Modelado (Entrenamiento)

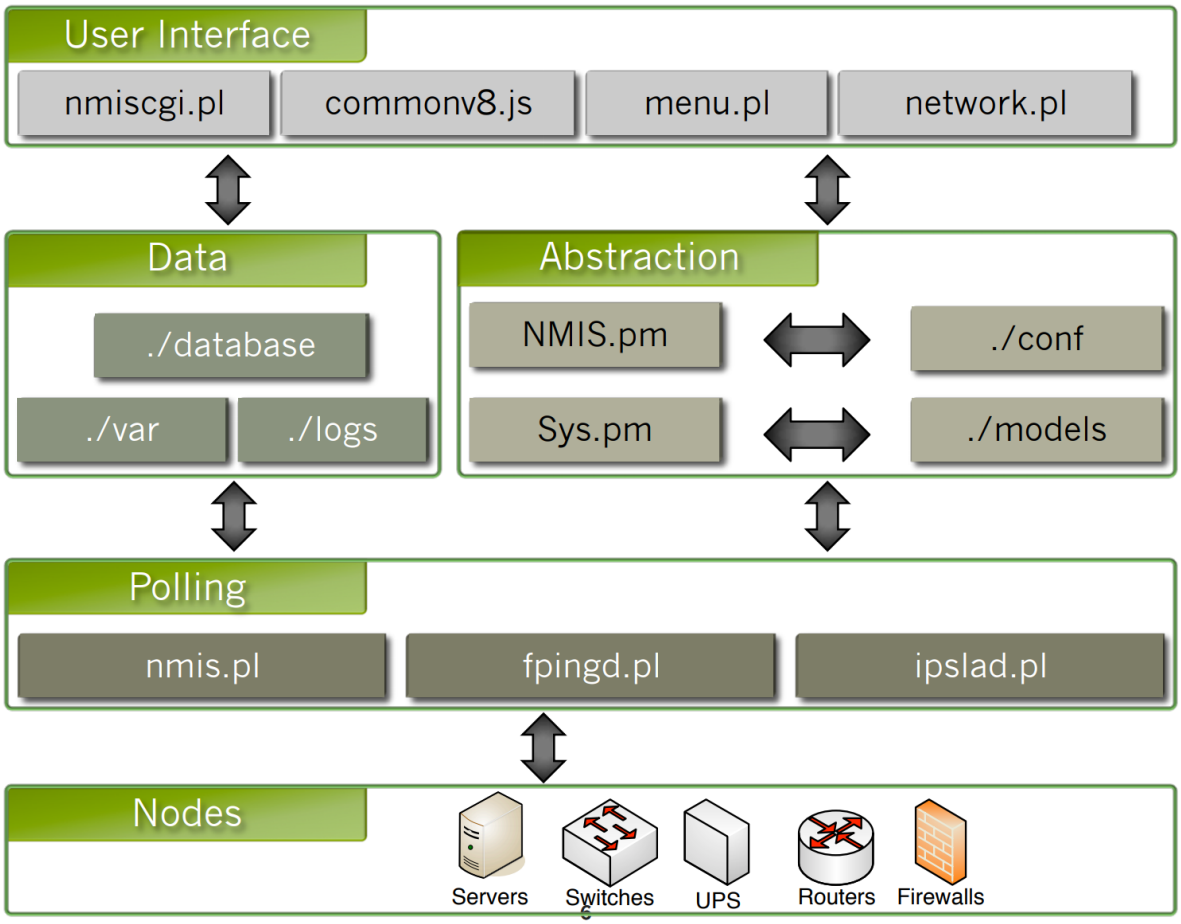
1- Introducción al modelado en NMIS 8

Arquitectura de NMIS 8

Se accede a la información de los nodos utilizando daemons o pollers que forman parte del sistema de poleo. La información obtenida en este nivel podría ser ICMP (Protocolo de mensajes de control de Internet), SNMP (Protocolo simple de administración de redes), etc.

Una vez que se recopila la información del dispositivo, se carga el Sistema de abstracción, este sistema incluye Sys.pm que especifica los métodos para poder comunicarse con los dispositivos, carga la información desde el directorio NMIS "VAR" y desde las bases de datos. El sistema de abstracción también incluye NMIS.pm, que carga la configuración de dispositivos y NMIS y es el responsable de cargar los modelos para los dispositivos, recopilar información de la base del dispositivo en ese modelo, escribir esta información en las bases de datos y en el directorio VAR.

En el nivel de la interfaz de usuario, Network.pl realiza la mayor parte del trabajo pesado, que carga el sistema de abstracción y los datos para producir las vistas, el tablero y todo lo que se muestra al usuario de manera significativa y útil.



Proceso de Modelado

Tipos de archivos de modelo

Estos son los archivos con los que debemos familiarizarnos para poder seguir con éxito el proceso de modelado.

Model.nmis: Este archivo contiene nuestras reglas de auto-descubrimiento del modelo, determinará, en función de ciertas condiciones, qué modelo debe cargarse.

Common - *. nmis: Colección de archivos que contiene secciones comunes, estos se incluyen en muchos modelos que comparten ciertas características.

Graph - *. nmis: Contiene las definiciones de gráficos para cada gráfico que se generará y se carga dinámicamente en tiempo de ejecución.

Model - *. nmis: Modelos de dispositivos que reúnen elementos relacionados obtenidos de los dispositivos, debe incluir toda la estructura necesaria para representar el dispositivo en NMIS.

Enterprise.nmis: Es parte de los archivos de configuración, se encuentra en el directorio "/ conf" y contiene las asignaciones de SNMP OID utilizadas por Model.nmis para que coincidan con los proveedores del dispositivo.

nodename-node.json y nodename-node.json: para cada nodo se genera y mantiene un nodo y un archivo de vista después de un ciclo de sondeo (Actualizar y recopilar). Contiene información de nodo en caché de SNMP MIBS y otros datos derivados, los datos que contiene se conservan temporalmente. El archivo de vista son datos que se mostrarán al presentar la interfaz de usuario.

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.json nodename-view.json	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.

Visión general

1. Recopilación de datos del dispositivo

El primer paso es recopilar datos del dispositivo, sysDescr (Descripción del sistema) y SysObjectId (Identificación del dispositivo por parte del proveedor). NMIS realiza un SNMP GET en la ruta y obtiene el sysDescr y el SysObjectId.

Del sysObjectID, obtenemos la ID y se compara con el archivo Enterprise.nmis para que coincida con el proveedor del dispositivo. En este ejemplo, el OID: 1.3.6.1.4.1.311 coincide con un dispositivo de Microsoft.

Del sysDescr obtenemos información detallada relacionada con el dispositivo, en este caso, la descripción del hardware y el software.

```
'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. Determinar el fabricante

Se debe hacer una comparación entre el sysObjectID obtenido y el listado de proveedor definidas en Enterprise.nmis, el OID 311 devolverá el valor "Enterprise", en este caso particular, es 'Microsoft'.

conf/Enterprise.nmis

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

3. Modelo de autodescubrimiento

Para obtener el modelo a cargar, se debe realizar un proceso de coincidencia en el archivo Model.nmis. El nombre del proveedor obtenido del paso anterior se usa para encontrar la sección que coincide con su valor, una vez que tengamos la sección que se usará, se realizará una expresión regular para que coincida el valor de cada artículo en esa sección con el sysDescr, este proceso debe hacerse en orden ascendente numérico.
En este ejemplo, el nombre del proveedor coincidirá con la sección "Microsoft" en el archivo Model.nmis, luego, buscará una coincidencia en orden, a partir de 10, intentará identificar si el valor 'Windows Versión 5.2' esta incluido en el sysDescr. Seguirá buscando hasta que se encuentre una coincidencia. En este caso, 'Windows Versión 6.1' es una coincidencia, por lo que el modelo a cargar está configurado en: 'Windows2008'.

models/Model.nmis

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

4. Cargar el modelo

El modelo obtenido del archivo Model.nmis sera cargado, para ello se buscara el modelo correspondiente, el nombre de archivo del modelo debe comenzar con "Model-" seguido del nombre del modelo como se especifica en Model.nmis y con la extensión ".nmis". En nuestro ejemplo, el modelo a utilizar es 'Windows2008', por lo que el archivo debe llamarse: Model-Windows2008.nmis

