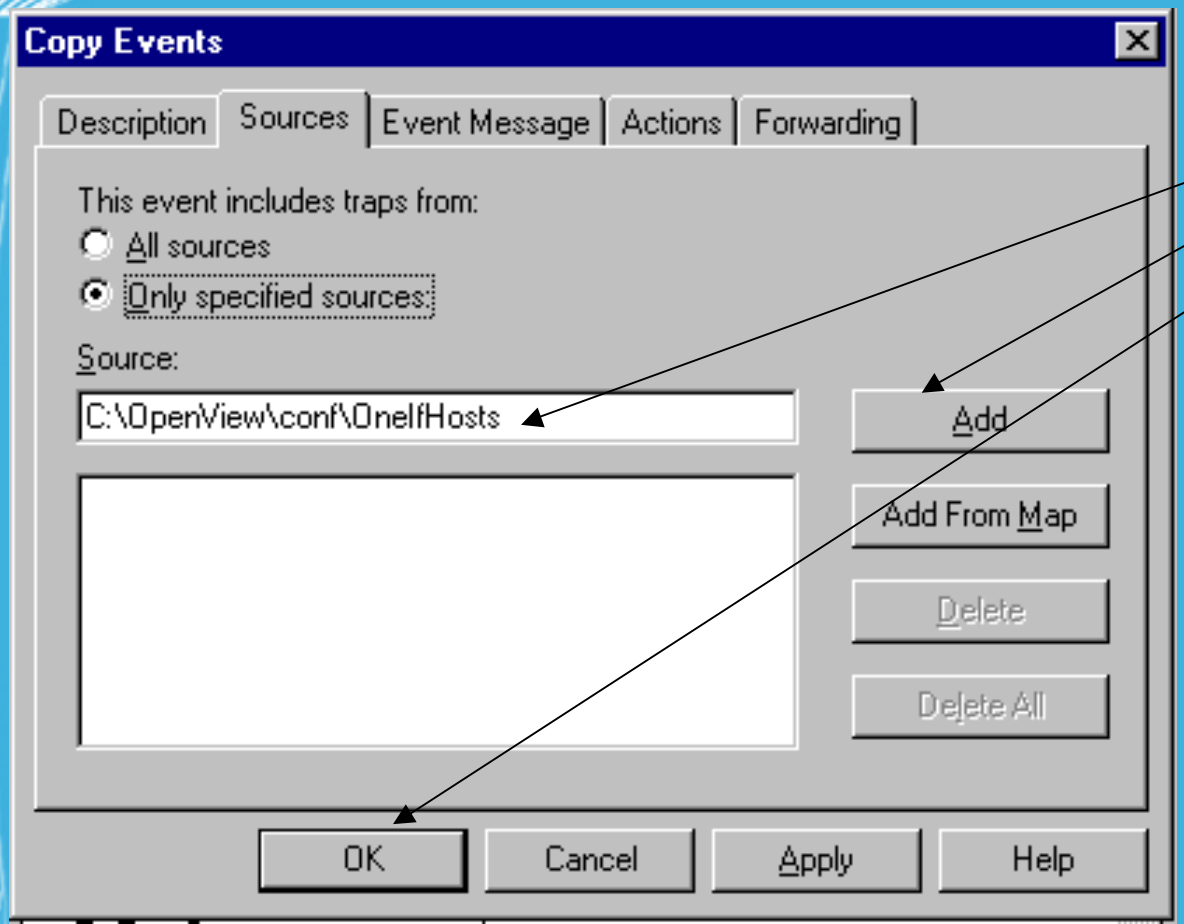
Script automates interface status events for multi-homed nodes



Steps:

4. Name it uniquely
5. Select Sources Tab
6. Specify Sources

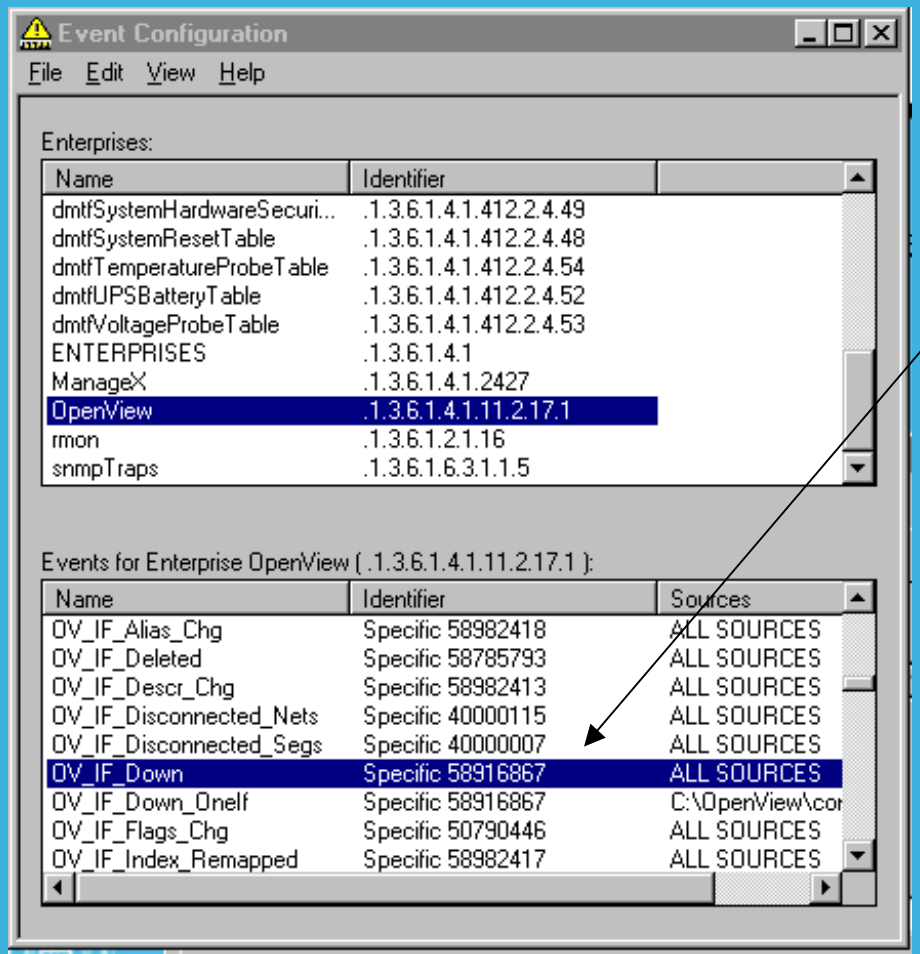
Script automates interface status events for multi-homed nodes



Steps:

7. Use an external file
8. Click “Add”
9. Click “OK”

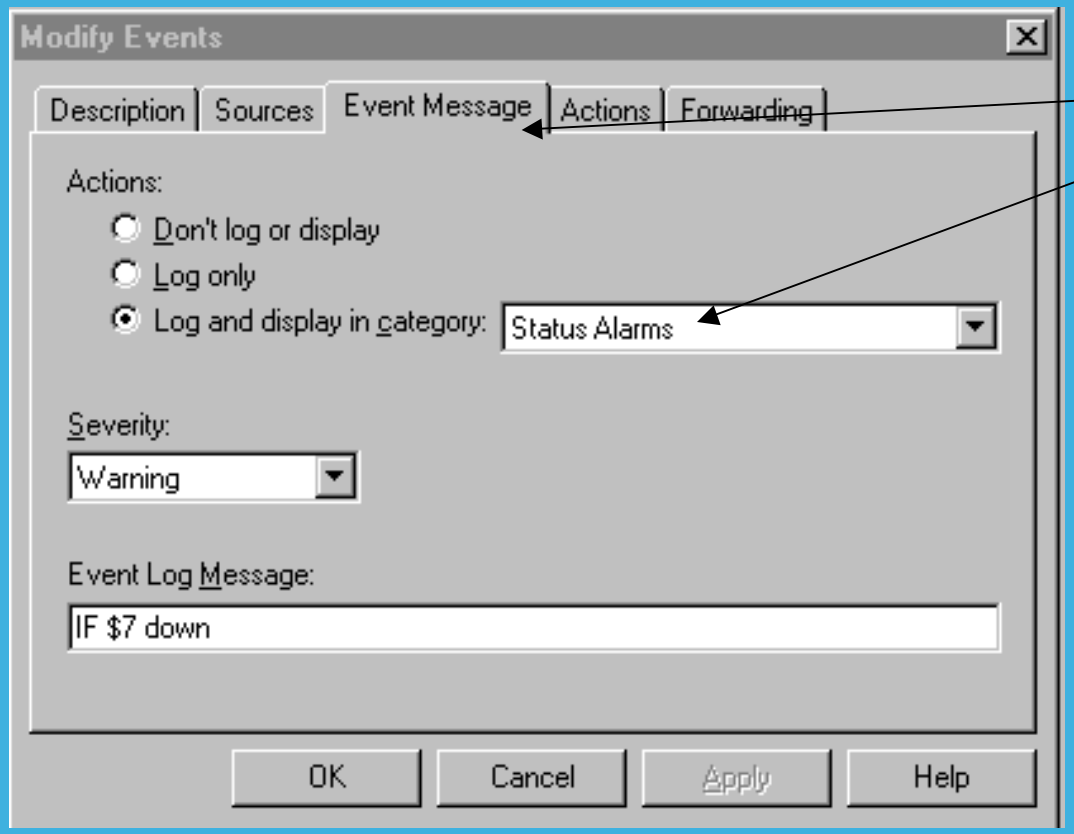
Script automates interface status events for multi-homed nodes



