NMIS8 Device Modeling Training

June 2013
Keith Sinclair, keiths@opmantek.com

Agenda (2.5 hours)

Time	Session
10 mins	NMIS Models Overview
10 mins	Decoding a MIB
10 mins	Adding a New Device
15 mins	Adding a New Metric to Node Health
30 mins	Adding a ‘system’ Section
25 mins	Adding a ‘systemHealth’ Section
15 mins	Adding Thresholding
5 mins	Running and Troubleshooting

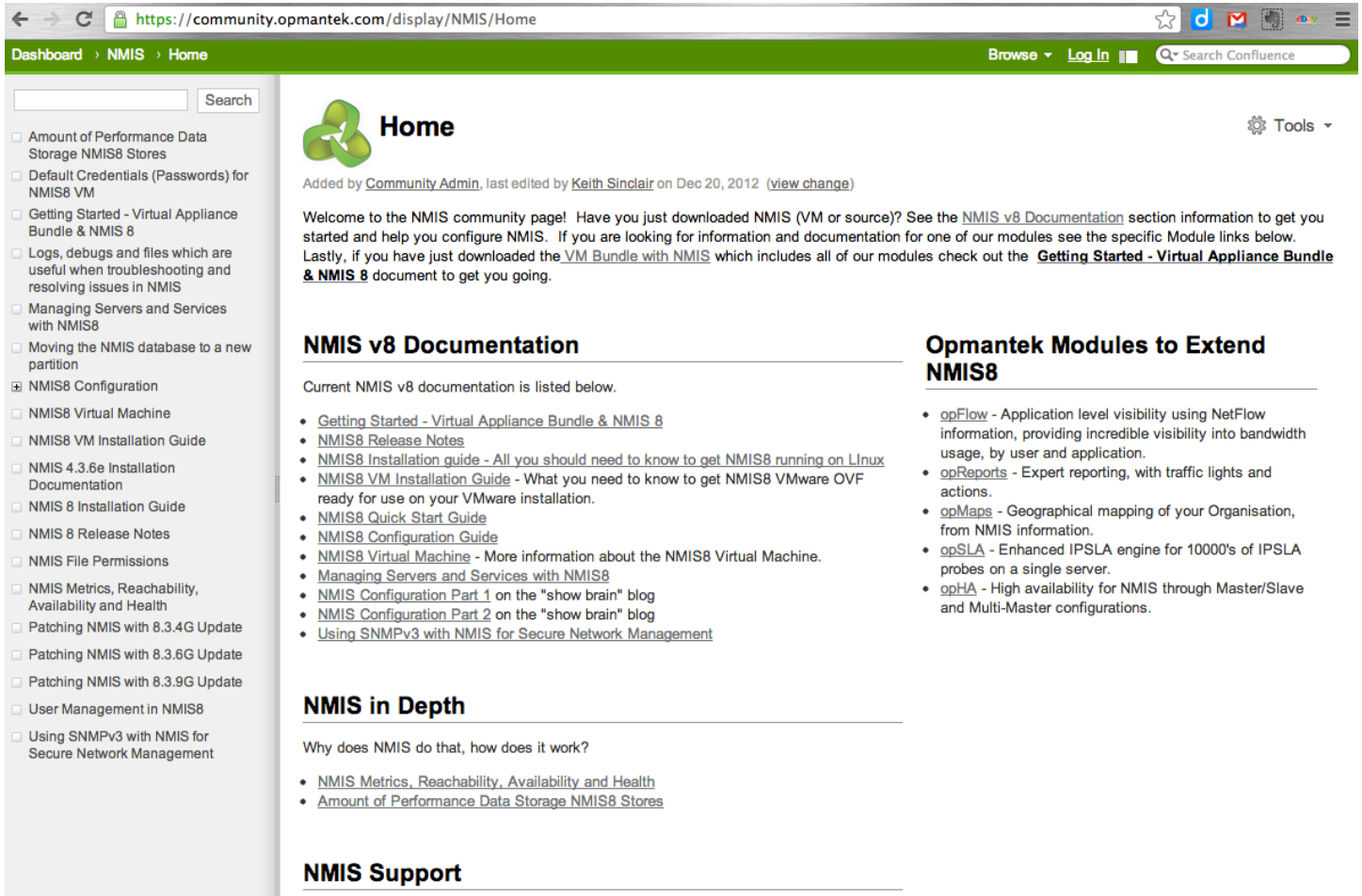
Agenda (1 hour)

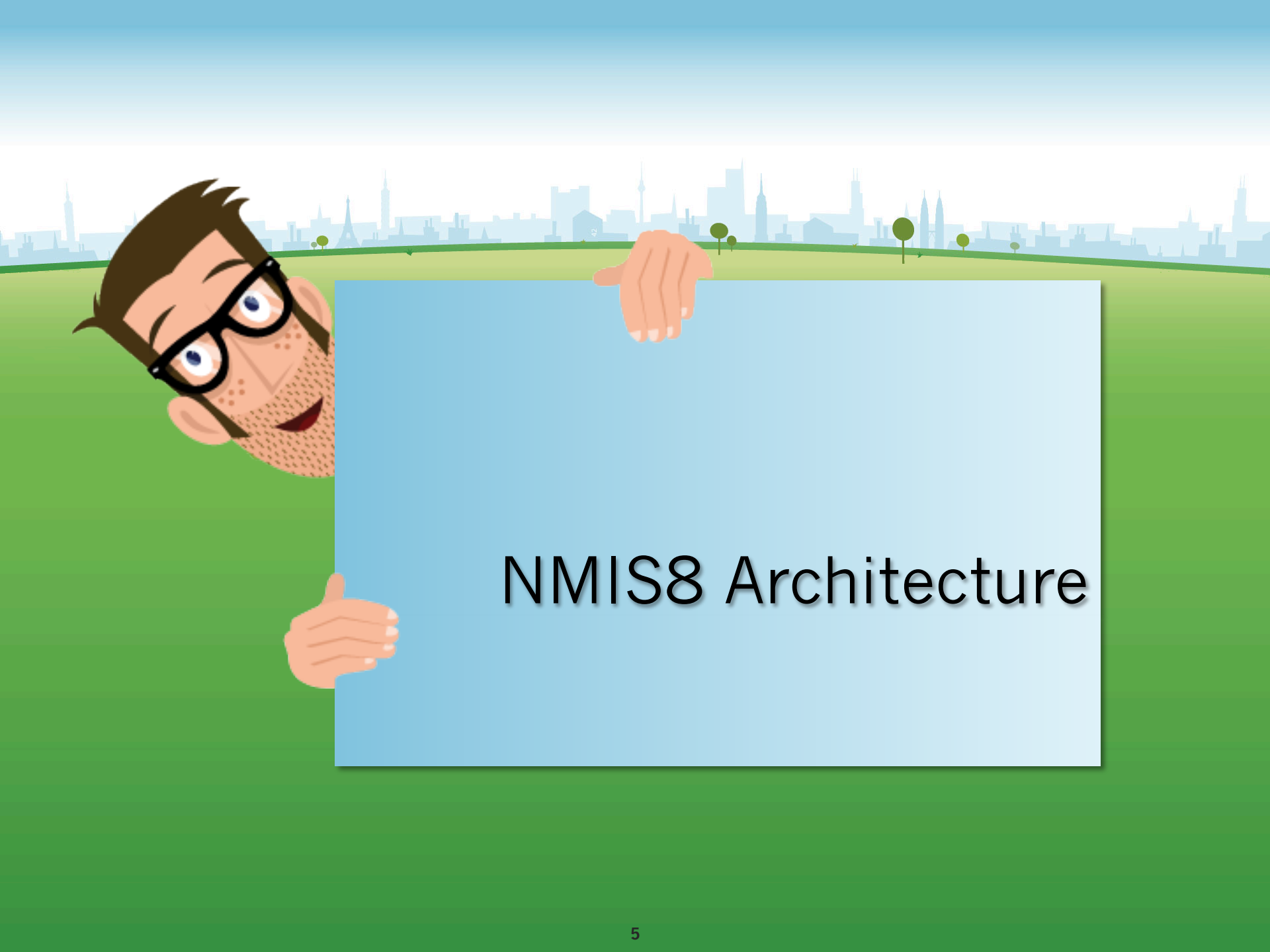
- NMIS Models Overview
- Decoding a MIB
- Adding a New Device
- Adding a New Metric to Node Health
- Adding a ‘system’ Section
- Adding a ‘systemHealth’ Section
- Adding Thresholding
- Running and Troubleshooting

Opmantek Community Wiki

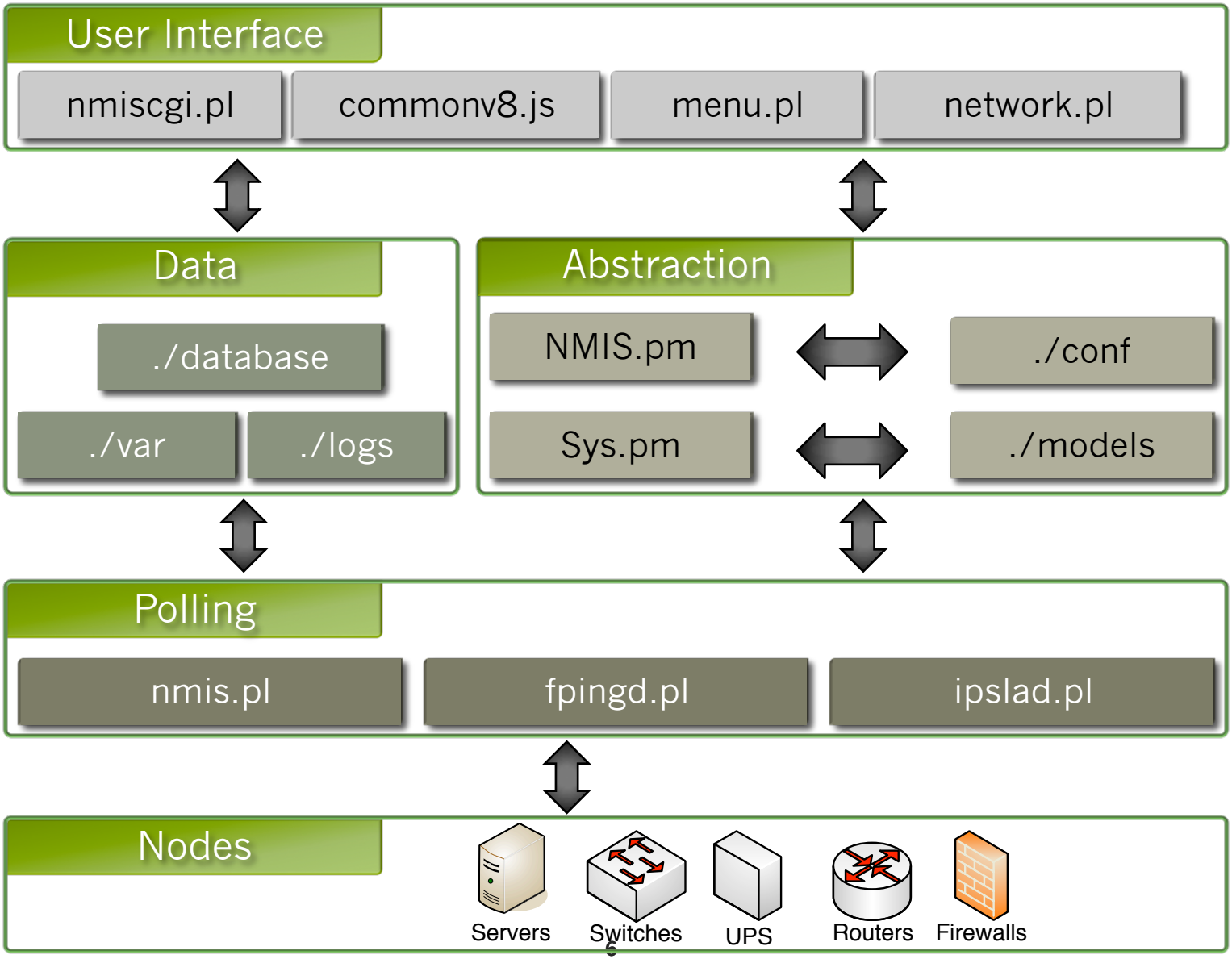
- Access all available documentation at the Opmantek Community Wiki. https://community.opmantek.com