```
$ ls -l /usr/local/nmis8/models/Model-Windows*
-rw-rw---- 1 nmis nmis 10920 Jul 10 14:51 Model-Windows2000.nmis
-rw-rw---- 1 nmis nmis  5761 Jul 10 14:51 Model-Windows2003.nmis
-rw-rw---- 1 nmis nmis  5964 Jul 10 14:51 Model-Windows2008.nmis
```

El modelo cargará todos los elementos comunes enumerados en la sección -common-.

models/Model-Windows2008.nmis

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

Luego, el modelo cargará secciones específicas enumeradas después de la sección -common-.

models/Model-Windows2008.nmis

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

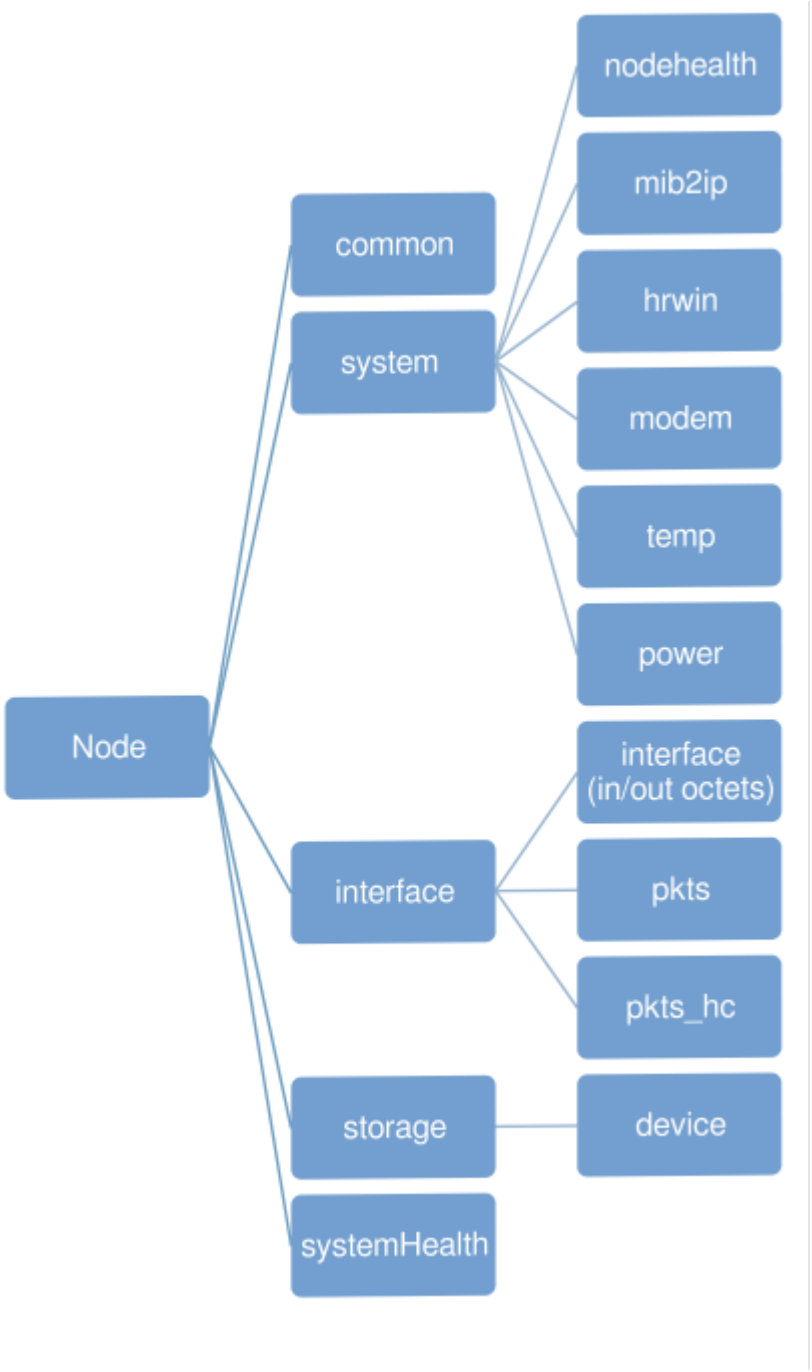
5. Cargando los datos

Uses this model to collect device specific information from the device or to load the cached data from nmis8/var. If there is no file on the var directory for the device, NMIS will start collecting SNMP data from the device.
This process happens every time when dealing with the devices. When using the GUI, the model will be loaded with no SNMP enabled, however, when using NMIS.pl the model will be loaded with SNMP enabled.

Se utiliza este modelo para recopilar información específica del dispositivo desde el dispositivo o para cargar los datos en caché de nmis8/var. Si no hay ningún archivo en el directorio "var" para el dispositivo, NMIS comenzará a recopilar datos SNMP del dispositivo.
Este proceso ocurre cada vez que se trabaja con los dispositivos. Cuando se utiliza GUI, el modelo se cargará sin SNMP habilitado, sin embargo, cuando use NMIS.pl, el modelo se cargará con SNMP habilitado.

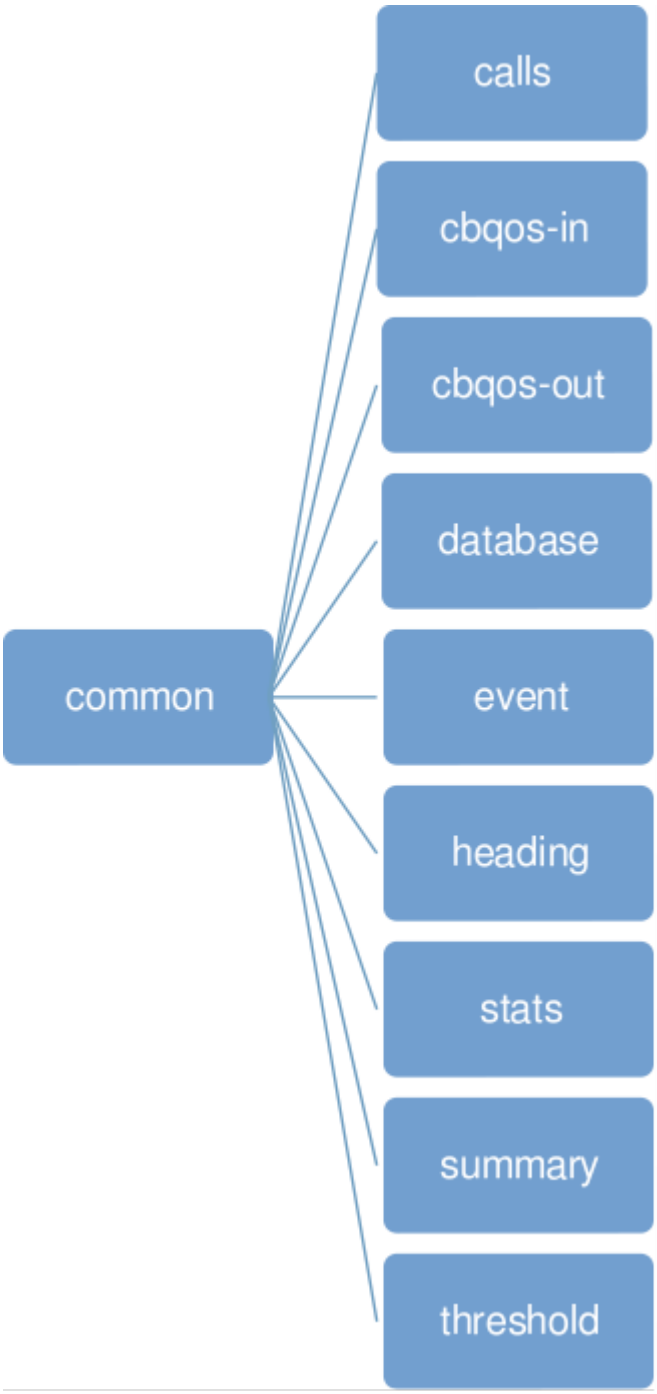
Estructura del modelo

Section	Description
common	Defines what common models to load
system	The device specific concepts, uses flat SNMP structures
interface	Everything related to the interfaces
environment	Currently temperature for indexed MIBS
storage	Disks and memory and virtual memory for servers
systemHealth	Custom modelling for SNMP table structures



Modelos Comunes

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.



Modelado de un dispositivo

Objetivo del modelado

¿Cuál es el objetivo para el modelado?, ¿Solo se requiere soporte de tipo estándar, o la recopilación más avanzada de datos del dispositivo?, ¿Desea recopilar algunos datos de rendimiento sobre cómo funciona un protocolo, o verificar la cantidad de sesiones que ejecuta un firewall?

A veces se puede encontrar el MIB del dispositivo en la documentación o en un documento técnico por parte del proveedor, pero a veces es muy difícil determinar dónde se almacenan los datos necesarios. Si es posible, pregunte a un experto en productos que esté familiarizado con ese producto específico.

Instrumentación de dispositivo

Otras veces, la gente quiere graficar la utilización del CPU y la Memoria, pero no todos los dispositivos admiten la recopilación de esta información, solo puede pedirle a NMIS que recopile algo para lo que el dispositivo tiene la instrumentación, las MIB del dispositivo deben mostrarnos si es posible.

Relevancia de la instrumentación

Para los enrutadores Cisco, es muy útil monitorear la carga del CPU, es una excelente métrica de cómo funciona el dispositivo, sin embargo, en algunos dispositivos Cisco más nuevos, el procesamiento se distribuye y se realiza en hardware, por lo que la carga del CPU sigue siendo útil, pero puede no estar proporcionando la información que necesita.

Verifique la operación de MIB

Ahora ya sabe lo que desea recopilar y monitorear, se debe verificar que la MIB funciona de manera correcta, tal como lo dice la documentación y verifique que funciona de la manera que usted cree que lo hace.

Qué necesitamos

Acceso al dispositivo

Puede modelar un dispositivo sin tener acceso a uno, pero es REALMENTE COMPLICADO, tener acceso de solo lectura SNMP al dispositivo es vital para realizar el modelado con éxito.

SNMP MIBS

Deberá tener todas las MIB estándar (IETF / IEEE) necesarias y las MIB específicas del proveedor para el dispositivo que se va a modelar.

SNMPWALK (SNMP Dump)

Una vez que tenga las MIB, la mejor manera de interpretar las MIB es completar un SNMPWALK del dispositivo, primero verifique que puede usar SNMP para acceder al dispositivo.

Then you need to do a full SNMP WALK, you will need to create a directory to store the MIBs in, for example ~/mibs, then copy the MIBs you obtained for the product and copy them to that folder, you will also need the standard MIBs, which are included in the NMIS distribution in /usr/local/nmis8/mibs/traps, copy these to the same folder, now verify that everything is working, you may get some errors from SNMP WALK about MIB compiling, but you can usually ignore those if you get a good output. For devices with proprietary MIB's or Enterprise MIBS, you should obtain them from the vendor, Google is very helpful and add them to your MIB's in ~ /mibs, before doing the SNMP walk.

Luego, debe hacer un SNMPWALK completo, deberá crear un directorio para almacenar las MIB en, por ejemplo, ~ / mibs, luego copie las MIB que obtuvo para el dispositivo y cópielas en esa carpeta, también necesitará MIB estándar, que se incluyen en la distribución NMIS en /usr/ local/nmis8/mibs /traps, cópielos en la misma carpeta, ahora verifique que todo esté funcionando, puede obtener algunos errores de SNMPWALK sobre la compilación MIB, pero por lo general, pueden ser ignorados si obtiene los datos que se expectan obtener de la salida del comando. Para dispositivos con MIB propietarios o MIBS Empresariales, debe obtenerlos del proveedor, Google es muy útil para encontrar MIBS, y deben agregarlos al directorio ~ / mibs, antes de realizar un SNMPWALK.

Comandos para ejecutar:

```
mkdir ~/mibs
cp <vendor mib file(s)> ~/mibs
cp /usr/local/nmis8/mibs/traps/* ~/mibs
snmpwalk -m ALL -M ~/mibs -v 2c -c GOODCOMMUNITY <HOSTNAME or IP ADDRESS> system
```

Salida esperada:

```
SNMPv2-MIB::sysDescr.0 = STRING: Hardware: Intel64 Family 6 Model 15 Stepping 6 AT/AT COMPATIBLE - Software:
Windows Version 6.1 (Build 7601 Multiprocessor Free)
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::enterprises. 311.1.1.3.1.1
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (40604629) 4 days, 16:47:26.29
SNMPv2-MIB::sysContact.0 = STRING: dc_admin@opmantek.com
SNMPv2-MIB::sysName.0 = STRING: kaos
SNMPv2-MIB::sysLocation.0 = STRING: Head Office
SNMPv2-MIB::sysServices.0 = INTEGER: 79
```

Ejecute un walk completo y redireccione la salida al un archivo para obtener el SNMP Dump.

```
snmpwalk -m ALL -M ~/mibs -v 2c -c GOODCOMMUNITY > snmp_dump.txt
```

Ejemplo de MIB Decoding

Tomemos este ejemplo, aquí snmpwalk no pudo encontrar el archivo MIB adecuado para traducir los datos SNMP devueltos desde el dispositivo. En este ejemplo, estamos interesados en los últimos tres elementos.

```
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
```

Sabemos que “SNMPv2-SMI::enterprises” se traduce siempre en “.1.3.6.1.4.1”, lo cual node deja con estos 3 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

We want to find out what “.1.3.6.1.4.1.6302.2.1.2.2.0” represents, so we google it and no luck, let’s try with a more general OID, so we go up the tree a bit and grab “.1.3.6.1.4.1.6302” to find out to who this OID is assigned to. Google reveals that it belongs to “Emerson Energy Systems”, using our well developed google skills we were able to find the ees-power.mib file.

Queremos saber qué representa ".1.3.6.1.4.1.6302.2.1.2.2.0", así que lo buscamos en Google y no podemos encontrarlo, intentemos con un OID más general, así que subimos un poco al árbol y tomamos ". 1.3.6.1.4.1.6302 ”para averiguar a quién está asignado este OID. Google revela que pertenece a "Emerson Energy Systems", utilizando nuestras habilidades para Googlear, pudimos encontrar el archivo ees-power.mib.

Queremos averiguar qué es cada elemento, por lo que tenemos que rastrear el árbol a través del contenido ASN.1 [1] de la MIB (Management Information Base).

Aquí se muestra parte del contenido de la MIB:

SNMPv2-SMI::enterprises.6302.2.1.

```
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 }
```

Primero podemos notar que las **empresas.6302** son equivalentes a **ees**, por lo que hacemos un reemplazo simple, después de este reemplazo seguimos bajando la MIB. El siguiente elemento es **ees 2**, que se convierte en **global**, continuando con el mismo método, descubrimos que **global 1** es **powerMIB**, **powerMIB 2** es **system** y finalmente **system 2** es **systemVoltage**.

SNMPv2-SMI::enterprises.6302.2.1.2 SNMPv2-SMI::enterprises.ees.global.powerMIB.system

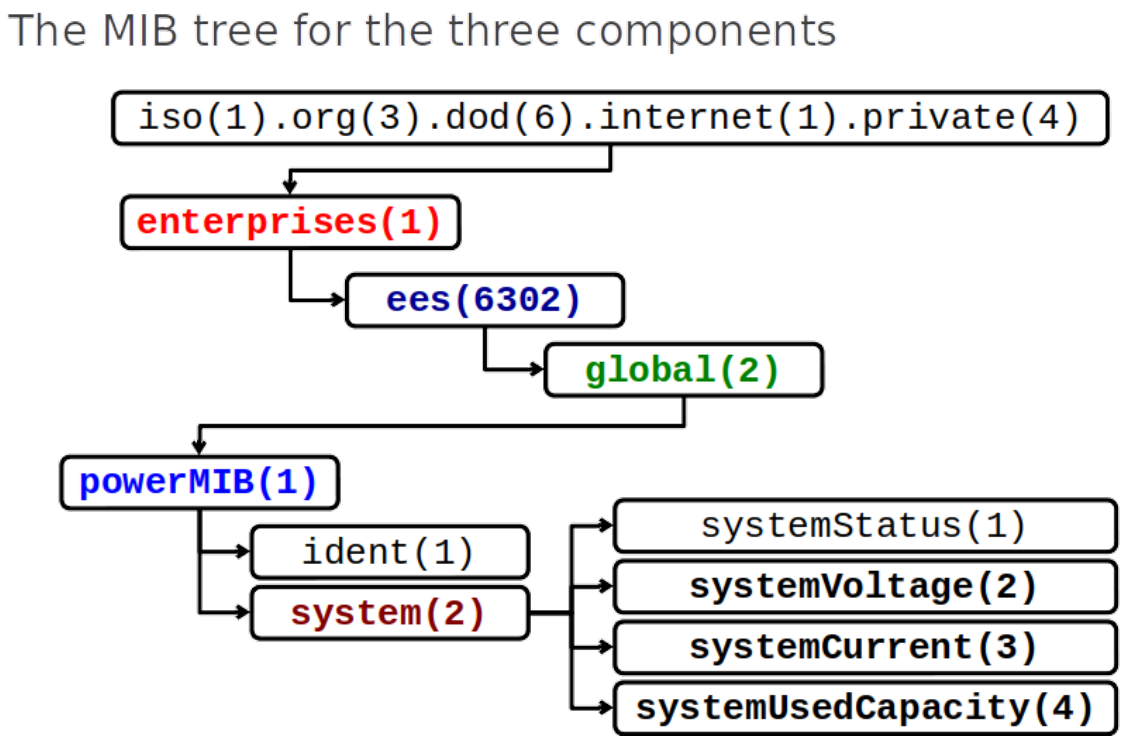
```
system OBJECT IDENTIFIER ::= { powerMIB 2 }
systemVoltage OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION "
                System voltage, stored as mV, including positive or negative
                sign. The integer 2147483647 represents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

Árbol MIB para los tres componentes.