Steps:

10. Select Original OV_IF_Down event
11. Edit – Modify Event

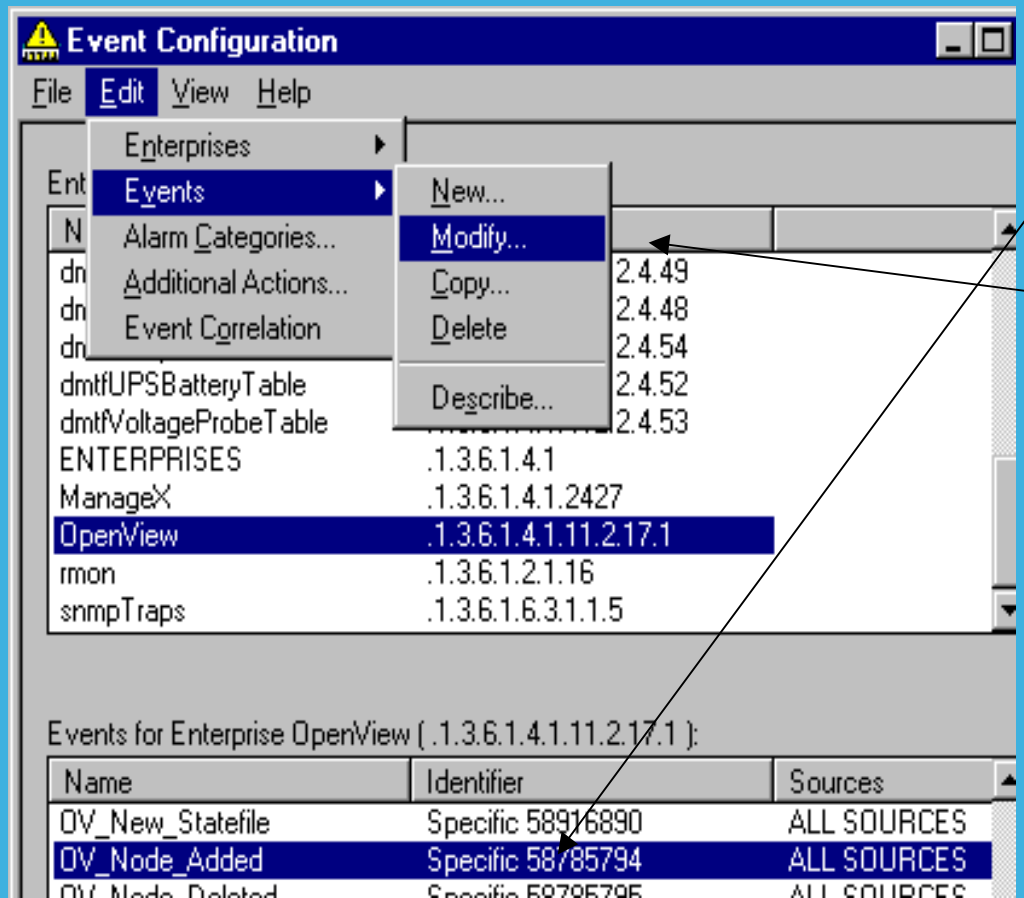
Script automates interface status events for multi-homed nodes