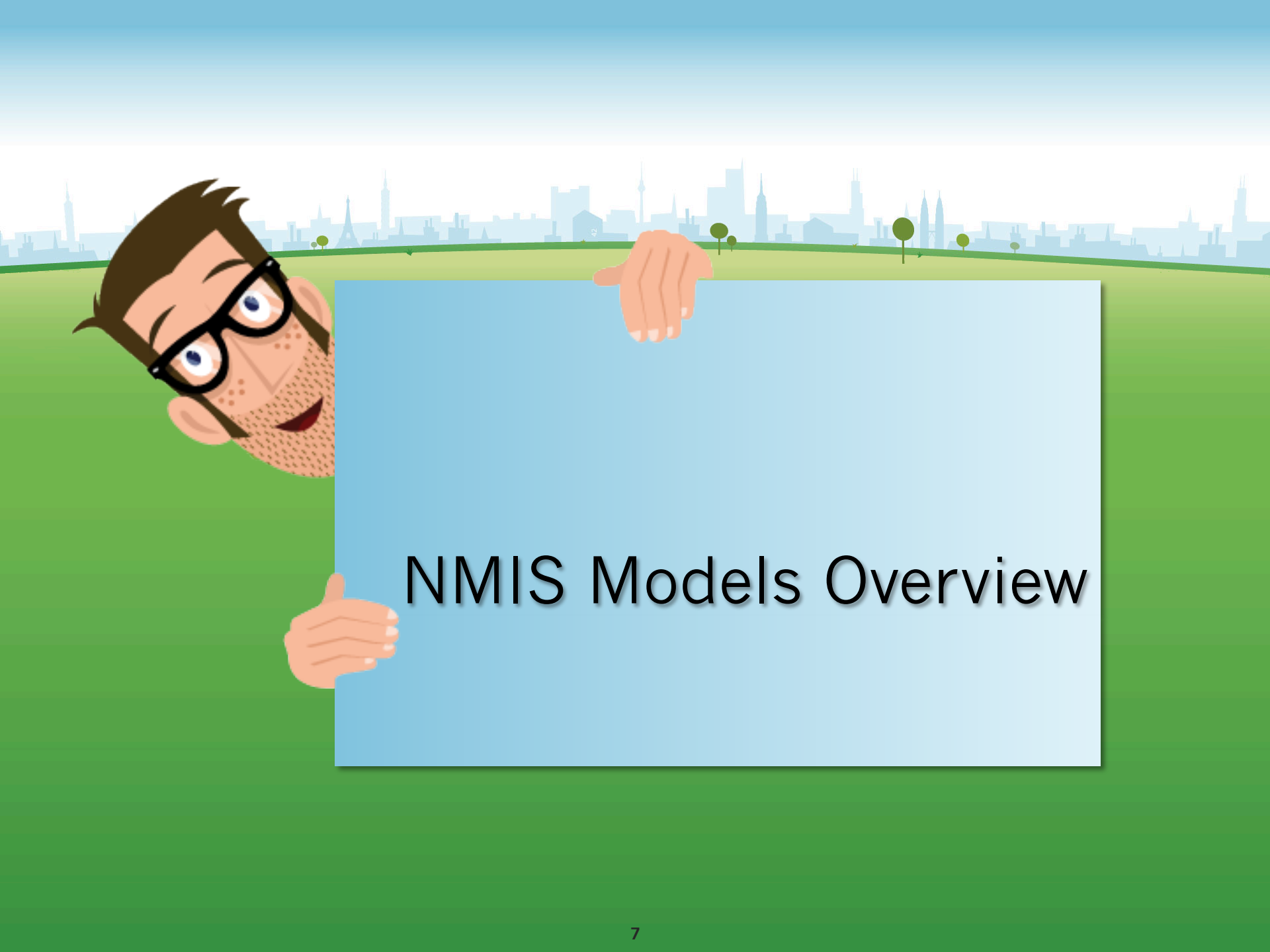
- Register @ https://opmantek.com "Join Community" top right.




NMIS8 Architecture

NMIS8 Architecture




NMIS Models Overview

Types of Model Files

File Type	Description
Model.nmis	The model auto-discovery rules to determine which model to use for which device.
Common-*.nmis	Model sections used by many different models
Graph-*.nmis	Graph definitions for each graph to be generated
Model-*.nmis	Device models which pull together related elements at runtime.
Enterprise.nmis	Part of the configuration files, contains the SNMP OID mappings to determine the vendors (or OEM).
nodename-node.nmis nodename-view.nmis	For each node a node and view file are generated. The node file cached information from the SNMP MIBS and other derived data. The view file is data to be displayed when presenting the User Interface.

Modeling Process

1. **Collect From Device:**
   - sysDescr
   - sysObjectId
2. **Determine the Vendor:** Compare the sysObjectID to the Enterprises defined in Enterprise.nmis
3. **Auto-discovery model:** Using the Vendor name and sysDescr compare to the model definitions in Model.nmis
4. **Load the model:** Use the model result from Model.nmis
5. **Load the data:** Use this model to collect device specific information from the device or to load the cached data from nmis8/var.

Modeling Process 1

1. **Collect From Device:**
   - sysDescr
   - sysObjectId
2. **Determine the Vendor:** Compare the sysObjectID to the Enterprises defined in Enterprise.nmis
3. **Auto-discovery model:** Using the Vendor name and sysDescr compare to the model definitions in Model.nmis

1.

```
'sysDescr' => 'Hardware: Intel64 Family 6
Model 15 Stepping 6 AT/AT COMPATIBLE -
Software: Windows Version 6.1 (Build 7600
Multiprocessor Free)'
'sysObjectID' =>
'1.3.6.1.4.1.311.1.1.3.1.1'
```

2.

```
'311' => {
 'OID' => '311',
 'Enterprise' => 'Microsoft'
},
```

3.

```
                'Microsoft' => {
                  'order' => {
                    '30' => {
    'Windows2000' => 'Windows 2000 Version
                      5.0'
                    },
                    '10' => {
     'Windows2003' => 'Windows Version 5.2'
                    },
                    '20' => {
     'Windows2008' => 'Windows Version 6.1'
                    }
                  }
                },
```

Modeling Process 2

4. **Load the model:** Use the model result from Model.nmis
   a. load any common models
   b. load instructions for main sections

5. **Load the data:** Use this model to collect device specific information from the device or to load the cached data from nmis8/var.

4a.

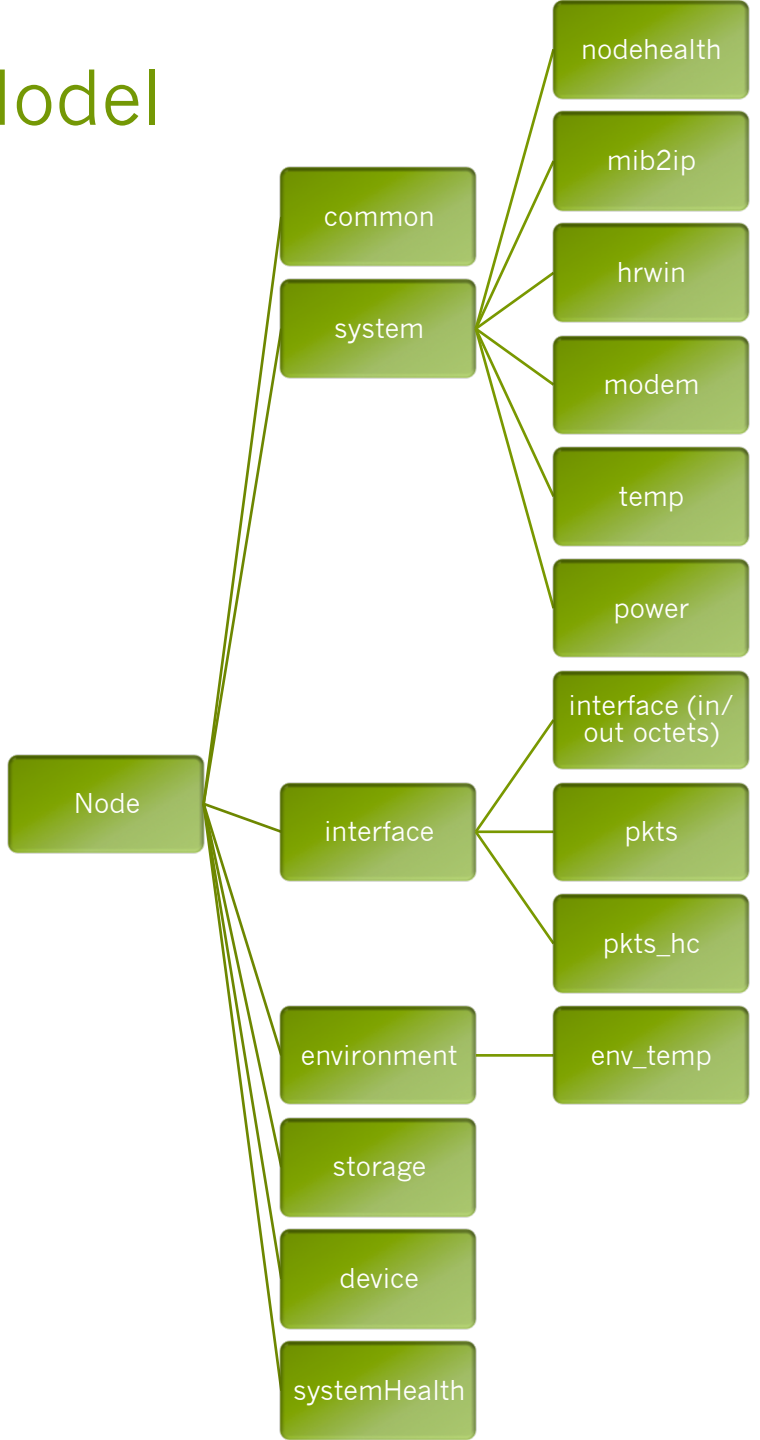
```
'-common-' => {
   'class' => {
     'database' => {
       'common-model' => 'database'
     },
--snip--
     'event' => {
       'common-model' => 'event'
     }
   }
},
```