Aquí tenemos una representación gráfica de cómo se descomponen los componentes del MIB.



2- Implementación de un nuevo dispositivo (I)

Agregar Modelos propios a NMIS

Como parte de esta capacitación, vamos a implementar nuestro propio modelo juntos, paso a paso, para comprender mejor cómo se realiza el proceso. Ahora que sabemos cómo decodificar una MIB y obtener los datos que necesitamos incorporar al modelo usando snmpwalk, agreguemos un nuevo modelo de dispositivo a NMIS.

Para ilustrar el proceso, imaginemos que acabamos de obtener un nuevo servidor de Sun Microsystems y necesitamos crear un modelo para él. Sabemos que el OID del proveedor es: 1.3.6.1.4.1.42 (Sun), también, ejecutamos un snmpwalk al dispositivo y obtuvimos el SysDescr y SysObjectId, con la siguiente información:

```
SNMPv2-MIB::sysDescr.0 = STRING: SunOS opensolaris 5.11 snv_101b i86pc
SNMPv2-MIB::sysObjectID.0 = OID: UCD-SNMP-MIB::solaris
```

Con esta información podemos comenzar a agregar nuestro modelo a NMIS. Primero, agreguemos el OID del proveedor al archivo /usr/local/nmis8/conf /Enterprise.nmis.

conf/Enterprise.nmis

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

Luego, en /usr/local/nmis8/models/Model.nmis, agregamos el nombre del nuevo modelo al proveedor, si no existe, también tenemos que agregar el proveedor. En este caso, estamos utilizando una expresión regular para hacer coincidir la información obtenida de SysDescr con el nombre del modelo.

models/Model.nmis

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

Es importante resaltar que los nombres utilizados en estos 3 archivos deben coincidir para agregar con éxito el modelo a NMIS.

Este es un proceso simple y se aplica a dispositivos que usan su propio Agente SNMP, sin embargo, muchos proveedores ahora usan Net-SNMP como Agente y se deben tomar en cuenta algunas consideraciones.

Por ejemplo, ahora tenemos otro dispositivo de Sun Microsystems, snmpwalk devolvió esta información:

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

Aquí, sysObjectID muestra que el agente utilizado por el dispositivo es NET-SNMP y el OID que se le asignó es: 1.3.6.1.4.1.8072, una vez más, con esta información procedemos a seguir el proceso mencionado anteriormente.

- Agregar el vendedor

conf/Enterprise.nmis

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

- Actualice o agregue la información relacionada con este proveedor en Models.nmis

models/Model.nmis

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

Con estos cambios, nuestro modelo se cargará para el dispositivo.

Tip: Muchos dispositivos ahora usan NET-SNMP como Agente, por ende, es una buena idea copiar un modelo existente y adaptarlo a nuestras necesidades en lugar de crear uno nuevo.

Agregar una nueva métrica al Node Health

Hasta ahora, hemos podido crear nuestro nuevo modelo, ahora es el momento de incorporar nuevas métricas. Es muy común y útil mostrar métricas en Node Health (estado del nodo). Digamos que tenemos un enrutador Cisco y es capaz de proporcionar la cantidad de rutas que ve el enrutador, para simplificar las cosas, vamos a editar un modelo existente para dispositivos Cisco. Como sabemos, se necesita un MIB dump; se puede obtener ejecutando SNMPWALK contra el dispositivo, ya que ahora solo nos interesa la cantidad de rutas, nos centraremos en estos 2 elementos:

```
IP-FORWARD-MIB::ipCidrRouteNumber.0 = Gauge32: 33554432
IP-FORWARD-MIB::inetCidrRouteNumber.0 = Gauge32: 7
```

Es posible que necesitemos trabajar con los OID para estos MIBS, por lo que los traducimos y obtuvimos:

- ipCidrRouteNumber = "1.3.6.1.2.1.4.24.3”
- inetCidrRouteNumber = "1.3.6.1.2.1.4.24.6"

Ahora que tenemos la información necesaria, es hora de tomar algunas decisiones de implementación sobre los nombres a usar y los OID que usaremos como fuente de datos. En este caso, hemos decidido lo siguiente:

- Llamaremos la nueva métrica “routenumber”
- El nombre del MIB es ipCidrRouteNumber
- El OID es 1.3.6.1.2.1.4.24.3
- El gráfico se llamara "routenumber"

Para agregar la métrica al Node Health con éxito, necesitaremos realizar los siguientes cambios:

1. Agregar una entrada a nmis_mibs.oid (Opcional pero útil)
2. Agregarlo al modelo Model-CiscoRouter.nmis para comenzar a recopilar.
3. Agregar una entrada a: Common-header.nmis
4. Cree un gráfico para verlo: Graph-routenumber.nmis

Aunque es opcional, es posible que queramos asignar el OID a un nombre MIB; esto podría hacerse agregándolo a /user/local/nmis8/mibs/nmis_mibs.oid.

mibs/nmis_mibs.oid

```
--snip--
"ipDefaultTTL"      "1.3.6.1.2.1.4.2"
"ipCidrRouteNumber" "1.3.6.1.2.1.4.24.3"
"inetCidrRouteNumber"   "1.3.6.1.2.1.4.24.6"
--snip--
```

Se recomienda hacerlo de esta manera, sin embargo, no es necesario pero conveniente porque esta asignación estará disponible para todos los modelos como MIB, por lo que si queremos crear otros modelos no hay necesidad de usar el OID (1.3.6.1 .2.1.4.24.3) para referirnos a él, usamos el nombre MIB (" **ipCidrRouteNumber**") en su lugar.

Ahora, agregamos la métrica a Model-CiscoRouter.nmis, necesitamos recopilar y almacenar los nuevos datos en el RRD, para lograr eso, tenemos que agregar una nueva variable llamada "RouteNumber" a "system -> rrd -> nodehealth -> snmp ", y especificar los valores obtenidos por esta nueva variable se obtienen del nombre MIB“ ipCidrRouteNumber ”(OID: 1.3.6.1.2.1.4.24.3). Es importante tener en cuenta que la longitud **máxima** para el nombre de la variable es de **16 caracteres**.

models/Model-CiscoRouter.nmis

```
--snip--
  'system' => {
    'nodeModel' => 'CiscoRouter',
    'nodeType' => 'router',
    'nodegraph' => 'health,response,cpu,mem-router,ip,frag,buffer,modem,calls',
    'rrd' => {
      'nodehealth' => {
        'threshold' => 'cpu,mem-proc',
        'graphtype' => 'buffer,cpu,mem-io,mem-proc,mem-router’ ,
        'snmp' => {
          'avgBusy5' => {
            'oid' => 'avgBusy5'
          },
          --snip--
          'RouteNumber' => {
              'oid' => 'ipCidrRouteNumber'
                }
        },
      },
--snip--
```

Como se mencionó anteriormente, el mapeo del OID es opcional, podemos señalar los valores utilizados por nuestra nueva variable especificando el OID. Es útil usar "snmpObject" para dar un nombre como nota para que otros usuarios sepan con qué está relacionado el OID.

models/Model-CiscoRouter.nmis

```
--snip--
  'system' => {
    'nodeModel' => 'CiscoRouter',
    'nodeType' => 'router',
    'nodegraph' => 'health,response,cpu,mem-router,ip,frag,buffer,modem,calls',
    'rrd' => {
      'nodehealth' => {
        'threshold' => 'cpu,mem-proc',
        'graphtype' => 'buffer,cpu,mem-io,mem-proc,mem-router’ ,
        'snmp' => {
          'avgBusy5' => {
            'oid' => 'avgBusy5'
          },
          --snip--
          'RouteNumber' => {
                 'snmpObject' => 'ipCidrRouteNumber'
             'oid' => '1.3.6.1.2.1.4.24.3'
                }
        },
      },
--snip--
```

Introduction a las gráficas en NMIS

Las gráficas en NMIS son generadas por RRD (Round Robin Database). Los ficheros de definiciones de gráficos son parámetros que después se pasan como parámetros a RRD para la generación del grafico. En la operación collect en NMIS, se recolectan los datos que después RRD va a utilizar para mostrar los datos.

Los conceptos mas importantes de RRD son:

1. **DS**: Data Source
2. **DST**: Data Source Type:
    a. COUNTER
    b. DERIVE
    c. ABSOLUTE
    d. GAUGE

Utilizan notación NPR - Notación Polaca Inversa.

Personalizar las gráficas y sus elementos

En esta etapa también agregamos el nombre del gráfico, hemos decidido que nuestro gráfico se llamará "routenumber", por lo que lo agregamos al tipo de gráfico ("system -> rrd -> nodehealth -> graphtype"), ahora el modelo conoce el nombre del gráfico a utilizar. De la misma manera, necesitamos decirle al modelo que se debe mostrar el gráfico, para eso; agregamos el nombre del gráfico a "nodegraph" ("system -> nodegraph"). Las métricas se mostrarán manteniendo el mismo orden especificado en el "nodegraph", en este caso el gráfico se mostrará en la parte inferior, ya que lo agregamos al final de la lista, podemos cambiar el orden según sea necesario.

models/Model-CiscoRouter.nmis

```
--snip--
  'system' => {
    'nodeModel' => 'CiscoRouter',
    'nodeType' => 'router',
    'nodegraph' => 'health,response,cpu,mem-router,ip,frag,buffer,modem,calls,routenumber',
    'rrd' => {
      'nodehealth' => {
        'threshold' => 'cpu,mem-proc',
        'graphtype' => 'buffer,cpu,mem-io,mem-proc,mem-router,routenumber’ ,
        'snmp' => {
          'avgBusy5' => {
            'oid' => 'avgBusy5'
          },
          --snip--
          'RouteNumber' => {
              'oid' => 'ipCidrRouteNumber'
                }
        },
      },
--snip--
```

Agregamos el encabezado que el gráfico debe mostrar a: models/Common-header.nmis

models/Common-heading.nmis

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

Ahora tenemos que crear el gráfico, el nombre del archivo debe comenzar con "Graph-" seguido del nombre exacto que se eligió para el gráfico, en este caso "routenumber", por lo que el nombre del archivo del gráfico es: Graph-routenumber.nmis.

El archivo gráfico tiene diferentes secciones; cada una define cómo se debe mostrar el gráfico. NMIS puede mostrar 2 tamaños de gráficos: "standard" y "short", podemos especificar qué detalles mostrar en cualquier caso.

Tenga en cuenta que el "title" y "vlabel" (etiqueta vertical) usa la denominación "standard" y "short" de forma predeterminada para definir la longitud del elemento y la sección "option" usa "standard" y "small" para definir el tipo de gráfico, sin embargo, estas variables se pueden cambiar si es necesario.

models/Graph-routenumber.nmis

```
'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'
  ]
}
```

Examinemos la primera línea dentro de "option -> standard" para comprender cuál es el significado de cada término.

