Managing Nodes with High Interface Counts

NMIS has been optimised to provide the critical information people need about the network. This means that it wants to keep all the information about your network up to date. As networks have gotten larger and larger in the last few years and more and more network processing is done in software, we have noticed a trend to devices with VERY high interface counts. Lately Opmantek has been seeing interface counts on single devices of over 100000 (one hundred thousand).

A key principle of NMIS has always been to filter out the noise and only provide information which is going to be important, e.g. primary links of networks, not every link to every user, and our customers agree that this is the primary focus of what they want to do. From these devices they want to manage with over 100K interfaces they are often only interested in performance data from 1000 or less interfaces. This article is going to cover some of the considerations and features in NMIS which will help you handle nodes with high interface counts.

NMIS Principles for Interface Collection

NMIS is working to provide a balance of enough information to manage your network without requiring too much information that you need 1 terabyte per day to store the data or 32 cores and 32 GB to poll all the interfaces. NMIS keeps 7 days of granular data and this can be configured to be as much as you like, discussed in Amount of Performance Data Storage NMIS8 Stores, this has been found to be a good balance of important operational information and disk usage. To do this the NMIS models have various settings and there are global configuration options which override the per model settings.

For example, by default, on most devices, NMIS will not collect data from an interface unless it has a description, e.g. if a Cisco Switch has 500 interfaces, NMIS would only collect the interfaces which have descriptions. By default, NMIS will not collect certain interface types, this is because we have found that many interface types do not contain all the data in the MIB, so why collect them.

Problems with SNMP Agents

NMIS talks to 1000's of different vendor products, and some of those devices have less than well performing SNMP agents. When you have a slow SNMP agent, along with a high interface count, the result can be very slow polling. For example we have found some SNMP agents which have 1000's of interfaces and an NMIS update takes less than 30 seconds, while other devices have taken over 60 minutes to run an update.

Which Nodes are Taking a Long Time to Poll

In NMIS 8.5.10 a feature was added for granular polling timers, this can be enabled with the NMIS configuration setting log_polling_time, set this value to true to see output like the sample below, the "doCollect" indicates it is from a collect (poll) cycle, the codename and then the model being used. This should assist to find which nodes are slowing down your poll cycle.

```
24-Aug-2015 17:55:07,nmis.pl::doCollect#976<br>Poll Time: cloud, PingOnly, 0.01
24-Aug-2015 17:55:07,nmis.pl::doCollect#976<br>Poll Time: gate, PingOnly, 0.03
24-Aug-2015 17:55:07,nmis.pl::doCollect#976<br>Poll Time: golden, PingOnly, 0.02
24-Aug-2015 17:55:07,nmis.pl::doCollect#976<br>Poll Time: bones, net-snmp, 0.31
24-Aug-2015 17:55:07,nmis.pl::doCollect#976<br>Poll Time: excalibur, ESXi, 0.26
24-Aug-2015 17:55:07,nmis.pl::doCollect#976<br>Poll Time: elli, Windows2008, 0.32
24-Aug-2015 17:55:08,nmis.pl::doCollect#976<br>Poll Time: meatball, CiscoRouter, 0.63
24-Aug-2015 17:55:08,nmis.pl::doCollect#976<br>Poll Time: gdc-sw1, Netgear-GS724T, 0.86
24-Aug-2015 17:55:08,nmis.pl::doCollect#976<br>Poll Time: omed, net-snmp, 0.27
24-Aug-2015 17:55:08,nmis.pl::doCollect#976<br>Poll Time: ran, net-snmp, 0.69
24-Aug-2015 17:55:09,nmis.pl::doCollect#976<br>Poll Time: NETGEAR-MANUAL, Netgear-Manual, 2.18
24-Aug-2015 17:55:09,nmis.pl::doCollect#976<br>Poll Time: wanedge1, CiscoRouter, 0.42
24-Aug-2015 17:55:10,nmis.pl::doCollect#976<br>Poll Time: yang, QNAP, 0.87
24-Aug-2015 17:55:10,nmis.pl::doCollect#976<br>Poll Time: kaos, Windows2012, 3.24
```

NMIS Configuration Options and Features

interface_max_number

NMIS can handle devices with very high device counts, but we have noticed that people don't really know that they have these devices or how many interfaces they have. Because managing 1000's of interfaces could fill the disk or prevent effective polling, the interface_max_number configuration option was added in NMIS 8.5.8G, the default setting is 5000.

This means that if NMIS finds a device with an interface count over the configured setting, it will NOT collect interface data from that node at all. This means that the NMIS administrator has to consciously enable handling of devices which high interface counts and they can monitor how well the device and NMIS are handling collecting the data.

