Modelling MIBS that use Indexes using the systemHealth section

Introduction

Often a section of data that is useful to have displayed in NMIS is presented in SNMP as a table. In order to model this NMIS modelling supports a "systemHealth" section that allows indexing to be used.

The net-snmp model has a section called "diskIOTable" which is an example of how to use the indexed modelling features of NMIS.

Model.nmis file

The systemHealth section is a "top-level" section which means it does not sit inside another section.

```
%hash = (
-- SNIP --
  'systemHealth' => {
    'sections' => 'diskIOTable',
    'sys' => {
      'diskIOTable' => {
      'indexed' => 'diskIOIndex',
      'headers' => 'diskIODevice',
      'snmp' => {
        'diskIOIndex' => {
          'oid' => 'diskIOIndex',
          'title' => 'IO Device Index'
        },
        'diskIODevice' => {
          'oid' => 'diskIODevice',
          'title' => 'IO Device Name'
        },
        -- SNIP --
      }
    },
    'rrd' => {
      'diskIOTable' => {
        'control' => 'CVAR=diskIODevice;$CVAR =~ /sda|sr|disk|xvda|dm\-/',
        'indexed' => 'true',
        'graphtype' => 'diskio-rw,diskio-rwbytes',
        'snmp' => {
          -- SNIP --
        }
      }
    }
  },
-- SNIP --
);
```

At the top of the systemHealth section a "sections" key tells NMIS which sections to display in the GUI (along the top column of links while viewing the node). This name must match the name of the section below.

In this example "diskIOTable" is used,

sys section

In the above code snippet there is a 'sys' section, this is where data that will be stored in the *Nodename*-node.nmis file is defined. This is also where data that is needed for gathering the RRD section is defined. If you want to see the latest value gathered by NMIS for these MIBS check the *Nodename*-node.nmis file for your node. The values defined inside the snmp section are like any other part of the model.

The sys->diskIOTable section specifies 2 keys:

1. indexed => 'diskIOIndex'
   This tells NMIS that the OID diskIOIndex will hold the index value to loop on.
2. headers => 'diskIODevice'
   This tells NMIS that when displaying the indexes, a column for diskIODevice should be shown. If the headers directive lists more than one key, then all the corresponding columns will be present; in the example above we could have included the diskIOIndex column, too (but the device name is more useful in this case).

NMIS versions before 8.4.8 require that you list the OID name-to-raw-oid association in `mibs/nmis_mibs.oid`; for the example above entries for diskIODevice and diskIOIndex would be required. To simplify the modelling process, 8.4.8 and later also support the key `index_oid` which lets you associate a raw oid with a name in one model file. Using that the example would look like this:

```
'sys' => {
  'diskIOTable' => {
   'indexed' => 'diskIOIndex',
   'index_oid' => '1.3.6.1.4.1.2021.13.15.1.1.1',
...
```

Another new feature in 8.4.8 is `index_regex`, which allows multi-element indexing: normally SNMP tables are indexed by the last, single numeric OID component. When NMIS does an update on a indexed entity, it iterates through all the known values for this index component and records them. This iteration does not work if the index consists of more than one number, as it does on certain equipment. In such cases you can set `index_regex`to a value that captures the OID components that vary between table elements. For example,

```
'index_regex' => '\.(\d+\.\d+\.\d+)$',
```

ensures that the last three numbers are used for indexing.

How this information looks like?

The OID 1.3.6.1.4.1.2021.13.15.1.1.1 is defined as follows in the mibs:

```
diskIOIndex OBJECT-TYPE
    SYNTAX      Integer32 (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Reference index for each observed device."
    ::= { diskIOEntry 1 }
```

Doing a snmpwalk

snmpwalk -c COMMUNITY -v 2c localhost 1.3.6.1.4.1.2021.13.15.1.1:

```
1.3.6.1.4.1.2021.13.15.1.1.1.1 = INTEGER: 1
1.3.6.1.4.1.2021.13.15.1.1.1.2 = INTEGER: 2
1.3.6.1.4.1.2021.13.15.1.1.1.3 = INTEGER: 3
1.3.6.1.4.1.2021.13.15.1.1.2.1 = STRING: "fd0"
1.3.6.1.4.1.2021.13.15.1.1.2.2 = STRING: "sdb"
1.3.6.1.4.1.2021.13.15.1.1.2.3 = STRING: "sda"
```

So, in this case, 1.3.6.1.4.1.2021.13.15.1.1.1 is the index OID, and by default, it will use the last digit (By default) as the index, 1, 2 & 3 in the example:

1.3.6.1.4.1.2021.13.15.1.1.1.1, 1.3.6.1.4.1.2021.13.15.1.1.1.2

rrd section

The rrd section defines what data will be collected and stored into rrd's. Once again, the values defined inside the snmp section are like any other part of the model.

The rrd->diskIOTable section specifies 3 keys:

1. 'control' => 'CVAR=diskIODevice;$CVAR =~ /sda|sr|disk|xvda|dm\-/',
   This tells NMIS that the OID diskIODevice should be checked and only capture the values into RRD if they match the regular expression given.
   Please see the Advanced Modelling: When a single SNMP variable isn't enough page for further info on custom variables.
2. indexed => 'true',
   Tell NMIS this is an indexed table, it will then go and use the index specified in the sys section above to iterate

3. graphtype => ''
   what graph-types will this rrd section create data for

Common-database.nmis file

The name of the rrd file is specified in this file. You will want a new set of files for your new section, to do that simply add a new line

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

As you can see, the file name has $index in the name so NMIS will create a new file for each index that it is gathering

Conclusion

NMIS should now be displaying a new section in the node view along with any graphs you have added in this section.

Also note, alerts can also be set on your new section, see this documentation page for more info!

As always, remember that you can compile your model file to check for syntax errors as well as run updates and collects with "model=true" (and generally without debug=true) to see what NMIS is gathering.