`DEF:`**`routes`**`=$database:`**`RouteNumber`**`:AVERAGE`

`DEF:<vname>=<rrdfile>:<ds-name>:<CF>`

vname: Variable interna donde se almacenarán los datos del RRD, en nuestro caso: "routes"

rrdfile: El RRD donde se almacenan los datos, en nuestro caso: $database (ya definido en Common-database.nmis)

ds-name: El nombre de la fuente de datos utilizado para almacenar datos particulares en el RRD, en nuestro caso "RouteNumber"

El ds-name(RouteNumber) debe ser exactamente el mismo que el nombre dado a la variable asignada en el modelo ("system -> rrd -> nodehealth -> snmp").

El siguiente elemento define el tipo de gráfico que se realizará con los datos.

`LINE1:`**`routes`**`#0000ff:Number of Routes`

`LINE:value[#color]:[legend]`

LINE1: Tipo de graficado, en nuestro caso: una línea.

Value: El valor o variable que contiene el valor, en nuestro caso: routes.

#color: Valor Hexadecimal del color, en nuestro caso: #0000ff (azul).

Legend: La leyenda asociada al valor.

Las dos últimas líneas son similares, son los valores calculados que se muestran como parte de la leyenda.

Básicamente, el gráfico se define en función del sistema de gráficos de la herramienta RRD y puede encontrar información adicional y mayores opciones aquí: https://oss.oetiker.ch/rrdtool/doc/rrdgraph.en.html

3- Implementación de un nuevo dispositivo (II)

System Section

Hemos agregado una nueva métrica a la sección nodeHealth, pero a veces necesitamos agregar nuevos conceptos al modelo de dispositivo y crear sus propios archivos RRD.

Como lo hemos hecho anteriormente, necesitamos tomar algunas decisiones de implementación:

- ¿Qué datos recopilar?

- El nombre MIB a usar es: tcp
- La colección se llamará: tcp
- Los gráficos serán llamados: “tcp-conn” y “tcp-segs”

Para agregar la nueva colección a la sección "system" con éxito, necesitaremos realizar los siguientes cambios:

1. Agregue la sección "tcp" a la sección "system" del modelo
2. Agregue una entrada a: Common-header.nmis.
3. Agregue una entrada a: Common-database.nmis
4. Cree los gráficos: Graph-tcp-conn.nmis y Graph-tcp-segs.nmis
5. Agreguar nuevas MIBS a nmis_mibs.oid (Opcional)

Hicimos un SNMPWALK y obtuvimos la siguiente información relacionada con el tcp. Es importante notar que este MIBS es plano, solo hay un valor por MIB, lo que permite agregarlo a la sección "system".

```
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
```

Para ilustrar mejor este proceso, utilizaremos un modelo ya existente llamado: Model-net-snmp.nmis. Ahora agregamos la nueva sección llamada "tcp" en "System -> rrd".

Dentro de la sección "tcp", agregamos el "graptype", listando los nombres de los gráficos que decidimos anteriormente. Además, tenemos que agregar una nueva sección "snmp".

models/Model-net-snmp.nmis

```
  'system' => {
    --snip--
    'rrd' => {
      --snip—
      'tcp' => {
        'graphtype' => 'tcp-conn,tcp-segs',
        'snmp' => {
        }
      }
    }
  }
```

Ahora agregamos el OID a recopilar dentro de la nueva sección snmp, como se mencionó anteriormente, podemos asignar el OID a un nombre MIB en el archivo nmis_mibs.oid o simplemente podemos usar el OID numérico.

Además, agregamos el elemento "option", que indicará al rrd si los datos son un contador (Counter) o un indicador (Gauge), y los límites inferior y superior, siendo "U" equivalente a Ilimitado (Unlimited).

Cuando no especificamos ‘option’ por defecto: Se utiliza GAUGE,0:U. Vemos un ejemplo de como son los valores que se guardan en RRD dependiendo del valor:

```
Values       = 300, 600, 900, 1200
Step         = 300 seconds
COUNTER DS   =    1,  1,   1,    1
DERIVE DS    =    1,  1,   1,    1
ABSOLUTE DS  =    1,  2,   3,    4
GAUGE DS     = 300, 600, 900, 1200
```

Si hemos definido un rrd con una opción incorrecta, lo podemos cambiar. Pero tenemos que tener en cuenta, que si este gráfico ya se ha creado, necesitamos regenerarlo. Para ello, se puede emplear la herramienta de NMIS admin/rrd_tune. Existen varios tipos dependiendo de las métricas.

Para descubrir estos detalles, tenemos que observar la salida del SNMPWALK, esto también nos ayudará a obtener los nombres utilizados en cada OID. En nuestro caso, el MIBS muestra que el elemento "tcpCurrEstab" es un indicador (gauge).

models/Model-net-snmp.nmis

```
'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' => ‘1.3.6.1.2.1.6.15’,
          'snmpObject' => 'tcpOutRsts',
          'option' => 'counter,0:U'
        }
      }
    }
  }
}
```

A continuación, tenemos que agregar los encabezados de nuestros nuevos gráficos en Common-header.nmis, estos son los encabezados que se mostrarán al mostrar gráficos en NMIS. Si no está definido, verá un mensaje como este: "heading not defined in Model"

models/Common-heading.nmis

```
--snip--
 'heading' => {
   'graphtype' => {
      --snip--
      'tcp-conn' => 'TCP Connections',
      'tcp-segs' => 'TCP Segments'
   }
 }
--snip--
```

Ahora tenemos que agregar una nueva entrada a Common-database.nmis, es importante que coincida con el nombre utilizado en Common-database.nmis y el nombre de la nueva sección.

models/Common-database.nmis

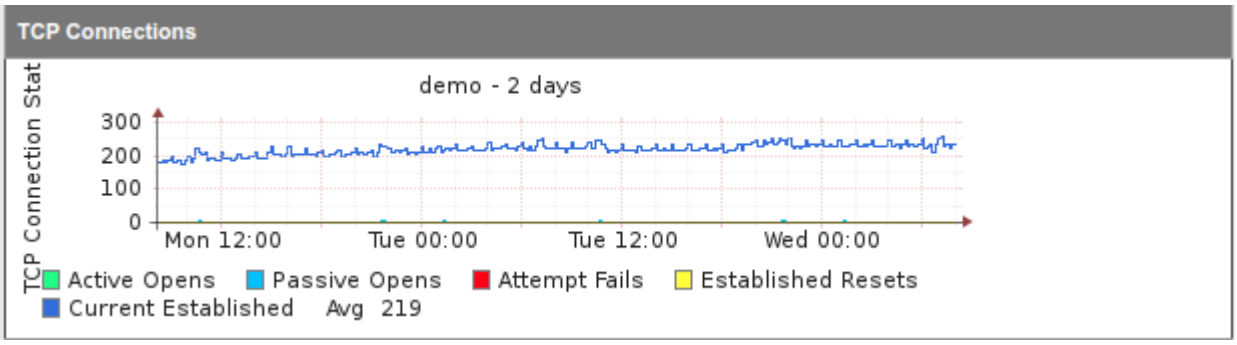
```
--snip--
'tcp' => '/nodes/$node/health/tcp.rrd',
--snip--
```

Nuestro último paso es hacer gráficos significativos con los datos recopilados. Es el mismo proceso que ya discutimos al agregar una métrica. En este caso, se agregaron más variables al gráfico y se utilizan áreas y pilas en lugar de líneas. Aquí está la implementación de Graph-tcp-conn.nmis.

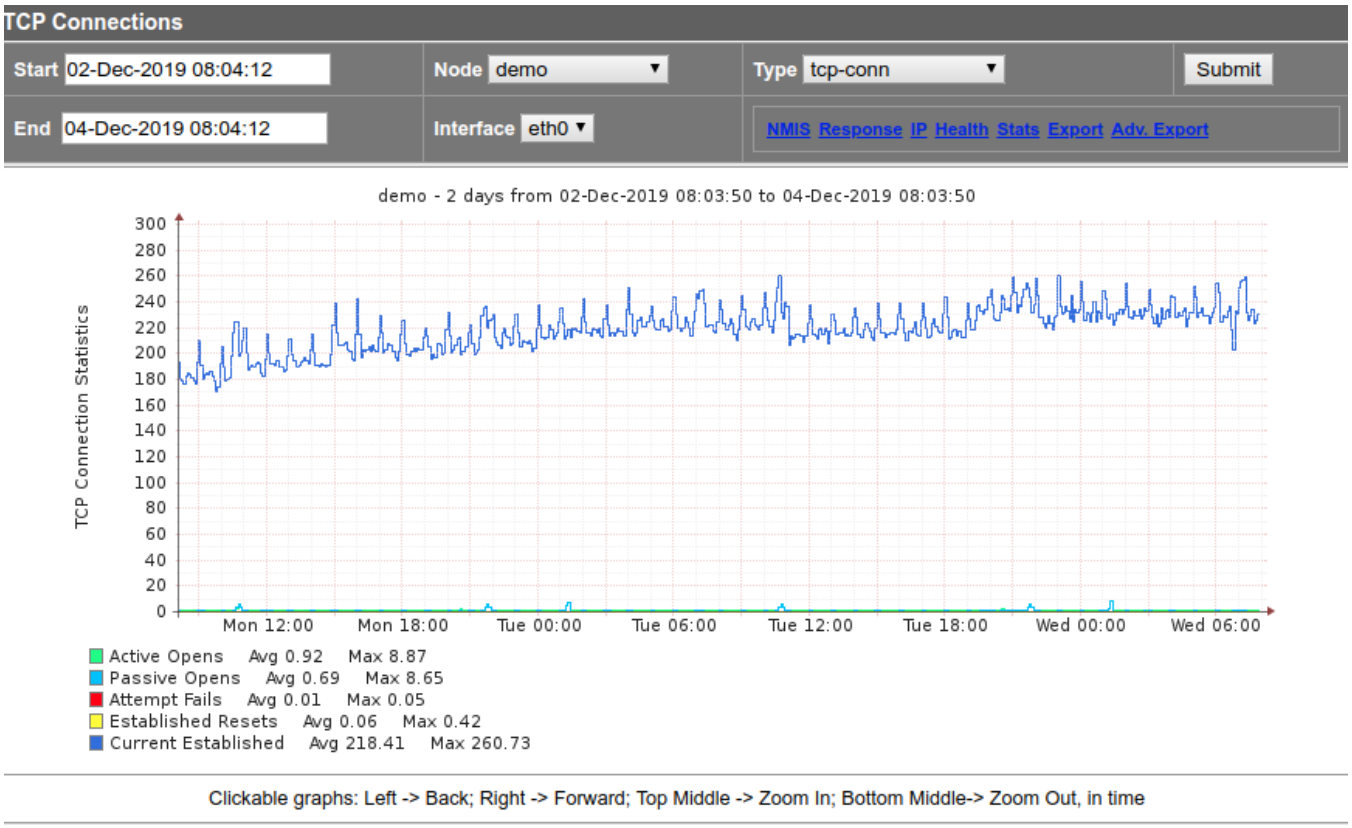
models/Common-heading.nmis

```
'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',
  ]
}
```

Finalmente, la versión pequeña del gráfico debería verse así:



Así es como se ve la versión estándar:



Sección "SystemHealth" (Indexes)

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.

This time we need to add Disk IO information as a table to our model. The net-snmp model already has a section called "diskIOTable", however, we will be using its development as an example.

Once again, we need to take some implementations decisions:

A menudo, una sección de datos que es útil mostrar en NMIS se presenta en SNMP como una tabla. Para modelar estos datos, NMIS ofrece la sección "systemHealth" que permite utilizar datos de SNMP indexados.

Esta vez necesitamos agregar información de DiskIO como tabla a nuestro modelo. El modelo net-snmp ya tiene una sección llamada "diskIOTable", sin embargo, utilizaremos su desarrollo como ejemplo.

Una vez más, debemos tomar algunas decisiones de implementación:

- ¿Qué tabla de datos se va a recopilar?
- El nombre de la MIB es diskIOTable.

- La colección se llamará diskIOTable.