4b.

```
'system' => {
        --snip--
},
'interface' => {
        --snip--
},
'storage' => {
        --snip--
},
'device' => {
        --snip--
},
```

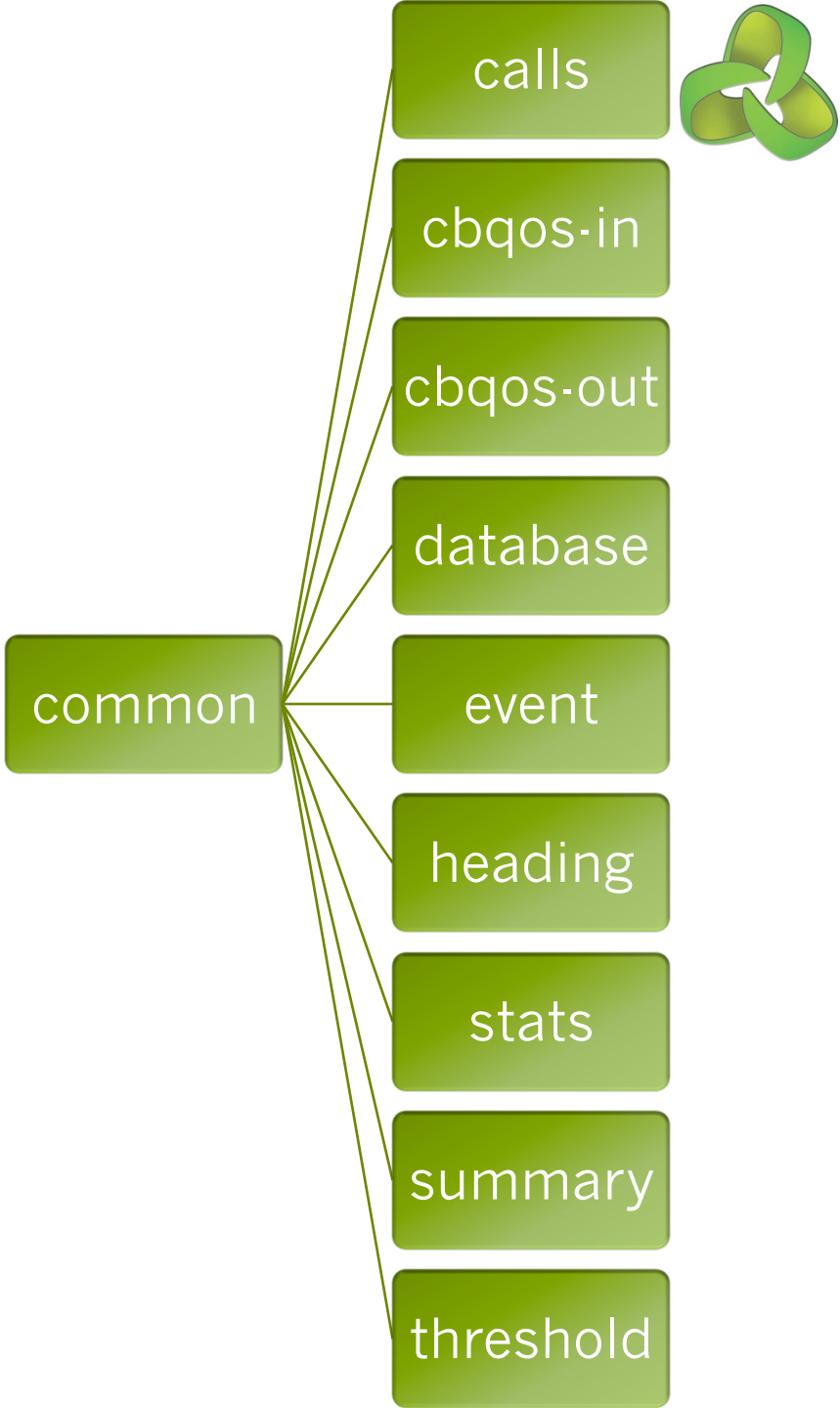
Main Structure of the Model

Section	Description
common	Defines what common models to load
system	The device specific concepts
interface	Everything related to the interfaces
environment	Currently temperature for indexed MIBS
storage	Disks and memory and virtual memory for servers
device	What things are connected to the device
systemHealth	Custom modeling for SNMP table structures



Common Models

Section	Description
calls	Monitoring for calls (less used today)
cbqos-in	Monitoring of the input traffic for Cisco Class Based QoS MIBS (HQoS)
cbqos-out	Monitoring of the output traffic for Cisco Class Based QoS MIBS (HQoS)
database	Definitions for the RRD database file names and folders
event	Event policy, determining what criticality what events are.
heading	Headings for various screens.
stats	RRD options for extracting stats from performance data.
summary	RRD options for generating summary data.
threshold	Thresholding policies for devices.



Process for Modelling a Device

- Have to collect information from a device
- The supported MIBS for the device
- SNMPWALK of the device is usually enough
- Review the MIB for the data elements to collect


Decoding a MIB

MIB Decoding example.

- Lets take this example, we are interested in the last three elements.

```
SNMPv2-SMI::enterprises.6302.2.1.1.1.0 = STRING: "Emerson Network Power"
SNMPv2-SMI::enterprises.6302.2.1.1.2.0 = STRING: "System Manager"
SNMPv2-SMI::enterprises.6302.2.1.1.3.0 = STRING: "1.6a 001"
SNMPv2-SMI::enterprises.6302.2.1.1.4.0 = ""
SNMPv2-SMI::enterprises.6302.2.1.2.1.0 = INTEGER: 6
SNMPv2-SMI::enterprises.6302.2.1.2.2.0 = INTEGER: 54014
SNMPv2-SMI::enterprises.6302.2.1.2.3.0 = INTEGER: 787097
SNMPv2-SMI::enterprises.6302.2.1.2.4.0 = INTEGER: 67
```

- So "SNMPv2-SMI::enterprises" is always ".1.3.6.1.4.1" which makes these three MIBS:
  - .1.3.6.1.4.1.6302.2.1.2.2.0
  - .1.3.6.1.4.1.6302.2.1.2.3.0
  - .1.3.6.1.4.1.6302.2.1.2.4.0
- Google them..... in this case no luck
- Google "1.3.6.1.4.1.6302" reveals "Emerson Energy Systems"
- A little more detective work, we find the ees-power.mib file.
- http://www.emersonnetworkpower.com/en-US/Brands/EnergySystems/Pages/ensys_SoftwareSupport.aspx

Download ees-power.mib

- We want to figure out what each element is, so we have to trace the tree through the ASN.1 content of the MIB (Management Information Base).

```
SNMPv2-SMI::enterprises.6302.2.1.2
```

```
ees OBJECT IDENTIFIER ::= { enterprises 6302 }
global OBJECT IDENTIFIER ::= { ees 2 }
powerMIB MODULE-IDENTITY
        LAST-UPDATED "200310140730Z"
        ORGANIZATION "
                Emerson Energy Systems (EES)"
        CONTACT-INFO "
                Emerson Energy Systems
                141 82 Stockholm
                Sweden"
        DESCRIPTION "
                Emerson Energy Systems (EES) Power MIB, revision B."
        ::= { global 1 }
ident OBJECT IDENTIFIER ::= { powerMIB 1 }
system OBJECT IDENTIFIER ::= { powerMIB 2 }
```

Now down the tree to the final targets

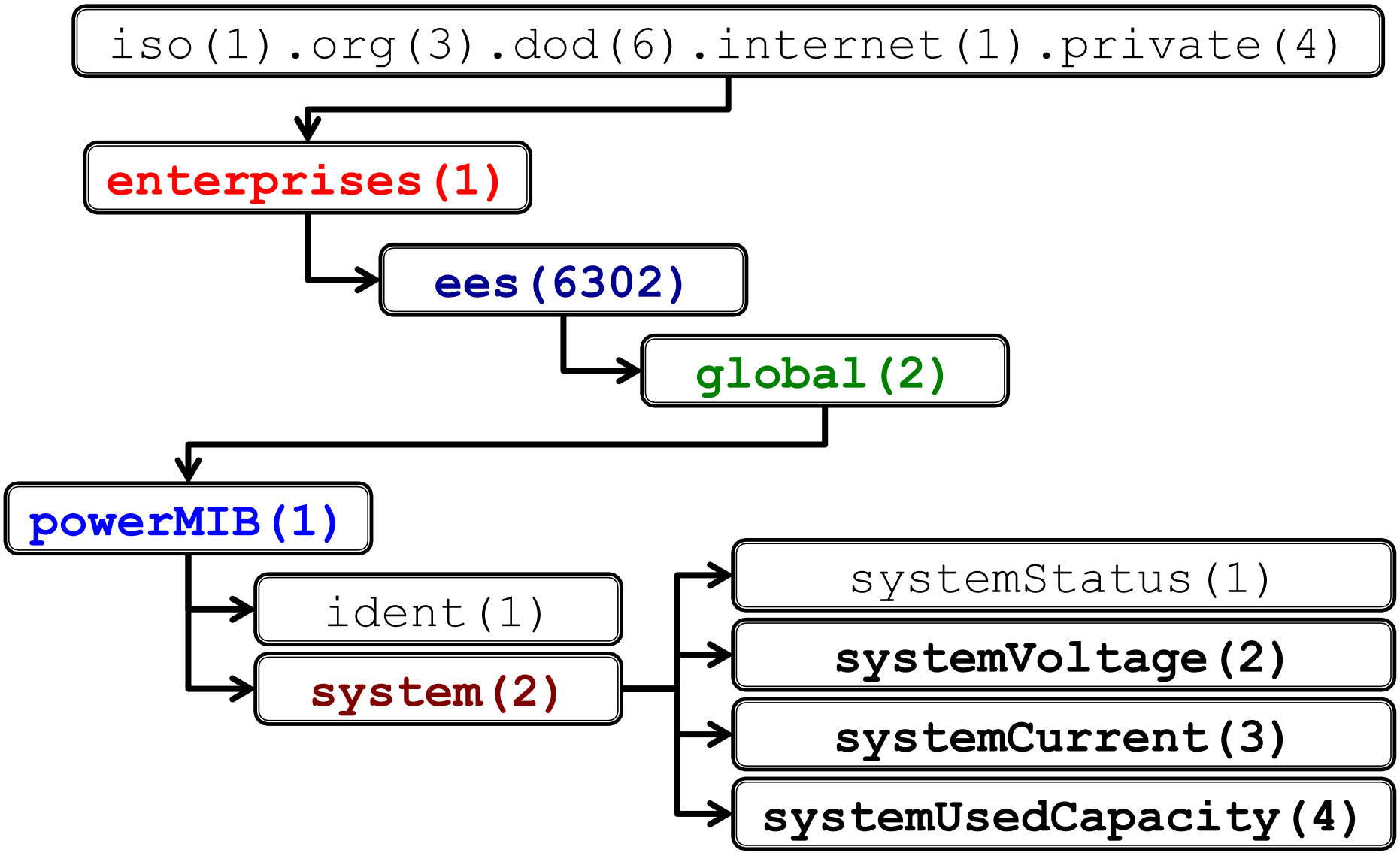
SNMPv2-SMI::enterprises.ees.global.powerMIB.system

```
system OBJECT IDENTIFIER ::= { powerMIB 2 }
systemVoltage OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION "
                System voltage, stored as mV, including positive or negative
                sign. The integer 2147483647 represents invalid value."
        ::= { system 2 }

systemCurrent OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION "
                System current, stored as mA, including positive or negative
                sign. The integer 2147483647 represents invalid value."
        ::= { system 3 }

systemUsedCapacity OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION "
                Used capacity, stored as % of the total capacity.
                The integer 2147483647 represents invalid value."
        ::= { system 4 }
```