Steps:

12. Select Event Msg tab
13. Change from Log Only to Log Status
14. Click “OK”
15. Repeat steps 2-14 for OV_IF_Up Event

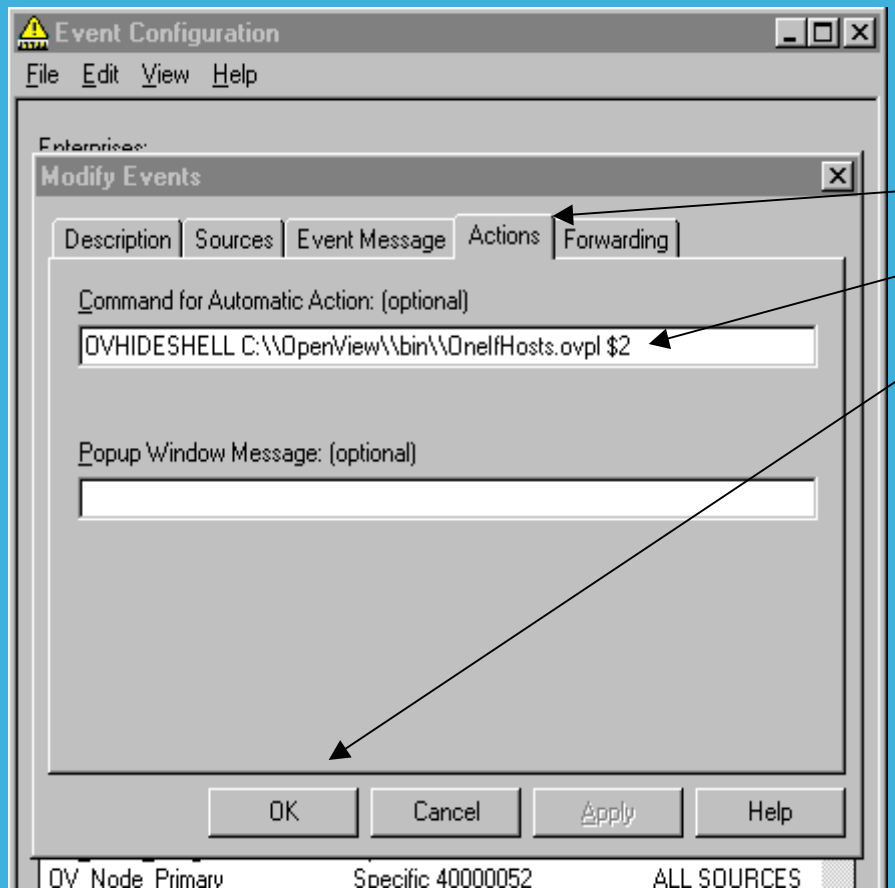
Script automates interface status events for multi-homed nodes



Steps:

16. Select Node Added event
17. Edit – Events – Modify

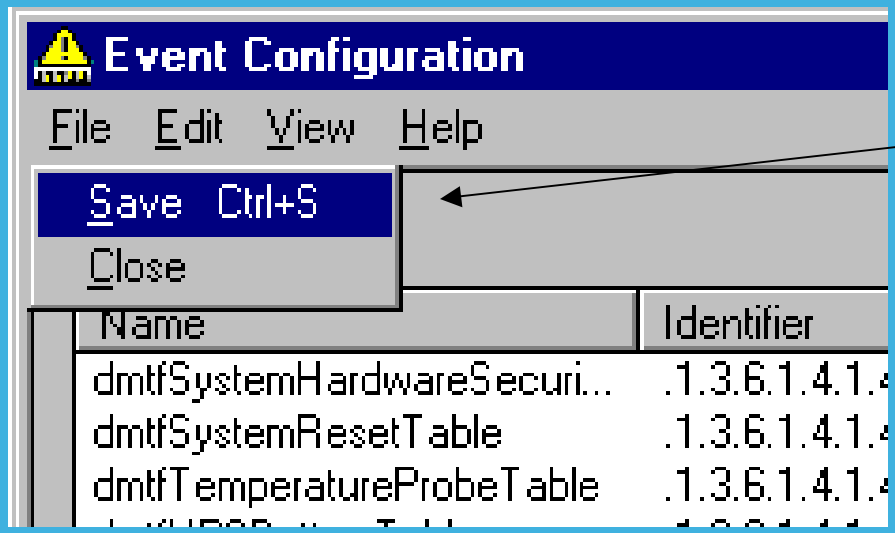
Script automates interface status events for multi-homed nodes