Tendremos que hacer los siguientes cambios:

1. Agregar la sección systemHealth al modelo.
2. Agregar cualquier MIBS adicional a nmis_oids.nmis (opcional)
3. Agregar una entrada a: Common-header.nmis
4. Agregar una entrada a: Common-database.nmis
5. Crear los gráficos necesarios : Graph-diskio-rw.nmis y Graph-diskio-rwbytes.nmis
6. Agregar la nueva sección diskIOTable a model_health_sections en Config.nmis

Esta es la información obtenida con SNMPWALK. Es importante notar que este MIBS está indexado, lo que hace posible el uso de tablas en la sección SystemHealth.

```
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
```

Aquí tenemos una construcción de tabla SNMP estándar. Para cada disco que tenemos, hay un índice asignado y cada disco tendrá "diskIOIndex, diskIODevice, diskIONRead, etc.".

Archivo del modelo

La sección "systemHealth" es una sección "top-level", lo que significa que no esta agregada a ninguna otra sección.

```
%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 --
);
```

En la parte superior de la sección **systemHealth**, el item "**sections**" le dice a NMIS qué secciones mostrar en la GUI (incluido en el listado del menu desplegable: **systemHealth**). Este nombre debe coincidir con el nombre de la sección a continuación.

En este ejemplo se usa `"diskIOTable"` .

sección sys

En el fragmento de código anterior vemos la sección '**sys**', aquí es donde se definen los datos que se almacenarán en el archivo Nodename-node.nmis. También se definen los datos necesarios para recopilar la sección RRD. Si desea ver el último valor reunido por NMIS para estos MIBS, se debe revisar Nodename-node.nmis para el nodo. Los valores definidos dentro de la sección snmp se hacen de la misma manera como se ha hecho en otras partes del modelo.

La sección sys-> diskIOTable especifica 2 items:

1. `indexed => 'diskIOIndex'`
   Esto le dice a NMIS que el OID diskIOIndex mantendrá el valor del índice a utilizar durante cada iteración o bucle.
2. `headers => 'diskIODevice'`
   Esto le dice a NMIS que al mostrar los índices, se debe mostrar una columna para diskIODevice. Si la directiva de encabezados enumera más de una clave, entonces todas las columnas correspondientes estarán presentes; en el ejemplo anterior también podríamos haber incluido la columna diskIOIndex (pero el nombre del dispositivo es más útil en este caso).

3. `snmp => {}`
   Esto le dice a NMIS qué datos recopilar.
   3.a `oid` Puede ser un nombre de la lista de nmis_mibs.oid o un OID.
   3.b `title` Titulo a mostrar.

Las versiones de NMIS anteriores a 8.4.8 requieren que enumere la asociación de nombre a OID sin procesar en mibs/nmis_mibs.oid; para el ejemplo anterior se requerirían entradas para diskIODevice y diskIOIndex. Para simplificar el proceso de modelado, 8.4.8 y versiones posteriores permiten el uso del item: index_oid que le permite asociar un oid sin procesar con un nombre en un archivo de modelo. Usando eso, el ejemplo se vería así:

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

Otra característica en 8.4.8 y posterior, es index_regex, que permite la indexación de elementos múltiples: normalmente las tablas SNMP son indexadas por el último componente OID numérico único. Cuando NMIS realiza una actualización en una entidad indexada, itera a través de todos los valores conocidos para este componente de índice y los registra. Esta iteración no funciona si el índice consta de más de un número, como lo hace en ciertos equipos. En tales casos, puede establecer index_regex para un valor que capture los componentes OID que varían entre los elementos de la tabla. El siguiente ejemplo asegura que los últimos tres números se usen para el indexado.

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

sección rrd

La sección RRD define qué datos se recogen y almacenan en los RRDs. Una vez más, los valores definidos dentro de la sección snmp son como en cualquier otra parte del modelo.

La sección rrd->diskIOTable especifica 3 items:

1. `'control' => 'CVAR=diskIODevice;$CVAR =~ /sda|sr|disk|xvda|dm\-/',`
   Esto le dice a NMIS que el OID diskIODevice debe ser revisado y sólo capturar los valores en RRD si coinciden con la expresión regular determinada.
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.
   Le dice a NMIS que esta es una tabla indexada, para luego utilizar el índice especificado en la sección "sys" anterior para iterar.

3. `graphtype => 'diskio-rw,diskio-rwbytes'`
   Lista los graph-types associados al los RRD's para ser graficados.

4. `snmp => 'diskio-rw,diskio-rwbytes'`
   Nombra los elementos que serán colectados.
   4.a `oid` Puede ser un nombre de la lista de nmis_mibs.oid o un OID.
   4.b `option` Indica si la data obtenida es del tipo "counter" o "indicador" y los limites superiores e inferiores.
   e.g: `'option' => 'counter,0:U'`
   4.c `title` Titulo a mostrar.

Common-heading.nmis

Estos son los encabezados que verá cuando muestre el gráfico en varias pantallas en NMIS. Si no está definido, verá un mensaje como este: "heading not defined in Model".

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

Common-database.nmis

El nombre del rrd se especifica en este archivo. Se pude crear un nuevo set de archivos rrds para esta nueva sección, para hacerlo simplemente se debe agregar una nueva línea en el common-database.

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

Como puede ver, el nombre del archivo tiene **$index** en el nombre, por lo que NMIS creará un nuevo archivo para cada índice que esté recopilando utilizando esta variable.

Gráficas en la sección SystemHealth

Graph-diskio-rw.nmis y Graph-diskio-rwbytes.nmis

Estos son los archivos utilizados para definir los gráficos. Definimos el "DEF" de RRD en función de lo que almacenó, definimos la LÍNEA o ÁREA a graficar, usamos algunos GPRINTS para la salida de texto y otra sintaxis de RRD para lograr el gráfico deseado.

```
'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',
  ]
}
```

Una vez realizado todo este procedimiento, las nuevas secciones aparecerán en el menú desplegable "System Health".

Personalizando las Gráficas

Estos son alguno de los valores disponibles en RRD para personalizar las gráficas;

Nombre	**Formato**
PRINT	*vname*:*format*[**:strftime**|**:valstrftime**|**:valstrfduration**]
GPRINT	*vname*:*format*
COMMENT	Text

LINE	[*width*]**:***value*[*#color*][**:**[*legend*][**:STACK**][**:skipscale**][**:dashes**[**=***on_s*[,*off_s*[,*on_s*,*off_s*]...]][**:dash-offset=***offset*]]]
AREA	*value*[*#color*][**:**[*legend*][**:STACK**][**:skipscale**]]
TICK	*vname#rrggbb*[*aa*][**:***fraction*[**:***legend*]]
TEXTALIGN	**:**{**left**|**right**|**justified**|**center**}
SHIFT	*vname***:***offset*

4- Implementación de Umbrales y Alertas

Introduction al Thresholding

NMIS8 incluye potentes funciones para el rendimiento y umbral operativo, que mejoran en gran medida las facultades de gestión de la red. Estos umbrales o thresholds dan como resultado alertas / eventos / notificaciones que NMIS puede enviar cuando ve cumpla sobrepase algún limite establecido. Los umbrales tienen controles muy granulares que, por defecto, se han configurado de manera bastante amplia.

Consideraciones e implementación de Umbrales

Consideraciones

El threshold se puede conseguir con los siguientes pasos:

- ¿Qué datos se recopilan que se pueden medir?
- Agregar las propiedad del threshold (limites) a la sección del modelo.
- Agregar los valores del threshold al archivo Common-threshold.nmis.
- Agregar la extracción de estadísticas (extraer datos del archivo RRD y darles el formato adecuado) al archivo Common-stats.nmis.
- Testear el nuevo threshold implementado.
- Considerar la implementacion de thresholds avanzados, usando controles (Agregando lógica booleana y expresiones regulares).

Considere las siguientes preguntas:

- ¿Qué tan factible seria el aplicar un threshold al elemento necesario?
- ¿Se pueden reducir / traducir / combinar las métricas en un threshold significativo?
- ¿Cuál debería ser el nombre del evento correspondiente para el threshold?
- El nombre del evento debe incluir "Proactive" al principio para que NMIS lo procese correctamente. ejm: "**Proactive** Temp" or **Proactive** CPU Load".

Implementación

Para implementar el threshold, primero tenemos que agregar la propiedad del threshold a la sección del modelo, en este caso la llamamos "env_temp", en el siguiente paso usaremos este nombre para vincular el modelo con el Common-threshold.nmis y Common-stats.nmis.

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

Agregamos los valores del threshold a Common-threshold.nmis, usando el nombre especificado anteriormente. El nombre del evento debe incluir "Proactive" al principio

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

A continuación, agregamos extracción de estadísticas a Common-stats.nmis

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

Once we have created the threshold, it is time to tested. The best way to test if it is working as desired, is by using nmis.pl.

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

Una vez que hemos creado el threshold, es hora de probarlo. La mejor manera de probar si funciona como se desea, es utilizando nmis.pl.

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

Para probarlo, ejecutamos "nmis.pl type=thresholds", y verificamos que los eventos se hayan creado. Después de eso, restablecemos el valor a su estado anterior, ejecutándose una vez más "nmis.pl type=thresholds", el evento debería cerrarse ahora.

Standard Thresholds (Comunes)

NMIS incluye un conjunto de thresholds estándar que se asocian comúnmente a algunos proveedores. Este es un resumen de estos thresholds.

Nombre del Threshold	Evento	Proveedor
available	Proactive Interface Availability	Common for all Vendors
calls_util	Proactive Calls Utilisation	Cisco
ccpu	Proactive CPU	Cisco
cpu	Proactive CPU	Cisco (the most common for Cisco devices)
cpuUtil	Proactive CPU	Alcatel, Zyxel

cpu_cpm	Proactive CPU	Cisco
env_temp	Proactive Temp	Cisco, Zyxel
hrsmpcpu	Proactive CPU	Microsoft
jnx_buffer	Proactive Buffer Utilisation	Juniper
jnx_cpu	Proactive CPU	Juniper
jnx_heap	Proactive Heap Utilisation	Juniper
jnx_temp	Proactive Temp	Juniper
mem-proc	Proactive Memory Free	Cisco
memUtil	Proactive Memory Utilisation	Alcatel, Zyxel
modem_dead	Proactive Dead Modem	Cisco
modem_unav	Proactive Modem Utilisation	Cisco
pkt_discards_in	Proactive Interface Discards Input Packets	Common for all Vendors
pkt_discards_out	Proactive Interface Discards Output Packets	Common for all Vendors
pkt_errors_in	Proactive Interface Error Input Packets	Common for all Vendors
pkt_errors_out	Proactive Interface Error Output Packets	Common for all Vendors
reachable	Proactive Reachability	Common for all Vendors
response	Proactive Response Time	Common for all Vendors
ssCpuRawIdle	Proactive CPU IO Idle	net-snmp (Linux, Solaris, etc)
ssCpuRawSystem	Proactive CPU IO System	net-snmp (Linux, Solaris, etc)
ssCpuRawUser	Proactive CPU IO User	net-snmp (Linux, Solaris, etc)
ssCpuRawWait	Proactive CPU IO Wait	net-snmp (Linux, Solaris, etc)
util_in	Proactive Interface Input Utilisation	Common for all Vendors
util_out	Proactive Interface Output Utilisation	Common for all Vendors

Crear Thresholds (Detallado)