The MIB tree for the three components




Adding a New Device

Steps to add a new device model

1. Get the sysDescr and sysObjectID
   - `"Device FooBar, Software Version"`
   - `1.3.6.1.4.1.424242.1.1`
2. Determine the Enterprise number and matching Vendor name
   - `"424242" => "Vendor Name"`
3. Add an entry to Enterprise.nmis
4. Add an entry to Model.nmis
5. Decide on the name for the new model
6. Copy an existing model, e.g. Model-Default.nmis
7. Edit and update the system section of the model

Steps and Examples

3. Add an entry to Enterprise.nmis

(/usr/local/nmis8/conf/Enterprise.nmis)

3.

```
'424242' => {
 'OID' => '424242',
 'Enterprise' => 'Vendor Name'
},
```

4. Add an entry to Model.nmis

4.

```
'Vendor Name' => {
 'order' => {
  '10' => {
   'ModelName' => 'Device FooBar'
  }
 }
},
```

7. Edit and update the system section of the model

7.

```
'system' => {
  'nodeType' => 'switch',
  'nodeVendor' => 'Vendor Name',
  'nodeModel' => 'ModelName',
  'rrd' => {
    'nodehealth' => {
      'snmp' => {
        'avgBusy5' => {
--snip--
```

A Real Example

3. Sun Microsystems in the Enterprise.nmis

(/usr/local/nmis8/conf/Enterprise.nmis)

3.

```
'42' => {
  'Enterprise' => 'Sun
Microsystems',
  'OID' => '42'
},
```

4. Sun Microsystems in Model.nmis

4.

```
'Sun Microsystems' => {
  'order' => {
    '10' => {
      'SunSolaris' => 'sol|Sun
SNMP|SunOS'
    }
  }
},
```

7. Sun Microsystems in system section of the model (Model-SunSolaris.nmis)

7.

```
 'system' => {
    'nodeType' => 'server',
    'nodeVendor' => 'Sun
Microsystems'
    'nodeModel' => 'SunSolaris',
--snip--
```

Use an existing model for a device

- It is common that a new product comes out which uses the same SNMP system as an existing product.
- NMIS needs to be told which model to use for that device.
- Sun Microsystems changed to use the Net-SNMP daemon a few years ago.

```
SNMPv2-MIB::sysDescr.0 = STRING: SunOS gsmolames1 5.10
Generic_142900-13 sun4v
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-
MIB::netSnmpAgentOIDs.3
```

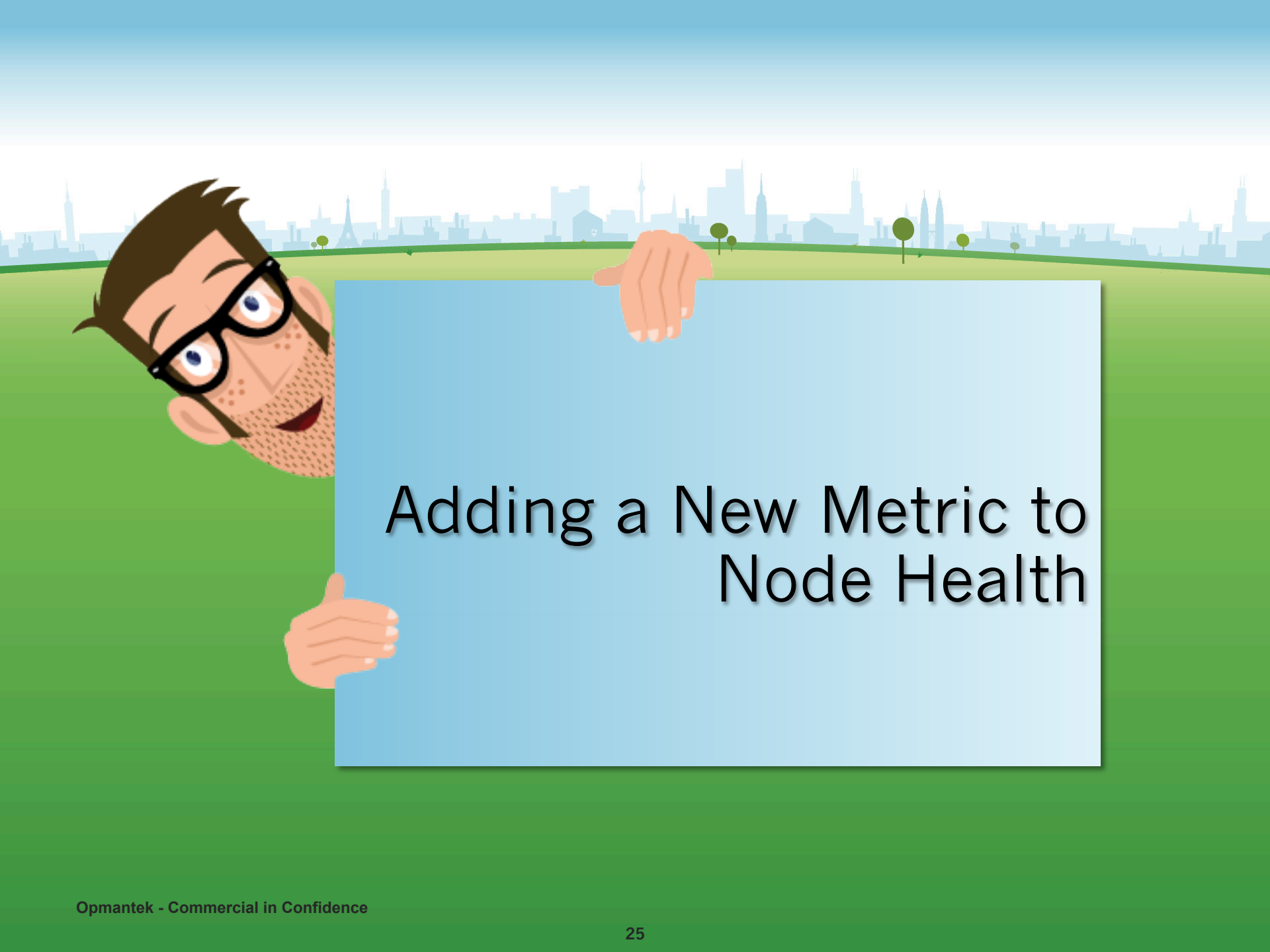
1. Collect the sysObjectID and sysDescr and look for existing vendors.
2. Update the Models.nmis with the pattern from sysDescr.
3. Done.

1.

```
'8072' => {
  'Enterprise' => 'net-snmp',
  'OID' => '8072'
},
```

2.

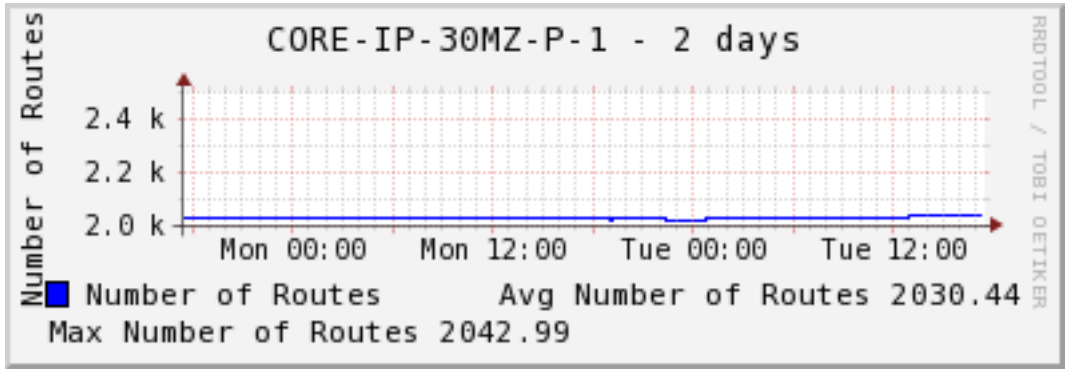
```
'net-snmp' => {
  'order' => {
    '10' => {
      'net-snmp' => 'Linux|SunOS|Darwin'
    }
  }
},
```


Adding a New Metric to
Node Health

Lets monitor the number of Routes in a device.

- The number of Routes seen by a router (layer 3 device)
- Looking at MIB dumps (SNMPWALK outputs) we can see:
  - IP-FORWARD-MIB::ipCidrRouteNumber.0 = Gauge32: 33554432
  - IP-FORWARD-MIB::inetCidrRouteNumber.0 = Gauge32: 7

- The OID's (Object ID) for these MIBS are:
  - ipCidrRouteNumber = "1.3.6.1.2.1.4.24.3"
  - inetCidrRouteNumber = "1.3.6.1.2.1.4.24.6"



Investigation of Support

- Detective work, collect information and review what is available, look for patterns.