Polling Locks

This feature was added in NMIS 8.5.10. Polling locks work that when an update or collect poll runs on a node, a lock is created, preventing another NMIS process from starting an update or collect. This is done because an update on a node with high interface counts can go for quite a while, we don't want another update running on the node at the same time. We also don't want NMIS running a collect on a node which will start an update if an update has never been run, and the server might get caught with too many blocked processes.

NMIS Modelling Options

The following modelling options were introduced in NMIS 8.5.10 to assist with managing devices with very high interface counts. We have found that the majority of devices do not require these but for some vendors using SNMPv1 or with very high interface counts, these features can improve polling performance dramatically.

A key concept with these features is to avoid running an update cycle for interfaces when changes are detected and to minimise the number of bulk walks NMIS does, most devices handle this no problem, but when there are slow SNMP engines or VERY high interface counts, optimisation is needed.

To configure these options, a '`custom`' model section can be added to a model. The '`custom`' model section needs to be a top-level section, ie. on the same nesting level as '`-common-`' for example:

```
%hash = (
  '-common-' => {
    # lots of lines skipped
  },
  'custom' => {
    # the custom model options go here
  },
  # and the rest of the model follows here
  'interface' => {
...
```

An example `custom` section is shown below, which contains default settings, ie. as if the custom section was not present at all. You'll find a description of each of the features in more detail in the next sections.

Default Interface Handling Options

```
  'custom' => {
    'interface' => {
      'ifAdminStatus' => 'true',
      'ifNumber' => 'true',
      'ifLastChange' => 'false',
      'ifTableLastChange' => 'false',
      'skipIpAddressTableOnSingle' => 'false',
    }
  },
```

Suggested Interface Handling for Devices with Interface Polling Issues

```
'custom' => {
  'interface' => {
    'ifAdminStatus' => 'false',
    'ifNumber' => 'false',
    'ifLastChange' => 'true',
    'ifTableLastChange' => 'true',
    'skipIpAddressTableOnSingle' => 'true',
  }
},
```

Model custom - interface - ifAdminStatus

Default: true

Every poll cycle NMIS is checking the ifAdminStatus and ifOperStatus to see if interfaces have changed state or not. This is not a problem with regular nodes, but when slow SNMP agents combine with high interface counts, this can take extra time. In the model for the node you can disable this processing by adding a top level modelling section as below. NMIS will still check the interface status of interfaces being collected but will not look for interfaces which have changed state.

Model custom - interface - ifNumber

Default: true

Every poll cycle the number of interfaces is verified using the SNMP MIB ifNumber, if this value has changed from the last poll, an interface has been added or deleted and by default NMIS will run an interface update to recalculate what needs to be polled.

Model custom - interface - ifLastChange

Default: false

If disabling ifAdminStatus, you really need a way to check that interfaces have changed or not, which is done with this option set to true. This will use the mib value ifLastChange to determine if an interface has changed state and then run an update on that interface alone.

The description of this MIB variable is: "The value of sysUpTime at the time the interface entered its current operational state. If the current state was enter ed prior to the last re-initialization of the local network management subsystem, then this object contains a zero value."

Model custom - interface - ifTableLastChange

Default: false

When enabled, the interface update process for all interfaces will be skipped if this value has not changed. NMIS will collect this value from the node and compare it to the last time it was collected, if there time is newer than it will run a full interface update.

The SNMP variable ifTableLastChange holds the value of sysUpTime when the interface table last changed, that is when an interface was added or deleted. The default value for this is false. When set to true the NMIS update process will check this value for each node and only run a complete update of the interface table when this value has changed. For a node with high interface counts, this can result in not needing to run an update very often at all. Which tends to work well as these devices do not often have new interface being added.

The description of this MIB variable is: "The value of sysUpTime at the time of the last creation or deletion of an entry in the ifTable. If the number of entries has been unchanged since the last re-initialization of the local network management subsystem, then this object contains a zero value."

Model custom - interface - skipIpAddressTableOnSingle

Default: false

When enabled, the IP address table will not be updated when a single interface is being updated.

Model custom - interface - skipIfType skipIfDescr

Skip Interfaces is used for mega devices to get rid of unwanted interfaces. Big SP switches for example. If skipIfType or skipIfDescr regex is matched, then just skip this interface all together. Must be used with ifNumber = false, ifAdminStatus = false, ifLastChange = false, ifTableLastChange = false

For example, skip all the PPP interfaces on a DSLAM so you can monitor the Ethernet uplinks.

```
    'skipIfType' => 'ppp',
    'skipIfDescr' => 'Virtual-Access|noSuchInstance'
```