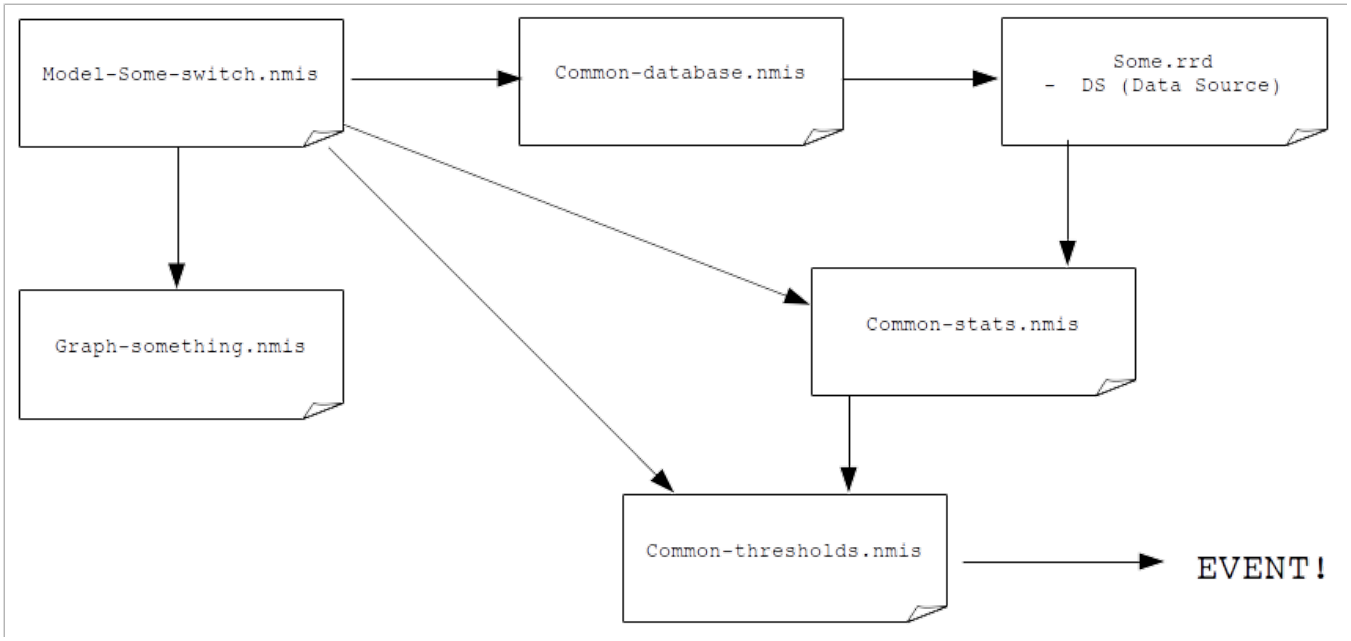
Archivos

Los archivos que deben de ser modificados son:

- /usr/local/nmis8/models/Model-Some-switch.nmis
- /usr/local/nmis8/models/Common-database.nmis
- /usr/local/nmis8/models/Common-header.nmis
- /usr/local/nmis8/models/Common-stats.nmis
- /usr/local/mmis8/models/Common-threshold.nmis

Relación

Relación de los archivos entre sí, puede ser útil visualizar cómo interactúan.



Atributos Comunes

Hay varios atributos comunes que deben coincidir entre estos archivos para que el threshold funcione adecuadamente. En un intento por demostrar la relación entre estas variables, utilizaremos las siguientes etiquetas.

alpha - Esta es la variable del threshold definida en el archivo Model-Some-switch.nmis y debe colocarse en la sección del nombre del Common-threshold nmis

bravo - Esta es la variable `graphtype` definida en el archivo Model-Some-switch.nmis, la cual establece el nombre del gráfico que NMIS invocará para un archivo RRD particular. Para que el encabezado del gráfico sea correcto, esta variable con su descripción asociada debe agregarse al archivo Common-header.nmis.

charlie - Este es el nombre del archivo RRD en el que se escribirán los datos obtenidos a través de SNMP. Esta variable se define en el archivo Model-Some-switch.nmis. También se le debe hacer referencia en el archivo Common-database.nmis para que el RRD se cree y actualice en el directorio correcto. Se hace referencia a esta variable en el archivo Common-stats.nmis para que NMIS sepa dónde encontrar datos estadísticos.

delta - Variable relacionada específicamente a un SNMP OID y se define en el archivo Model-Some-switch.nmis. Este debe ser el título 'DS' (Data Source) dentro del archivo RRD que contendrá los datos reales obtenidos para el OID específico. Se hace referencia a esta variable en el archivo Common-stats.nmis para proporcionar el DS para los datos específicos que deben presentarse.

echo - Esta variable se establece en el archivo Common-stats.nmis. Se utiliza para realizar cálculos sobre la variable "zebra" en el lenguaje RRD. Esta variable luego se pasa al archivo Common-threshold.nmis para activar y enviar el evento correspondiente.

```
### Model-Some-Switch.nmis

%hash = (

     'systemHealth' => {

          'rrd' => {

               '<charlie>' => {

                    'graphtype' => '<bravo>',

                    'indexed' => 'true',

                    'threshold' => '<alpha>'

                    'snmp' => {

                         '<delta>' => {

                              'oid' => 'hrProcessorLoad',

                              'option' => 'gauge,0:U'

                         }

                    }

               }

          }

     }

)
```

```
### Common-database.nmis

'<charlie>'   =>   '/nodes/$node/health/<charlie>-$index.
rrd',
```

```
### Common-threshold.nmis

%hash = (

     'threshold' => {

          'name' => {

               '<alpha>' => {

                    'item' => '<echo>',

                    'event' => 'Proactive CPU',

                    'select' => {

                         'default' => {

                              'value' => {

                                   'fatal' => '90'

                                   'critical'     =>
                                   '80',

                                   'major' => '70',

                                   'minor' => '60',

                                   'warning' => '50'

                              }

                         }

                    }

               }

          }

     }

)
```

```
### Common-stats.nmis

%hash = (

     'stats' => {

          'type' => {

               '<charlie>' => {

                    'DEF:<echo>=$database:<delta>:
                    AVERAGE',

                    'PRINT:<echo>:AVERAGE:<echo>=%1.0f',

               }

          }

     }

)
```

```
### Common-heading.nmis

'<bravo>' => 'Processor Load',
```

5. Implementación de umbrales de NMIS (II)

Thresholds Simples y Avanzados

Thresholds Simples

En NMIS, un threshold simple se define de la siguiente manera:

- El nombre
- Nombre del evento (el cual debe empezar con la palabra "Proactive" para poder ser procesado correctamente).
- Un Select
- Los valores por defecto del threshold.

En el archivo /usr/local/nmis8/models/Common-threshold.nmis, se muestra de la siguiente manera:

```
'cpu' => {
  'item' => 'avgBusy5min',
  'event' => 'Proactive CPU',
  'select' => {
    'default' => {
      'value' => {
        'critical' => '70',
        'fatal' => '80',
        'minor' => '50',
        'warning' => '40',
        'major' => '60'
      }
    }
  }
},
```

Set de limites para el thresholds del "Core CPU"

Sin embargo, los Core Devices son más sensibles a la carga de la CPU. Por lo tanto, queremos usar un conjunto diferente de valores en el threshold. Algo como:

'critical' => '60',

'fatal' => '70',

'minor' => '40',

'warning' => '30',

'major' => '50'

Pero, ¿cómo hacer que esto solo se aplique a los Core Devices?

Controles Avanzados en el Thresholding

Por ejemplo, diferentes thresholds para Core Devices. Revisar en Common-thresholds le dará algunas idea de como se aplican, pero se pueden agregar muchas "selecciones" y tener propiedades como:

- $name
- $node
- $host
- $group
- $roleType
- $nodeModel
- $nodeType
- $nodeVendor
- $sysDescr
- $sysObjectName
- others for interface
- Almost unlimited possibilities.

Por lo tanto, podemos crear un threshold más específico para Core Devices (NMIS ya lo tiene configurado de forma predeterminada).

```
'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'
      }
    }
  }
},
```

Estos se ejecutan en el orden definido en la sección "select" y si no existe una coincidencia en el control, se utilizara el threshold por default.

Opciones de control avanzado

Las siguientes son las opciones de control disponibles:

Propiedades del nodo:

- $name
- $node
- $host
- $group
- $roleType
- $nodeModel
- $nodeType
- $nodeVendor
- $sysDescr
- $sysObjectName

Objetos Indexados como interfaces:

- $ifAlias
- $Description
- $ifDescr
- $ifType
- $ifSpeed
- $ifMaxOctets
- $maxBytes
- $maxPackets
- $entPhysicalDescr

Objetos indexados recientemente agregados en NMIS 8.6G

- $hrStorageDescr
- $hrStorageType
- $hrStorageUnits (disk block size)
- $hrStorageSize (disk size in blocks)
- $hrStorageUsed (disk used in blocks)
- $hrDiskSize (disk size in bytes, hrStorageSize * hrStorageUnits)
- $hrDiskUsed (disk used in bytes, hrStorageUsed * hrStorageUnits)
- $hrDiskFree (disk free in bytes)

Ejemplos de Controles

Los controles son pequeñas piezas de código que se evaluarán cuando sea necesario, por lo que es posible que desee hacer alguno de los siguientes.

Resultado	Control
Usar este threshold si la velocidad de la interface se encuentra entre 1 y 5 megabits/segundo	$ifSpeed <= 50000 and $ifSpeed >= 10000
Usar este threshold si la velocidad de la interface es de 100 megabits	$ifSpeed == 100000000
Usar este threshold si la velocidad de la interface es de 10 megabits	$ifSpeed == 10000000
Usar este threshold si la velocidad de la interface es de 1 gigabits	$ifSpeed == 1000000000
Usar este threshold si el tamaño del disco es mayor a 100 gigaytes	$hrDiskSize >= 104857600000
Usar este threshold para todos los dispositivos con dirección IP 192.168	$host =~ /192\.168/
Usar este threshold para todos los dispositivos en el grupo "Sales"	$group eq "Sales"
Usar este threshold para todos los dispositivos Cisco IOS	$sysDescr =~ /Cisco IOS/

Alertas Básicas

Una alerta es un evento personalizado generado al testear el valor de una OID o de una variable personalizada y producir un resultado booleano (verdadero o falso). Si el test devuelve "verdadero", se genera un evento y se ejecutará a través del sistema de escalado, falso no generará una alerta. Más tarde, cuando el test que resulto en "verdadero", devuelve ahora "falso", el evento se cerrara.

Los valores requeridos para la alerta son:

test => La operación booleana usando $r para determinar si la alerta debe ser generada.
event => Nombre que se le dará al evento cuando se genere.
level => Nivel de prioridad que se le dará al evento cuando se genere.

Test

Esta puede ser cualquier expresión en Perl, y su resultado de evaluación se interpretará tal como perl procesa booleanos (es decir, un **string vacío, 0 y undef** significa falso, cualquier otra cosa significa verdadero).

$r contiene el valor del oid y debe usar $r para comprobar la condición para la que desea generar la alerta. Todos los operadores de Perl están disponibles:

```
# Números
"==" devuelve verdadero si el argumento izquierdo es numéricamente igual al argumento del lado derecho.
"! =" devuelve verdadero si el argumento izquierdo no es numéricamente igual al argumento del lado derecho.
"<" devuelve verdadero si el argumento izquierdo es numéricamente menor que el argumento del lado derecho.
">" devuelve verdadero si el argumento izquierdo es numéricamente mayor que el argumento del lado derecho.
"<=" devuelve verdadero si el argumento izquierdo es numéricamente menor o igual que el del lado derecho.
"> =" devuelve verdadero si el argumento izquierdo es numéricamente mayor o igual que el del lado derecho.

# Texto
"eq" devuelve verdadero si el argumento izquierdo es textualmente igual al argumento del lado derecho.
"ne" devuelve verdadero si el argumento izquierdo no es textualmente igual al argumento del lado derecho.
"lt" devuelve verdadero si el argumento izquierdo es textualmente menor que el argumento del lado derecho.
"gt" devuelve verdadero si el argumento izquierdo es textualmente mayor que el argumento del lado derecho.
"le" devuelve verdadero si el argumento izquierdo es textualmente menor o igual que el argumento del lado
derecho.
"ge" devuelve verdadero si el argumento izquierdo es textualmente mayor o igual que el argumento del lado
derecho.
```

A quick note on stringwise versy numerical comparison: in numeric mode, the expressions will be converted to numbers (i.e. string "0003" becomes the number 3 for comparison). In string mode the expressions' characters are compared one by one. If $r is "0003", $r == 3 is true, but $r eq 3 is false.

Una nota rápida sobre la comparación numérica versus la comparación textual: en modo numérico, las expresiones se convertirán en números (es decir, el texto "0003" se convierte en el número 3 para la comparación). En modo textual, los caracteres de las expresiones se comparan uno por uno. Si $r es "0003", $r == 3 es verdadero, pero $ r eq 3 es falso.

¿Dónde agregar la alerta?