- Looking through a few devices MIB Dumps (SNMPWALK outputs), we can see it is widely supported by Cisco, Alcatel and many other vendors.
  - Alcatel-OmniSwitch685048.mib:IP-FORWARD-MIB::ipCidrRouteNumber.0 = Gauge32: 5
  - C3745-12-3-11T.mib:ip.ipForward.ipCidrRouteNumber.0 = Gauge32: 18
  - Cisco-800-meatball.mib:IP-FORWARD-MIB::ipCidrRouteNumber.0 = Gauge32: 7
  - Cisco-C3750-12.2.55.SE4.mib:IP-FORWARD-MIB::ipCidrRouteNumber.0 = Gauge32: 432
  - asr1004.mib:IP-FORWARD-MIB::ipCidrRouteNumber.0 = Gauge32: 1
  - c7200-12-3-8T-2.mib:ip.ipForward.ipCidrRouteNumber.0 = Gauge32: 19
  - cisco1800-asgard.mib:IP-FORWARD-MIB::ipCidrRouteNumber.0 = Gauge32: 4
  - emea_3620.mib:ip.ipForward.ipCidrRouteNumber.0 = Gauge32: 6
  - iou-bne1-names.mib:ip.ipForward.ipCidrRouteNumber.0 = Gauge32: 28
  - iox-gsr.mib:ip.ipForward.ipCidrRouteNumber.0 = Gauge32: 5
  - mds.mib:IP-FORWARD-MIB::ipCidrRouteNumber.0 = Gauge32: 2
  - n5k.mib:IP-FORWARD-MIB::ipCidrRouteNumber.0 = Gauge32: 33554432
  - r19-7505.mib:ip.ipForward.ipCidrRouteNumber.0 = Gauge32: 4
  - rtp3640.mib:ip.ipForward.ipCidrRouteNumber.0 = Gauge32: 3
  - sj-nettools-3640-1.mib:ip.ipForward.ipCidrRouteNumber.0 = Gauge32: 15
  - sjc5-gb2.mib:IP-FORWARD-MIB::ipCidrRouteNumber.0 = Gauge32: 774
  - ucs-fabric.mib:IP-FORWARD-MIB::ipCidrRouteNumber.0 = Gauge32: 33554432

Let's add it to Cisco IOS Routers First

- Implementation Decisions:
  - We will call the new metric "routenumber"
  - The MIB name is ipCidrRouteNumber
  - The OID is 1.3.6.1.2.1.4.24.3
  - The graph will be called routenumber
- We will need to make the following changes:
  1. Add it to the Model-CiscoRouter.nmis model to start collecting.
  2. Add an entry to: Common-heading.nmis
  3. Add an entry to nmis_oids.nmis
  4. Create a graph to view it: Graph-routenumber.nmis

Modelling routenumber 1

1. Add it to the Model-CiscoRouter.nmis model to start collecting.
2. Add an entry to: Common-heading.nmis
3. Add an entry to nmis_oids.nmis

1.

```
  'system' => {
    'nodeModel' => 'CiscoRouter',
    'nodeType' => 'router',
    'rrd' => {
      'nodehealth' => {
        'snmp' => {
          'avgBusy5' => {
--snip--
          'RouteNumber' => {
            'oid' => 'ipCidrRouteNumber'
          }
        },
        'threshold' => 'cpu,mem-proc',
        'graphtype' => 'buffer,cpu,mem-
io,mem-proc,mem-router,routenumber'
      },
```

2.

```
'heading' => {
  'graphtype' => {
     --snip--
     'routenumber' => 'Number of Routes'
  }
}
```

3.

```
"ipCidrRouteNumber"   "1.3.6.1.2.1.4.24.3"
```

Modelling routenumber 1A

1. Instead of having to modify the MIB, you can put the OID straight into the Model.

1.

```
'system' => {
  'nodeModel' => 'CiscoRouter',
  'nodeType' => 'router',
  'rrd' => {
    'nodehealth' => {
      'snmp' => {
        'avgBusy5' => {
--snip--
        'RouteNumber' => {
          'snmpObject' => 'ipCidrRouteNumber'
          'oid' => '1.3.6.1.2.1.4.24.3'
        }
      },
--snip--
```

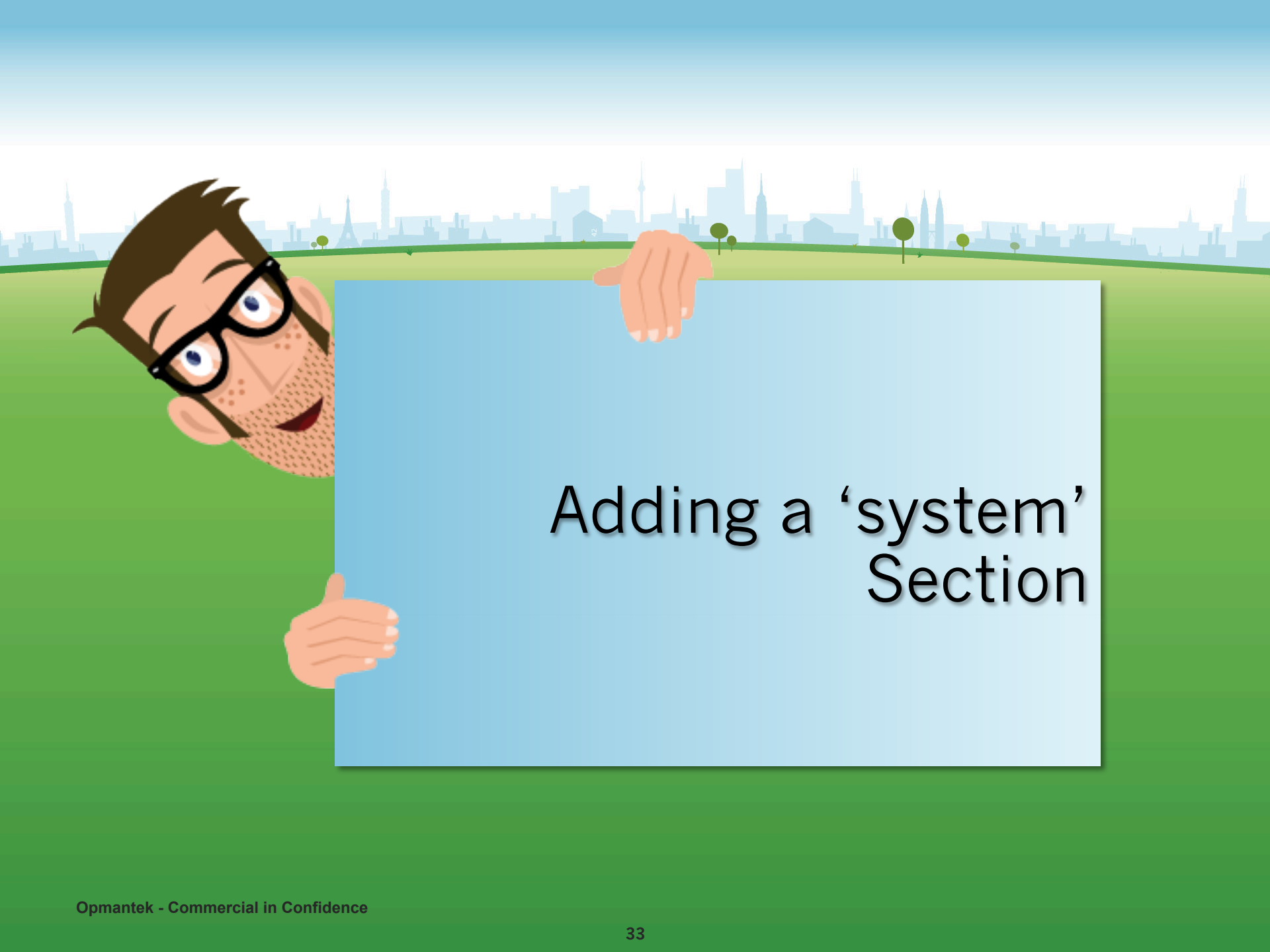
Modelling routenumber 2

4. Create a graph to view it: Graph-routenumber.nmis
   a. Update the title.
   b. Define a vertical label.
   c. Define a standard graph (this is RRD RPN expressions)
   d. Define a small graph.

```
'title' => {
  'standard' => '$node - $length from
$datestamp_start to $datestamp_end',
  'short' => '$node - $length'
},
'vlabel' => {
  'standard' => 'Number of Routes'
},
'option' => {
  'standard' => [
    'DEF:routes=$database:RouteNumber:AVERAGE',
    'LINE1:routes#0000ff:Number of Routes',
    'GPRINT:routes:AVERAGE:Avg Number of Routes
%1.2lf',
    'GPRINT:routes:MAX:Max Number of Routes
%1.2lf'
  ],
  'small' => [
    'DEF:routes=$database:RouteNumber:AVERAGE',
    'LINE1:routes#0000ff:Number of Routes',
    'GPRINT:routes:AVERAGE:Avg Number of Routes
%1.2lf',
    'GPRINT:routes:MAX:Max Number of Routes
%1.2lf'
  ]
}
```

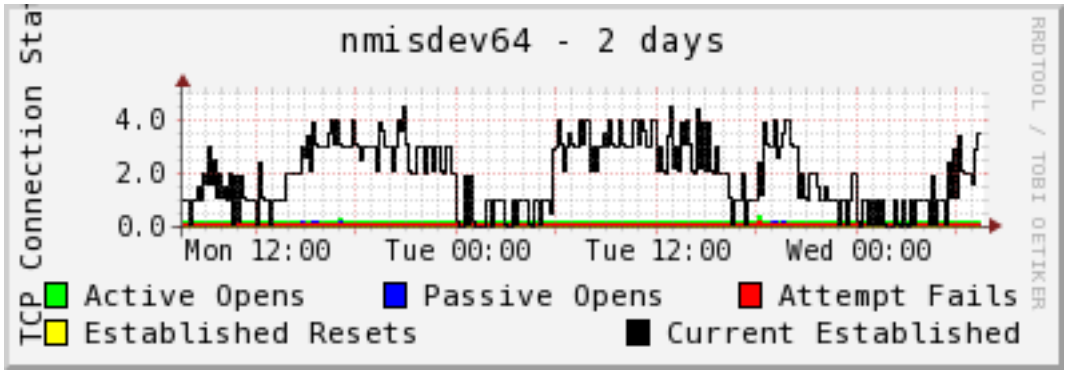
RRDTool Primer - Read This!