Steps:

18. Select “Actions”
19. Add Call to OneIfHosts
20. “OK”
21. Add OneIfHosts script to “trusted commands” (see man/ref pages for trustedCmds.conf)

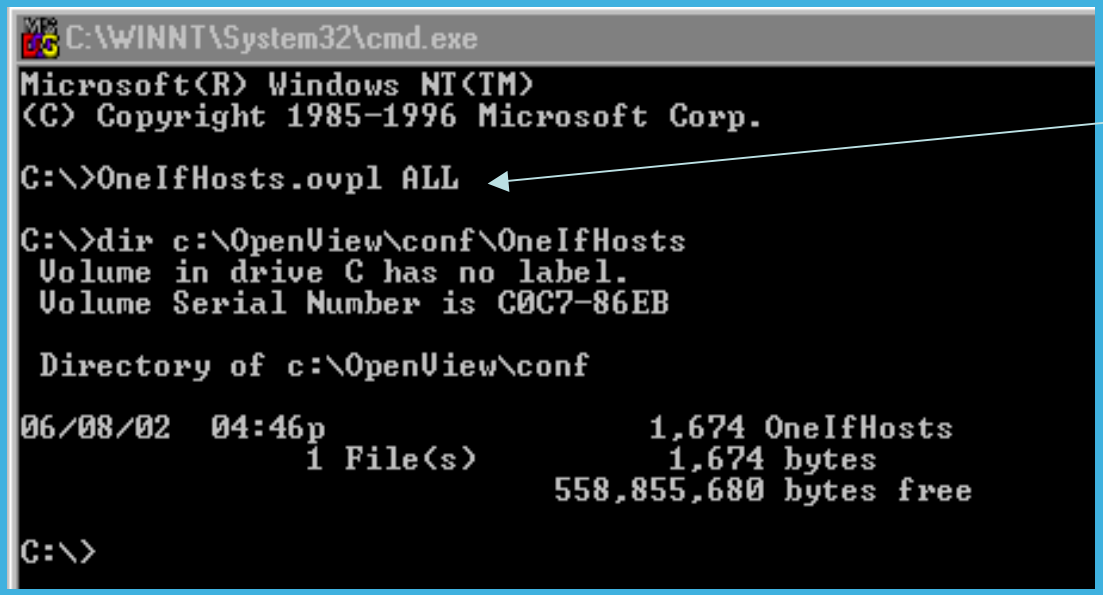
Script automates interface status events for multi-homed nodes



Steps:

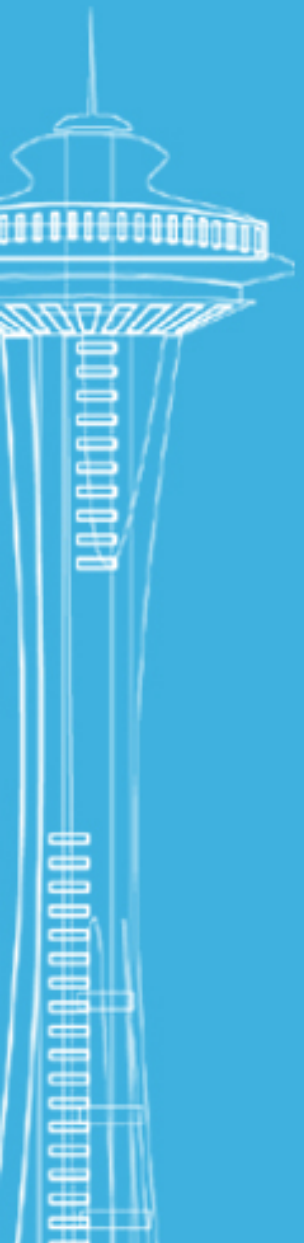
22. File – Save

Script automates interface status events for multi-homed nodes



Steps:

23.. Run script with ALL option to populate sources file with singly-homed hosts



Fin

Questions, time permitting…