La alerta se puede agregar a las variables actuales que se están poleando desde los dispositivos, o se puede agregar una nueva sección. Por ejemplo, se podría agregar una nueva sección en Model-> system-> sys que podría verse como en el siguiente ejemplo ("--snip--" indica que se ha eliminado parte del código del modelo irrelevante para el ejemplo para mayor claridad):

```
%hash = (
  --snip--
  'system' => {
     --snip--
    'sys' => {
      'standard' => {
        --snip--
      },
      'alerts' => {
        'snmp' => {
          'tcpCurrEstab' => {
            'oid' => 'tcpCurrEstab',
            'title' => 'TCP Established Sessions',
            'alert' => {
              'test' => '$r > 250',
              'event' => 'High TCP Connection Count',
              'level' => 'Warning'
            }
          }
        }
      },
       --snip--
    }
    --snip--
  }
  --snip--
);
```

Adding the alert also adds the information to the "Device Details" panel, so you get the last polled value displayed all the time. Note that when you add such a basic alert its variable is collected independently of any other variables that your model might collect.

Al agregar la alerta también se agrega la información al panel "Device Details", para que se muestre el último valor obtenido todo el tiempo. Se debe tener en cuenta que cuando agrega una alerta así de básica, su variable se recopila independientemente de cualquier otra variable que pueda recopilar el modelo.

TCP Established Sessions	106

Ejemplo

The following is an example of the layout of an alert (in this example serialNum is taken from Model-CiscoRouter.nmis) and uses a string based (stringwise) comparison:

El siguiente es un ejemplo del diseño de una alerta (en este ejemplo, serialNum se toma del Model-CiscoRouter.nmis) y utiliza una comparación textual (stringwise):

```
'serialNum' => {
  'oid' => 'chassisId',
  'title' => 'Serial Number',
  'alert' => {
    'test' => '$r ne "SomeSerialNumber"',
    'event' => 'Serial Number Invalid',
    'level' => 'Critical'
  }
}
```

$r en este caso es el valor del OID chassisId. Esto generará el evento "ALERTA: Serial Number Invalid" cuando el oid chassisId no sea igual a "SomeSerialNumber".

También se puede utilizar una comparación numérica, en este caso cuando el valor es 0, se genera la alerta, cuando el valor no es 0, la alerta se borra:

```
'cipSecGlobalActiveTunnels' => {
  'oid' => 'cipSecGlobalActiveTunnels',
  'title' => 'Global Active Tunnels',
  'alert' => {
    'test' => '$r == 0',
    'event' => 'No tunnels present',
    'level' => 'Critical'
  }
}
```

Alertas más avanzadas

Las alertas también se pueden crear en la sección 'alerts' del modelo. los permisos creados en esta sección tienen la ventaja de poder usar valores de una sección completa de datos para determinar si la alerta debe activarse o no; sin embargo, tales alertas NO pueden acceder a las variables colectadas /modeladas en la sección 'system' y, como tales, son principalmente útiles para el modelado de systemHealth.

Si el modelo no tiene la sección 'alerts', se puede agregar en el nivel más externo del modelo, al mismo nivel de '-common-' y 'system'.

Un ejemplo concreto siempre hace las cosas mas claras:

```
%hash = (
 '-common-' => {
   -- snip --
 },
 'system' => {
   -- snip --
 },
 'storage' => {
   -- snip --
 },
 'alerts' => {
   'services' => {
     'HighProcessMemoryUsage' => {
       'type' => 'test',
       'test' => 'CVAR1=hrSWRunPerfMem;$CVAR1 > 300000',
       'value' => 'CVAR1=hrSWRunPerfMem;$CVAR1 * 1',
       'unit' => 'KBytes',
       'element' => 'hrSWRunName',
       'event' => 'High Process Memory Usage',
       'level' => 'Warning'
     }
   }
 }
);
```

Analicemos el ejemplo anterior.

- 'services' define de qué sección se toman los valores que se utilizan para la alerta. En este caso, "services" no se encuentra en este modelo porque es una sección especial definida solo para servidores. Normalmente no tendrá que preocuparse por secciones especiales. ¡Tenga en cuenta que NO PUEDE usar la sección 'system' para alertas avanzadas!
- 'HighProcessMemoryUsage': Crea la etiqueta o id para la alerta.
- 'type' => 'test': Especifica que la alerta se generara a partir de una sola condición. Las opciones son: ['test', 'threshold-rising', 'threshold-falling']
- 'test' => 'CVAR1=hrSWRunPerfMem;$CVAR1 > 300000' define una variable personalizada y luego la utiliza para ejecutar una operación booleana.
  - Consulte el siguiente párrafo para mayor información sobre variables personalizadas.

- 'value' => 'CVAR1=hrSWRunPerfMem;$CVAR1 * 1': Define como el valor que activa la alerta debe ser reportado y mostrado en la GUI.
- 'unit' => 'KBytes': La unidad con la que se mostrará el valor anterior.
- 'element' => 'hrSWRunName': El OID / valor que genera la alerta, un descriptor o identificador. En este caso, muestra el nombre del proceso que tiene un uso elevado de memoria.
- 'event' => 'High Process Memory Usage': Establece el nombre del evento generado por la alerta.
- 'level' => 'Warning': Nivel con el que se activará el evento. Cuando se usa thresholding, esto no se aplica ya que el propio threshold definen el nivel.

Variables personalizadas

Tenga en cuenta que en las versiones de NMIS anteriores a 8.6 solo se puede usar una variable personalizada en el test expression conocida como CVAR. Esta limitación se ha eliminado en NMIS 8.6, esta limitación nunca se aplicó a las expresiones de valor o control.

Como se describe con más detalle en la sección de Modelado avanzado, sus expresiones de prueba, control y valor pueden contener definiciones y accesos de variables personalizadas. La sintaxis es sencilla:

`CVAR=snmp_var_name;` o `CVAR3=other_snmp_var;` busca por el valor de un objeto snmp (para el cual se hace la recolección de datos en el modelo) y lo almacena en una de las 11 variable personalizadas disponibles (CVAR y CVAR0 al CVAR9). Cuando se desee acceder al valor, se debe usar: `$CVAR` , `$CVAR3` etc.

Please note that the substitution is performed on a purely textual basis before perl is given the expression to evaluate; if you want to use string variable contents you should double-quote your access expressions, e.g.

Tenga en cuenta que la sustitución se realiza sobre una base puramente textual antes de que perl reciba la expresión para evaluar; Si desea utilizar contenidos de manera textual, se debe poner entre comillas.

```
CVAR2=ifAlias; "$CVAR2" =~ /some description/
```

Depurando un Threshold

Veamos un ejemplo:

./nmis.pl type=threshold debug=1 node=thor

```
22:51:26 init, info of node=thor loaded
22:51:26 init, no loading of cfg of node=thor
22:51:26 init, loading model Model-net-snmp for node thor
22:51:26 loadModel, INFO, model Model-net-snmp loaded (from cache)
22:51:26 doThreshold, Starting Thresholding node=thor
22:51:26 runThrHld, WORKING ON Threshold for thrname=response,reachable,available type=health item=
22:51:26 runThrHld, Found Configured Threshold for health, changing to "-4 hours"
22:51:26 getSummaryStats, Start type=health, index=, start=-4 hours, end=now
22:51:26 getFileName, filename of type=health is /data/database8/nodes/thor/health/reach.rrd
22:51:26 getDBName, returning database name=/data/database8/nodes/thor/health/reach.rrd for sect=health,
index=, item=
22:51:26 runThrHld, processing threshold response
22:51:26 getThresholdLevel, Start threshold=response, index= item=
22:51:26 getThresholdLevel, found threshold=response entry=default
22:51:26 getThresholdLevel, threshold=response, item=response, value=0.01
22:51:26 getThresholdLevel, result threshold=response, level=Normal, value=0.01, thrvalue=150, reset=150
22:51:26 thresholdProcess, Proactive Response Time, Normal, , value=0.01 reset=150
```

Threshold y escalados

Las alertas y los umbrales generan eventos que tienen un estado: Este evento esta abierto hasta que la alerta o el umbral vuelve a un valor dentro de los limites.

Una vez que tenemos un evento, podemos definir realizar un escalado, un mecanismo que nos permite establecer notificaciones de manera flexible basada en cuanto tiempo lleva el evento abierto.

Los escalados en NMIS se realizan en dos localizaciones para configurar los niveles y las acciones.

La primera es en System > System Configuration > NMIS Configuration > Escalations, que almacena la configuración en el fichero Config.nmis. Aquí es donde se guardan los niveles de escalado. Un nivel de escalado relaciona una cantidad de tiempo con un nombre. Asi, por ejemplo, el nivel de escalado por defecto escalate0 ocurre inmediatamente (A los 0 segundos). Escalate1 se lanza a los 300 segundos, y así sucesivamente. Los nombres y los tiempos se pueden configurar.

Las acciones se configuran en System > System Configuration > Escalations. Aqui es donde NMIS va a ver que debería de ocurrir cuando un evento se lanza y como se va a tratar en el tiempo.

6. Conceptos Adicionales

Thresholding VS Alertas

Una **alerta** es un evento personalizado generado por un test que evalúa un OID o una variable y produce un resultado verdadero o falso. Si el test devuelve verdadero, se genera un evento que recorrerá el sistema de escalado. Después, cuando el test devuelva falso, el evento sera eliminado.	Un **threshold** es un umbral que compara los datos de rendimiento con una tabla de valores predeterminados. Estos, resultan en alertas, eventos y notificaciones.

Diferencias

Alertas	**Thresholds**
Evalúa un único valor como verdadero o falso. Cuando el test se evalúa cómo falso > Se dispara la alerta. Son comparaciones simples, de un valor que se obtiene tras el collect. Un simple valor, sin valores históricos, ni estadísticas, ni matemáticas.	Evalúan una tabla de valores con un valor previamente almacenado (Una media de valores recolectados). Utilizan estadisticas, por defecto, los valores de los últimos 15 minutos (Pero se puede configurar).

Depuración de modelado de dispositivos

1. Despues de cambiar el modelo, efectuar un update y un collect para el dispositivo en el que estas trabajando:

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

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

1. Utilizando la opción model=true visualizara una salida muy útil para asistir en lo que se esta recolectando y si hay algún error:

```
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
```

Herramientas para trabajar con Modelado

Herramientas para trabajar con modelos: Tools for Working with NMIS Models

Trabajando con Multiples nodos y modelos

Tools for Working with NMIS Models

7- Modelado Avanzado (I)

Introducción al modelado avanzado

1. Hay ocasiones en las que vamos a modelar un dispositivo o en alguna situación donde colectar una variable simple SNMP no es suficiente.
2. Por ejemplo, cuando las propiedades SNMP se necesitan combinar para proporcionar una medida significativa.
3. A partir de la version de NMIS 8.4.8G, se soportan opciones de modelado avanzado, como puede ser el uso de variables avanzadas, o CVARs.

Opciones de modelado avanzado

Control

El control nos permite incluir una condición booleana para definir si una variable si va a collectar.

Calculate

Nos permite preprocesar un valor antes de guardarlo.

Replace

El resultado de la colección se puede reemplazar por un valor dado de una tabla de búsqueda predefinida. En este caso, el valor 1 o 0 será reemplazado por la cadena "sí" o "no".

ejm: `replace => {`

```
      '1' => 'si',

      '0' => 'no'

    }
```

No graphs

Si no tenemos la necesidad de mostrar un gráfico como parte de una tabla de la sección systemHealth, la opción 'graphtype' debe reemplazarse por: `'no_graphs' => '1'`

No guardar en RRD (nosave)

If data is collect using snmp by the model to be displayed but there is no need to save it to RRD, the option "nosave" should be used.

Si los datos se recopilan utilizando snmp por el modelo para ser mostrados solamente, pero no es necesario guardarlos en RRD, se debe utilizar la opción "nosave".

ejm: `'option' => 'nosave'`

Tenga en cuenta que la configuración de nosave desactiva las alertas para el objeto dado.

index_regex - Regex OID

Used in the "SystemHealth" section, 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,

Utilizado en la sección "SystemHealth", permite la indexación de elementos múltiples: normalmente las tablas SNMP son indexadas por el último componente numérico de un OID. Cuando NMIS realiza una actualización en una entidad indexada, itera a través de todos los valores conocidos para este componente de índice y los registra. Esta iteración no funciona si el índice consta de más de un número, como lo hace en ciertos equipos. En tales casos, puede establecer index_regex en un valor que capture los componentes OID que varían entre los elementos de la tabla. Por ejemplo,

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

asegura que los últimos tres números se usen para indexar.