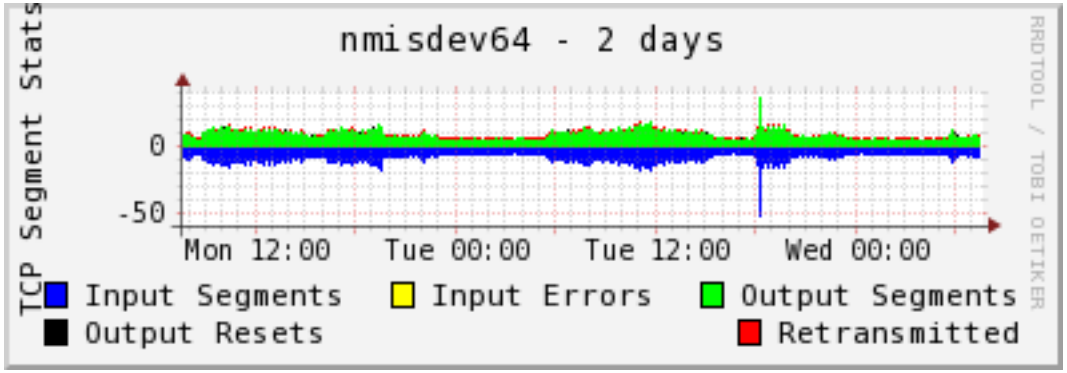
- http://oss.oetiker.ch/rrdtool/tut/cdeftutorial.en.html
- http://oss.oetiker.ch/rrdtool/doc/rrdgraph_rpn.en.html


Adding a ‘system’
Section

Monitoring a new concept in the device.

- Figure out the MIBS you need
- Look for a simple pattern.
- Use the ‘system’ section for flat objects and use the systemHealth section for table constructs





The system Section

Added in NMIS 8.1.1, supports new collections without coding

- “tcp” is included in the Model-net-snmp.nmis file.
- Implementation Decisions:
  - What data to collect?
  - The MIB name is **tcp**
  - The collection will be called tcp
- We will need to make the following changes:
  1. Add the tcp section to the system section of the model
  2. Add any required MIBS to nmis_oids.nmis (optional)
  3. Add an entry to: Common-heading.nmis
  4. Add an entry to: Common-database.nmis
  5. Create any required graphs to view data: Graph-tcp-conn.nmis and Graph-tcp-segs.nmis

What does the “tcp” MIB look like?

RFC1213-MIB::tcpActiveOpens.0 = Counter32: 487232

RFC1213-MIB::tcpPassiveOpens.0 = Counter32: 110120

RFC1213-MIB::tcpAttemptFails.0 = Counter32: 99301

RFC1213-MIB::tcpEstabResets.0 = Counter32: 75577

RFC1213-MIB::**tcpCurrEstab**.0 = **Gauge32**: 72

RFC1213-MIB::tcpInSegs.0 = Counter32: 12879179

RFC1213-MIB::tcpOutSegs.0 = Counter32: 11516662

RFC1213-MIB::tcpRetransSegs.0 = Counter32: 428664

RFC1213-MIB::tcpInErrs.0 = Counter32: 6

RFC1213-MIB::tcpOutRsts.0 = Counter32: 69835

Modelling tcp 1

1. Add the tcp, this goes under the rrd section.
   a. Collect some performance data.
   b. `graphtype` what graphs will be created from this information.
   c. `snmp` what to collect,
      ① `oid` can be something nmis_oids.nmis or an OID
      ② `option` is the data a counter or gauge, and what are the lower and upper limits

```
'system' => {
  --snip--
  'rrd' => {
    --snip—
    'tcp' => {
      'graphtype' => 'tcp-conn,tcp-segs',
      'snmp' => {
        'tcpActiveOpens' => {
          'oid' => 'tcpActiveOpens',
          'option' => 'counter,0:U'
        },
        'tcpPassiveOpens' => {
          'oid' => 'tcpPassiveOpens',
          'option' => 'counter,0:U'
        },
        --snip—
        'tcpCurrEstab' => {
          'oid' => 'tcpCurrEstab',
          'option' => 'gauge,0:U'
        },
        --snip—
        'tcpOutRsts' => {
          'oid' => 'tcpOutRsts',
          'option' => 'counter,0:U'
        }
      }
    }
  }
}
```

Modelling tcp 2

2. Add any required MIBS to nmis_oids.nmis

   a. if you wanted to use names instead of OID's add entries like this.

```
"tcp" "1.3.6.1.2.1.6"
"tcpActiveOpens" "1.3.6.1.2.1.6.5"
"tcpAttemptFails" "1.3.6.1.2.1.6.7"
"tcpEstabResets" "1.3.6.1.2.1.6.8"
"tcpInErrs" "1.3.6.1.2.1.6.14"
"tcpInSegs" "1.3.6.1.2.1.6.10"
"tcpMaxConn" "1.3.6.1.2.1.6.4"
"tcpOutRsts" "1.3.6.1.2.1.6.15"
"tcpOutSegs" "1.3.6.1.2.1.6.11"
"tcpPassiveOpens" "1.3.6.1.2.1.6.6"
"tcpRetransSegs" "1.3.6.1.2.1.6.12"
"tcpRtoAlgorithm" "1.3.6.1.2.1.6.1"
"tcpRtoMax" "1.3.6.1.2.1.6.3"
"tcpRtoMin" "1.3.6.1.2.1.6.2"
"tcpStatCurConns" "1.3.6.1.4.1.1872.2.1.8.11.1"
"tcpStatHalfOpens" "1.3.6.1.4.1.1872.2.1.8.11.2"
"tcpStats" "1.3.6.1.4.1.1872.2.1.8.11"
```

Modelling tcp 3

3. Add an entry to: Common-heading.nmis

```
'tcp-conn' => 'TCP Connections',
'tcp-segs' => 'TCP Segments'
```

   a. these are the headings you will see when displaying the graph in various screens in NMIS.

   b. If not defined you will see a message like this: “heading not defined in Model”

Modelling tcp 4

4. Add an entry to: Common-database.nmis

```
'tcp' => '/health/$nodeType/$node-tcp.rrd',
```

   a. **tcp** must match the model section.

   b. use variables for various options

Modelling tcp 5

5. Create any required graphs to view data: Graph-tcp-conn.nmis and Graph-tcp-segs.nmis

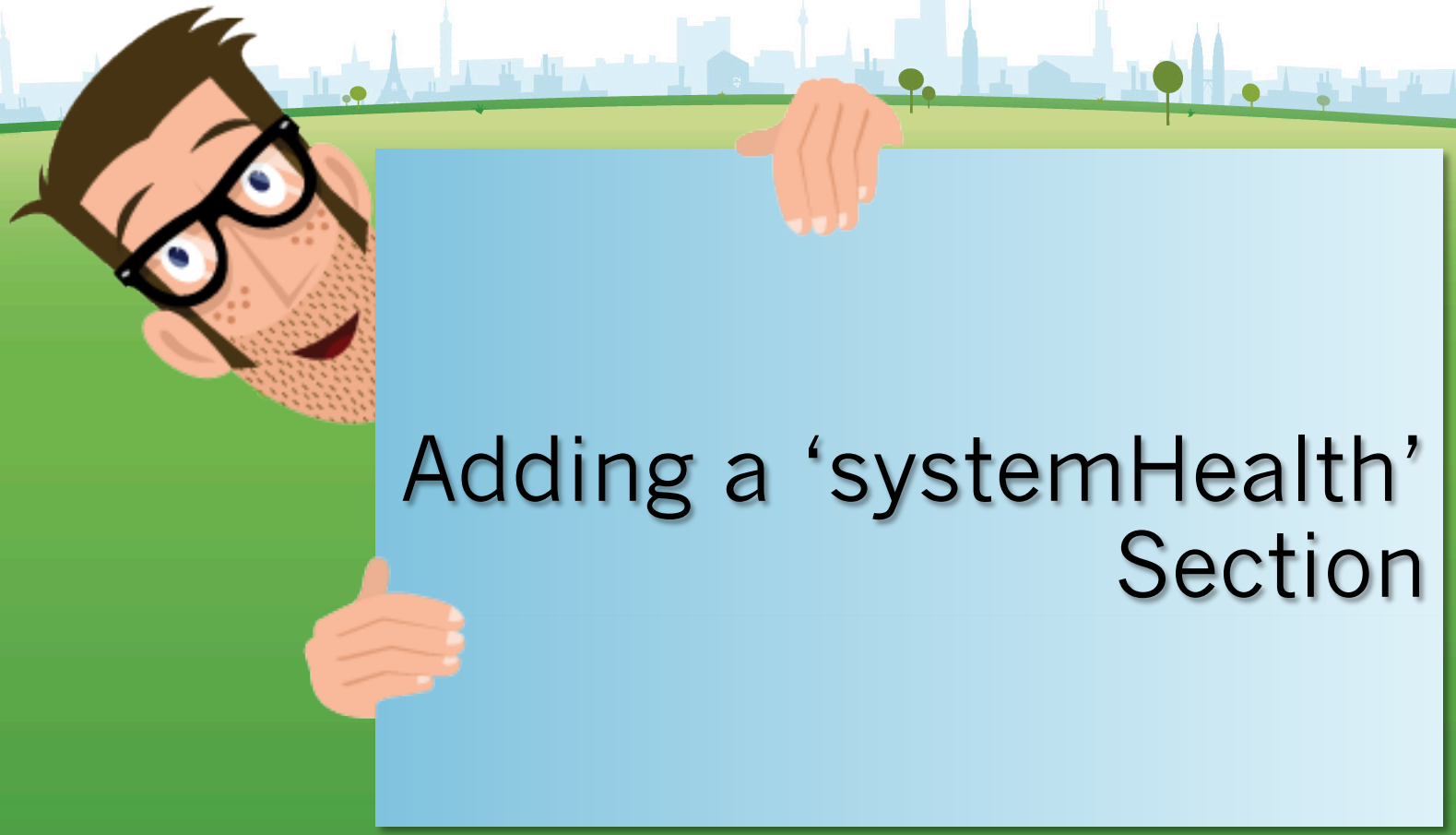
   a. Define the RRD "DEF" based on what you stored.

   b. Defined the LINE or AREA to graph

   c. Use some GPRINTS for text output.

   d. RRD is a language!

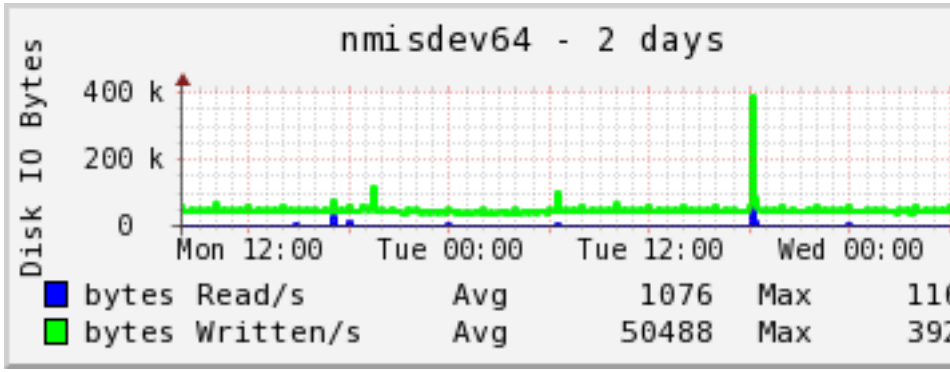
```
'title' => {
  'standard' => '$node - $length from
$datestamp_start to $datestamp_end',
  'short' => '$node - $length'
},
'vlabel' => {
  'standard' => 'TCP Segment Statistics',
  'short' => 'TCP Segment Stats'
},
'option' => {
  --snip--
  'small' => [
    'DEF:tcpInSegs=$database:tcpInSegs:AVERAGE',
    'DEF:tcpInErrs=$database:tcpInErrs:AVERAGE',
    'DEF:tcpOutSegs=$database:tcpOutSegs:AVERAGE',
    'DEF:tcpOutRsts=$database:tcpOutRsts:AVERAGE',
    'DEF:tcpRetransSegs=
$database:tcpRetransSegs:AVERAGE',
    'CDEF:tcpInSegsSplit=tcpInSegs,-1,*',
    'CDEF:tcpInErrsSplit=tcpInErrs,-1,*',
    'AREA:tcpInSegsSplit#0000ff:Input Segments',
    'STACK:tcpInErrsSplit#ffff00:Input Errors',
    'AREA:tcpOutSegs#00ff00:Output Segments',
    'STACK:tcpOutRsts#000000:Output Resets',
    'STACK:tcpRetransSegs#ff0000:Retransmitted',
  ]
}
```


Adding a ‘systemHealth’
Section

The systemHealth Section

Added in NMIS 8.3.18G, supports new collections without coding

- diskIOTable is included in the Model-net-snmp.nmis file.
- Implementation Decisions:
  - What table of data to collect?
  - The MIB name is **diskIOTable**
  - The collection will be called **diskIOTable**



- We will need to make the following changes:
  1. Add the systemHealth section to the model file
  2. Add any required MIBS to nmis_oids.nmis (optional)
  3. Add an entry to: Common-heading.nmis
  4. Add an entry to: Common-database.nmis
  5. Create any required graphs to view data: Graph-diskio-rw.nmis and Graph-diskio-rwbytes.nmis
  6. Add new section diskIOTable to sections in the header

What does the diskIOTable look like?

It is a standard SNMP Table construct.

UCD-DISKIO-MIB::diskIOIndex.**26** = INTEGER: **26**

UCD-DISKIO-MIB::diskIODevice.**26** = STRING: **sda**

UCD-DISKIO-MIB::diskIONRead.**26** = Counter32: 3524873216

UCD-DISKIO-MIB::diskIONWritten.**26** = Counter32: 3281483776

UCD-DISKIO-MIB::diskIOReads.**26** = Counter32: 1574933

UCD-DISKIO-MIB::diskIOWrites.**26** = Counter32: 182695521

UCD-DISKIO-MIB::diskIONReadX.**26** = Counter64: 7819840512

UCD-DISKIO-MIB::diskIONWrittenX.**26** = Counter64: 1150037751808

Modelling diskIOTable 1a

1. Add the systemHealth section if not already there.
   a. Two parts, sys and rrd
   b. the sys section holds non-rrd info, “the discovery”
   c. the rrd section values to store in the rrd
   d. Inside each section the MIBS required for discovery, display and performance collection and storage.

```
 'systemHealth' => {
    'sections' => 'diskIOTable',
    'sys' => {
      'diskIOTable' => {
        'indexed' => 'diskIOIndex',
        'headers' => 'diskIODevice',
        'snmp' => {
          --snip--
        }
      }
    },
    'rrd' => {
      'diskIOTable' => {
        'control' => 'CVAR=diskIODevice;$CVAR =~ /
sda|sr|disk/',
        'indexed' => 'true',
        'graphtype' => 'diskio-rw,diskio-rwbytes',
        'snmp' => {
          --snip–
        }
      }
    }
  },
```

Modelling diskIOTable 1b - sys

1. The sys section defines how to treat the data.

   a. `indexed` is the SNMP variable used to index this data, it is the “primary key”

   b. `headers` what will be used as the headers when displayingthe rrd section values to store in the rrd

   c. `snmp` what to collect,

      ① `oid` can be something nmis_oids.nmis or an OID

      ② `title` what to call it when it displayed.

```
'sys' => {
   'diskIOTable' => {
     'indexed' => 'diskIOIndex',
     'headers' => 'diskIODevice',
     'snmp' => {
       'diskIOIndex' => {
         'oid' => 'diskIOIndex',
         'title' => 'IO Device Index'
       },
       'diskIODevice' => {
         'snmp' => 'diskIODevice',
         'title' => 'IO Device Name'
       },
       --snip--
     },
   },
 },
```

Modelling diskIOTable 1c - rrd

2. The rrd says what to collect and store in the RRD file.

   a. `control` is to limit the saving of data for anything not matched by the control.

   b. `indexed` is this an indexed section or not

   c. `graphtype` what graphs will be created from this information.

   d. `snmp` what to collect,

      ① `oid` can be something nmis_oids.nmis or an OID

      ② `option` is the data a counter or gauge, and what are the lower and upper limits

      ③ `title` what to call it when it displayed.

```
  'rrd' => {
      'diskIOTable' => {
        'control' => 'CVAR=diskIODevice;$CVAR
=~ /sda|sr|disk/',
        'indexed' => 'true',
        'graphtype' => 'diskio-rw,diskio-
rwbytes',
        'snmp' => {
          'diskIONReadX' => {
            'oid' => 'diskIONReadX',
            'option' => 'counter,0:U',
            'title' => 'The number of bytes
read from this device since boot'
          },
          'diskIONWrittenX' => {
            'oid' => 'diskIONWrittenX',
            'option' => 'counter,0:U',
            'title' => 'The number of bytes
written from this device since boot'
          },
          --snip--
        }
      }
    }
```

Modelling diskIOTable 2

2. Add any required MIBS to nmis_oids.nmis

   a. if you wanted to use names instead of OID's add entries like this.

```
"diskIOIndex" "1.3.6.1.4.1.2021.13.15.1.1.1"
"diskIODevice" "1.3.6.1.4.1.2021.13.15.1.1.2"
"diskIONRead" "1.3.6.1.4.1.2021.13.15.1.1.3"
"diskIONWritten" "1.3.6.1.4.1.2021.13.15.1.1.4"
"diskIOReads" "1.3.6.1.4.1.2021.13.15.1.1.5"
"diskIOWrites" "1.3.6.1.4.1.2021.13.15.1.1.6"
"diskIOLA1" "1.3.6.1.4.1.2021.13.15.1.1.9"
"diskIOLA5" "1.3.6.1.4.1.2021.13.15.1.1.10"
"diskIOLA15" "1.3.6.1.4.1.2021.13.15.1.1.11"
"diskIONReadX" "1.3.6.1.4.1.2021.13.15.1.1.12"
"diskIONWrittenX""1.3.6.1.4.1.2021.13.15.1.1.13
"
```

Modelling diskIOTable 3

3. Add an entry to: Common-heading.nmis

```
'diskio-rw' => 'Disk IO Blocks',
'diskio-rwbytes' => 'Disk Read Write Bytes'
```

   a. these are the headings you will see when displaying the graph in various screens in NMIS.

   b. If not defined you will see a message like this: “heading not defined in Model”

Modelling diskIOTable 4

4. Add an entry to: Common-database.nmis

```
'diskIOTable' => '/health/$nodeType/$node-diskiotable-$index.rrd',
```

   a. **diskIOTable** must match the model section.

   b. use variables for various options

   c. $index used for indexed objects like the diskIOTable

Modelling diskIOTable 5

5. Create any required graphs to view data: Graph-diskio-rw.nmis and Graph-diskio-rwbytes.nmis

   a. Define the RRD “DEF” based on what you stored.

   b. Defined the LINE or AREA to graph

   c. Use some GPRINTS for text output.

   d. RRD is a language!

```
'title' => {
  'standard' => '$node - $length from
$datestamp_start to $datestamp_end',
  'short' => '$node - $length'
},
'vlabel' => {
  'standard' => 'Disk IO Activity'
},
'option' => {
  'standard' => [
    'DEF:diskIOReads=
$database:diskIOReads:AVERAGE',
    'DEF:diskIOWrites=
$database:diskIOWrites:AVERAGE',
    'LINE2:diskIOReads#0000ff:Blocks Read/s\t',
    'GPRINT:diskIOReads:AVERAGE:Avg %8.2lf',
    'GPRINT:diskIOReads:MAX:Max %8.2lf\\n',
    'LINE2:diskIOWrites#00ff00:Blocks Written/s
\t',
    'GPRINT:diskIOWrites:AVERAGE:Avg %8.2lf',
    'GPRINT:diskIOWrites:MAX:Max %8.2lf\\n',
  ]
}
```

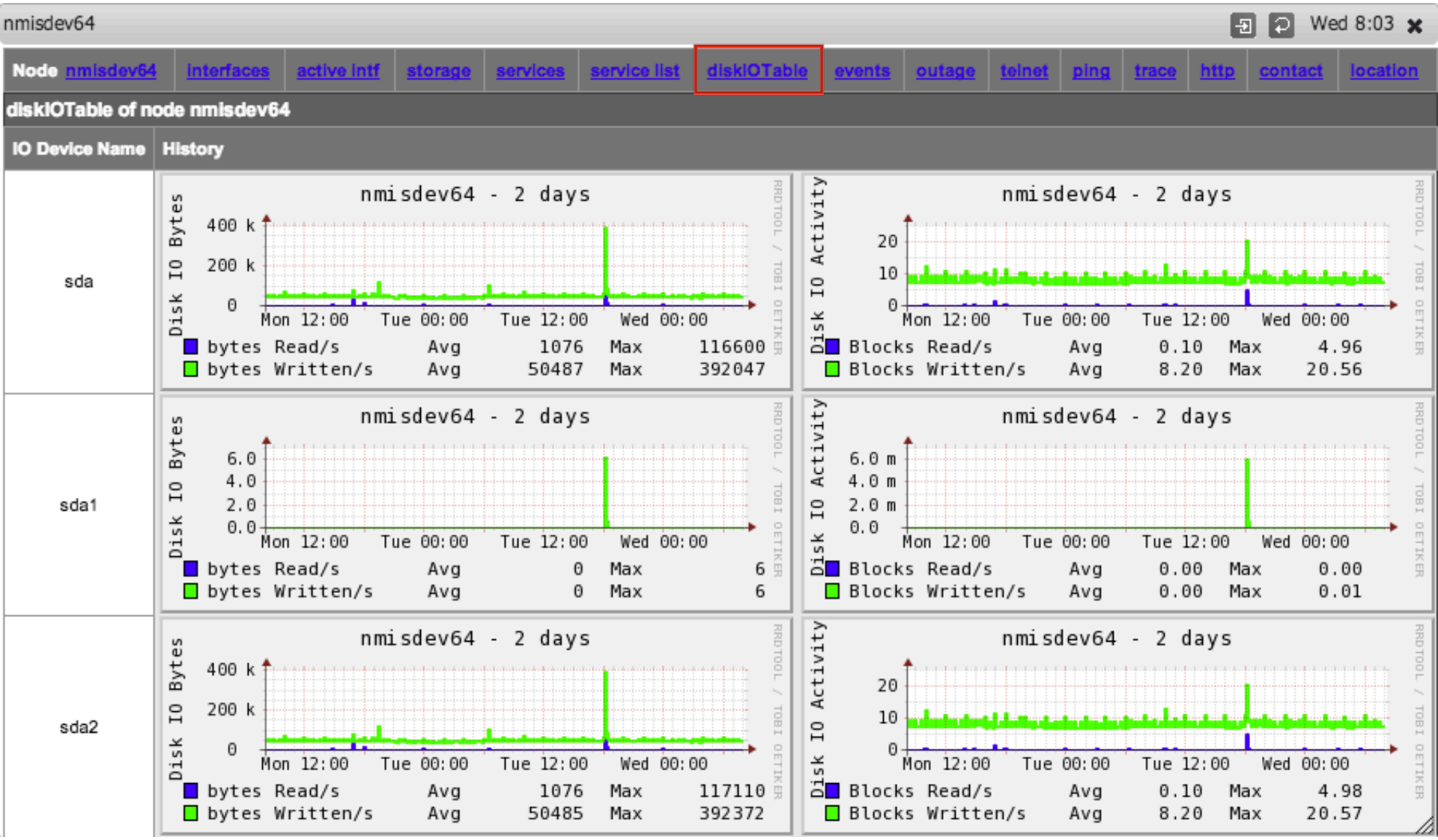
Modelling diskIOTable 6

6. Add new section diskIOTable to sections

```
'sections' => 'diskIOTable',
```

   a. This tells NMIS to start looking for these sections in the systemHealth model, it is like a bootstrap.

Automatically added to the Node View




Thresholding

Steps to add a new thresholding

1. What is being collected which can be thresholded?
2. Add a threshold property to the model section
3. Add threshold values to Common-threshold.nmis
4. Add statistics extraction to Common-stats.nmis
5. Test the thresholding
6. Advanced thresholds with controls

Thresholding Steps 1

- What do you want to threshold?
- Does it make sense?
- Can you boil down the metrics to a meaningful threshold?
- What event name to give to the thresholding event?
  - The event name MUST include “Proactive” at the beginning so that NMIS knows to handle it correctly. e.g. “Proactive Temp” or “Proactive CPU Load”

Steps and Examples

2. Add threshold property to the model section.

3. Add threshold values to Common-threshold.nmis

4. Add statistics extraction to Common-stats.nmis

2.

```
'systemHealth' => {
  --snip--
  'rrd' => {
    'env_temp' => {
      'indexed' => 'true',
      'threshold' => 'env_temp',
      --snip--
    }
  }
}
```

3.

```
'env_temp' => {
  'item' => 'currentTemp',
  'event' => 'Proactive Temp',
  'select' => {
    'default' => {
      'value' => {
        'fatal' => '90',
        'critical' => '80',
        'major' => '70',
        'minor' => '60',
        'warning' => '50'
      }
    }
  }
},
```

4.

```
'env_temp' => [
  'DEF:currentTemp=$database:currentTemp:AVERAGE',
  'PRINT:currentTemp:AVERAGE:currentTemp=%1.2lf',
],
```

Testing Thresholds

- Run thresholds manually

```
nmis.pl type=thresholds debug=true
```

- Look for the value being returned, and change the value to be below that
- Run the thresholds manually, check the event, check the results.
- Change the value back and run the thresholds again, the threshold should close as it goes under the value.

```
'env_temp' => {
  'item' => 'currentTemp',
  'event' => 'Proactive Temp',
  'select' => {
    'default' => {
      'value' => {
        'fatal' => '90',
        'critical' => '80',
        'major' => '70',
        'minor' => '60',
        'warning' => '5'
      }
    }
  }
},
```

Advanced Thresholds with Controls

- For example, different thresholds for core devices.
- Looking in Common-thresholds will give you some ideas, but you can add many “selects” and have properties like:
  - $name
  - $node
  - $host
  - $group
  - $roleType
  - $nodeModel
  - $nodeType
  - $nodeVendor
  - $sysDescr
  - $sysObjectName
  - others for interface
- Almost unlimited possibilities.

```
'cpu' => {
  'item' => 'avgBusy5min',
  'event' => 'Proactive CPU',
  'select' => {
    '10' => {
      'value' => {
        'critical' => '60',
        'fatal' => '70',
        'minor' => '40',
        'warning' => '30',
        'major' => '50'
      },
      'control' => '$roleType =~ /
core/'
    },
    --snip--
    'default' => {
      'value' => {
        'critical' => '70',
        'fatal' => '80',
        'minor' => '50',
        'warning' => '40',
        'major' => '60'
      }
    }
  }
},
```


Custom Alerting

Custom Alerting

- Custom alerting was added to NMIS to support very flexible modelling for fault related polling.
- This has been documented on the Opmantek Community WIKI @

https://community.opmantek.com/display/NMIS/Alerts+-+Using+models+to+generate+custom+events


Running and
Troubleshooting

Running your “models”

- After changing the model, run an update and a collect for the device you are working on.

```
/usr/local/bin/nmis8/nmis.pl type=update model=true node=nodename

/usr/local/bin/nmis8/nmis.pl type=collect model=true node=nodename
```

- The model=true will display some very handy debug output to assist in seeing what is being collected and if there are any errors.

```
MODEL getData nmisdev64 class=systemHealth:
  section=diskIOTable index=26 port=
    oid=diskIOReads name=diskIOReads index=26 value=594
    oid=diskIOWrites name=diskIOWrites index=26 value=24
    oid=diskIONReadX name=diskIONReadX index=26 value=2397184
    oid=diskIONWrittenX name=diskIONWrittenX index=26
value=30720
```

Debugging Device Modeling

- Running an NMIS debug for update and collect is a good idea, especially the collect:

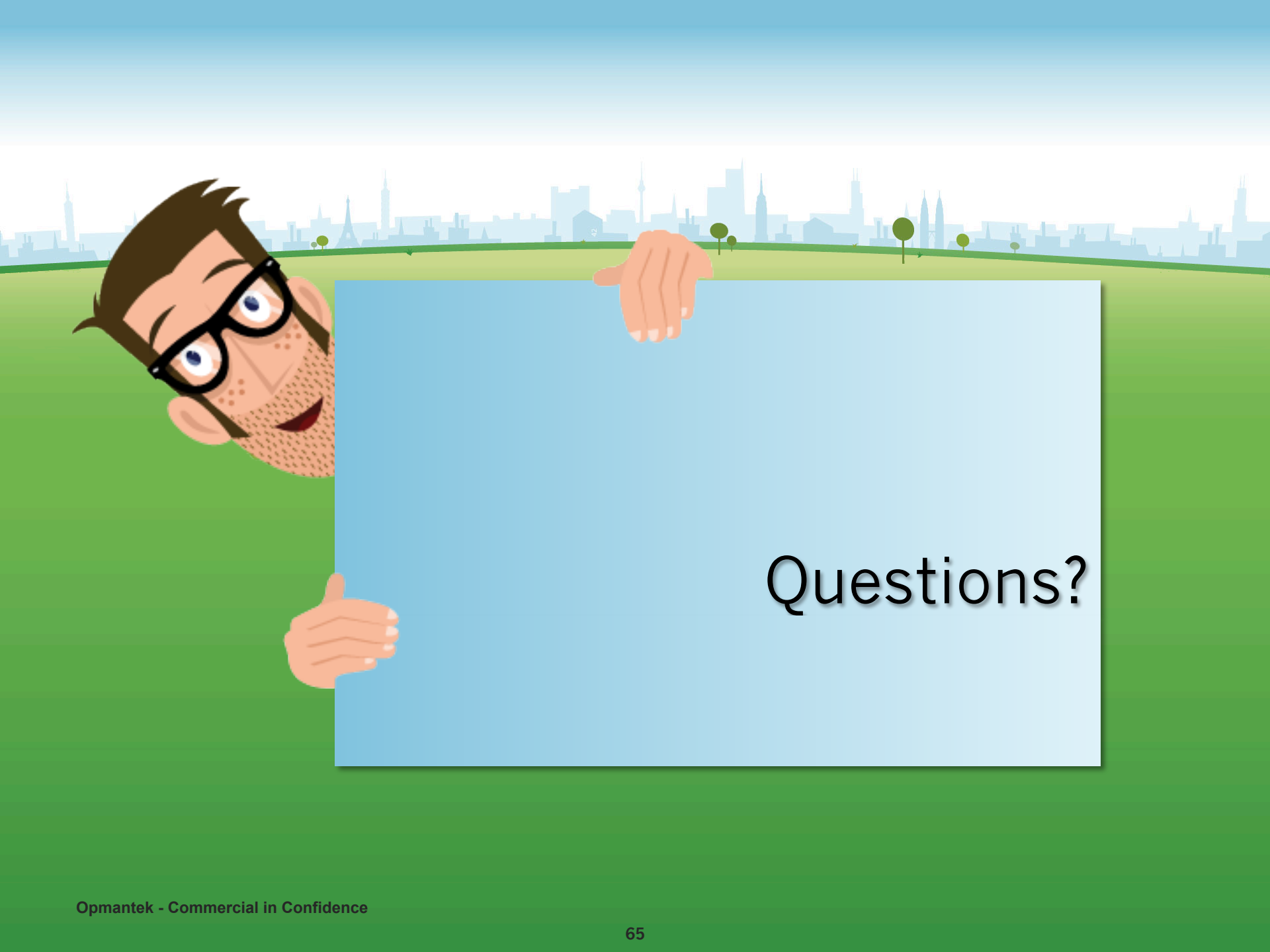
```
/usr/local/bin/nmis8/nmis.pl type=collect debug=true node=nodename
```

- In the debug output look for the “updateRRD” lines:

```
15:53:44 updateRRD, DS diskIOReads:diskIOWrites:diskIONReadX:diskIONWrittenX, 4
15:53:44 updateRRD, value N:1132592:19311700:14779841536:113704411136, 32 bytes
```

- To see what the modeling is resulting in, check the node file in the var folder, this file contains the raw model data and is similar to JSON.

```
/usr/local/bin/nmis8/var/nodename-node.nmis
```


Questions?