Opciones adicionales

calculate

Formato: expresión

El resultado de la expresión evaluada reemplaza el valor recolectado originalmente.

control

Formato: expresión

La expresión de control se evaluará cuando se consulte la sección correspondiente.

event

Formato: cadena de caracteres

Nombre del evento.

format

Formato: cadena de caracteres

Formato Printf para reescribir el valor recopilado.

graphtype

Formato: Listado de valores separado por comas

List of graph names. For each graph type there must be a graph definition file Graph-<graphtype>.nmis

Lista de nombres de gráficos. Para cada tipo de gráfico debe haber un archivo de definición de gráfico Graph- <graphtype> .nmis

indexed

Formato: Nombre de variable SNMP/WMI (o "true")

Declara que esta subsección está indexada por el valor de la variable dada.

level

Formato: Nivel de severidad

Fatal, Critical, Major, Minor, Warning o Normal (en orden descendente)

logging

Formato: booleano

Si un evento debe registrarse o no.

oid

Formato: oid SNMP

Debe contener un nombre conocido o un OID numérico.

option

Formato: cadena de caracteres

Declara el tipo de datos para este origen de datos RRD, o evita que la variable se guarde si se utiliza "nosave". Si no está definido, el valor predeterminado es "gauge, U: U".

order

Formato: numero

Orden de procesamiento, siempre procede del número más bajo al más alto.

replace

Formato: tabla de búsqueda de valores conocidos y su reemplazo

El resultado de la colección se puede reemplazar por un valor dado de esta tabla de búsqueda. Si no se conoce el valor, se utiliza "unknown". Si no se puede encontrar ningún reemplazo, queda el valor original.

statstype

Formato: cadena de caracteres

Nombre para este tipo en la sección "stats".

stsname

Formato: cadena de caracteres

Nombre del parámetro (...:stsname=...) de las reglas rrd de la sección "stats".

sumname

Formato: cadena de caracteres

Nombre del parámetro en el archivo de resumen.

threshold

Formato: Listado de nombres separados por comas.

Nombres de los thresholds que deben declararse en la sección "stats".

title

Formato: cadena de caracteres

Se usa como etiqueta para mostrar esta variable.

value

Formato: expresión

El resultado de la expresión evaluada reemplaza el valor recolectado originalmente.

Modelado avanzado: variables personalizadas

Ocasionalmente, se encontrará con un dispositivo o una situación en la que la recopilación de una sola variable SNMP no es suficiente, por ejemplo, cuando se deben combinar dos o más propiedades SNMP para proporcionar una medición significativa.

NMIS versión 8.4.8G y posteriores admiten el modelado de tales escenarios utilizando variables personalizadas o CVAR. Con este mecanismo, puede capturar temporalmente hasta 11 propiedades SNMP separadas como CVAR y definir una expresión compleja arbitraria (en perl) que transforma estos CVAR en la única medida que desea recopilar y / o mostrar.

Dónde y cómo usar CVAR

CVARs se puede usar:

- en test expressions y valor en el subsistema de alerta y thresholds de NMIS,
- para calcular expresiones en el subsistema de modelado en general,
- y desde NMIS versión 8.6 en adelante, también en expresiones de control en cualquier lugar (en versiones anteriores solo se soportaba un CVAR único como control).

Un ejemplo

El MIB DS3 define una variedad de contadores de errores para circuitos DS3 como "dsx3CurrentLCVs" que se basan en un intervalo de observación de 15 minutos y se reinician automáticamente al final del intervalo. Como el intervalo de inicio y finalización es arbitrario y depende del dispositivo que se establezca, simplemente capturar los contadores de errores en sí mismos no es factible. Sin embargo, la MIB de DS3 también especifica la variable "dsx3TimeElapsed" que contiene los segundos transcurridos desde el inicio del intervalo de observación actual. Al dividir el contador de errores sin procesar por el número de segundos en el intervalo, se obtiene una tasa normalizada de errores por segundo que funciona bien para la recopilación y la visualización.

Aquí hay un extracto del archivo de modelo relevante:

```
'systemHealth' =>
{
    'sections' => 'ds3Errors',
    'sys' =>
    {
        'ds3Errors' =>
        {
            'indexed' => 'dsx3CurrentIndex',
            'index_oid' => '1.3.6.1.2.1.10.30.6.1.1',
            'headers' => 'ds3intf,ds3linestatus',
            'snmp' => {
                'ds3intf' => {
                    'oid' => '1.3.6.1.2.1.2.2.1.2', # ifDescr
                    'title' => 'DS3 Interface',
                },
                'ds3linestatus' => {
                    'oid' => '1.3.6.1.2.1.10.30.5.1.10', # dsx3LineStatus
                    'title' => "DS3 Line Status",
                    'calculate' => 'my @x; my %triggers=(1,"No Alarm",2,"Rx Remote Alarm",4,"Tx Remote Alarm",
8,"Rx AIS",16,"Tx AIS",32,"Rx LOF",64,"Rx LOS",128,"Loopback",256,"Test Pattern",512,"Unknown",1024,"Near end
unavailable signal",2048,"Carrier Equip OOS"); while (my ($num,$txt)=each(%triggers)) { push (@x,$txt) if (int
($r) & int($num)); }; return join(", ",@x); ',
                },
...
                'ds3LCV' => {
                    'oid' => '1.3.6.1.2.1.10.30.6.1.6', # dsx3CurrentLCVs
                    'title' => 'Line Coding Violations per second',
                    'calculate' => 'CVAR1=ds3Elapsed; return ($CVAR1? $r/$CVAR1 : 0);',
        },
    },  # sys

    'rrd' => {
        'ds3Errors' => {
            'indexed' => 'true',
            "graphtype" => "ds3Errors",
            "snmp" => {
                'ds3Elapsed' => {
                    'oid' => '1.3.6.1.2.1.10.30.5.1.3', # dsx3TimeElapsed
                    'title' => 'elapsed seconds in current measurement interval',
                    'option' => 'gauge,0:U',
                },
...
                "ds3LCV" => {
                    'oid' => '1.3.6.1.2.1.10.30.6.1.6',
                    'option' => 'gauge,0:U',
                    'title' => "Line Coding Violations per second",
                    'calculate' => 'CVAR1=ds3Elapsed; return ($CVAR1? $r/$CVAR1 : 0);',
                },
    },  # rrd
},  # systemhealth
```

En el ejemplo anterior, la expresión "calculate" se usan de dos maneras:

- para transformar la variable "bitfield " (DS3 Line Status) en una lista textual más detallada de estados de componentes,
- y para dividir el recuento de errores sin procesar dsx3CurrentLCVs por la longitud del intervalo dsx3TimeElapsed.

En ambos casos, la sintaxis es muy sencilla:

- La expresión debe ser una declaración perl válida y devolver exactamente un valor.
- NMIS interpreta los tokens $r y CVAR0 a CVAR9; todo lo demás es perl.
- Definir y usar variables locales con "my" es aceptable, pero no intente cambiar ninguna variable NMIS global.
- `"CVAR1=some_snmp_var;"` define que objeto SNMP debe estar contenido en la variable CVAR1.
- You can use functions that were defined elsewhere in NMIS in your `calculate` expression.
- Puede usar funciones que se definieron en otra parte de NMIS usando la expresión "calculate".
- `"return $r/$CVAR1;"` devuelve el valor contenido en `CVAR1`. La variable `"$r"` representa la variable SNMP a la cual la expresión "calculate" esta adherida.

Tenga en cuenta que:

- El reemplazo de $CVARn en la expresión se realiza sobre una base puramente textual, antes de entregar la expresión al intérprete perl para su evaluación:
    - Para varibles textuales, se debe proveer la expression utilizando comillas.

```
calculate => 'CVAR1=somestringthing; return 42 if ("$CVAR1" eq "online");'
```

- Las variables numéricas se pueden usar directamente sin comillas.
- el acceso $CVARn se refiere al valor bruto de la propiedad nombrada, es decir. se evaluaron los datos antes de reemplazar o calcular expresiones para la propiedad nombrada.

Cómo mantener los datos temporales de CVAR fuera de las bases de datos RRD

As outlined above all the objects that you want to access via `CVAR`s must be defined in the same section. If your test/calculate expression is within an `rrd` section, all the other objects will have to be within that `rrd` section, too, and thus they would be collected by NMIS and stored in RRD - quite wasteful if these other variables are just temporary and only there to for access using one `CVAR` expression.

Como se describió anteriormente, todos los objetos a los que desea acceder a través de CVAR deben definirse en la misma sección. Si el "test/calculate expression" está dentro de una sección rrd, todos los demás objetos también deberán estar dentro de esa sección rrd, por lo que NMIS los recopilará y almacenará en RRD, lo cual es un desperdicio si estas otras variables son solo temporales y solo para acceder usando una expresión CVAR.

En las versiones 8.6.0 y superiores, puede evitar esto agregando una opción "nosave":

```
'snmp' => {
  'hrNumUsers' => {
     'oid' => 'hrSystemNumUsers',
     'option' => 'nosave',
  },
```

In the example above, `hrNumUsers` would be retrieved with SNMP, and other variables could be defined in terms of e.g. `CVAR3=hrNumUsers`, but `hrNumUsers` would not be saved.

En el ejemplo anterior, hrNumUsers se obtiene con SNMP, y se podrían definir otras variables en términos de CVAR3 = hrNumUsers, pero hrNumUsers no se guardará.

8- Modelado Avanzado (II)

Operación de Plugins en NMIS 8

Los plugins de NMIS8 se ejecutarán para cualquier nodo que esté activo, pero lógicamente un plugin necesita poder saber en qué tipo de nodo está operando. Los plugins generalmente incluirán al principio del código una declaración para buscar un tipo de modelo específico y si el modelo es de un tipo interesante para él, realizará sus tareas, de lo contrario se saltará (no devolverá nada).

Para tener un nodo "API Only", el mejor método es establecer "node" a "true" y "ping, collect, collect_snmp y collect_wmi" a "false" y cambiar manualmente el modelo de "automático" al nombre del modelo que sera utilizado por el plugin.

Debido a que el dispositivo no proporciona datos, no sabemos cómo descubrir automáticamente cómo comunicarnos con el dispositivo, NMIS necesita un pequeño empujón. Puede ver un ejemplo de esto con los modelos y plugins CiscoMerakiCloud y CiscoViptelaCloud, que funcionan con Cisco Meraki y Cisco SDN WAN (Viptela) respectivamente.

Creación de Plugins Básicos

1. Las funciones collect_plugin y update_plugin pueden ser llamadas de forma secuencia por cada nodo, independientemente (Y posiblemente en procesos separados), para ejecutarse al final de la operación de collect y update, respectivamente.
2. Todos los plugins existentes serán cargados, y todas las funciones disponibles que se hayan definido en update_plugin y collect_plugin se aplicaran a cada nodo, independientemente de si el plugin ha terminado de forma satisfactoria o no.

La interfaz a invocar es la misma para ambas funciones:

    a. Sub update_plugin(node => nodename, sys => objeto sys en modo lectura/escritura, config => El objeto con la configuración de NMIS en modo solo lectura).
    b. Sub collect_plugin(node => nodename, sys => objeto sys en modo lectura/escritura, config => El objeto con la configuración de NMIS en modo solo lectura).

Valores de retorno:

- (0 o undef, el resto se ignora): Significa que no se han realizado cambios, o que el plugin declino finalizar la ejecución. Por ejemplo, si el nodo no encontro un match para el modelo.
- (1, el resto se ignora): Significa que se hicieron cambios al objeto sys, para que nmis pueda saberlo y guardar la información del nodo (nodeinfo) y los componentes de la vista.

- (2, lista de mensajes de error): Significa que el plugin a fallado su trabajo, y NMIS debería de registrar los mensajes de error. No se guardara la información de nodeinfo o de la vista del nodo.

Ejemplos

******* Use CMD8TEMP plugin as example (https://support.opmantek.com/browse/SUPPORT-6061)

Configuracion de poleo de NMIS 8

Versiones NMIS hasta NMIS 8.6.7G inclusive

A continuación se indica cómo se comportará NMIS cuando se le brinden varias opciones de configuración, esto se refiere específicamente a activo, ping y recopilación.

Descripción del poleo	active	ping	collect	plugins	services
Poleo de nodos regulares el nodo utilizara ICMP y obtendrá datos SNMP y WMI (si se configuraron las credenciales)	si	si	si	Se ejecutara	se colectara si está configurado
Nodos con "SNMP Only", se colecta SNMP pero no ICMP.	si	no	si	Se ejecutara	se colectara si está configurado
Ping Only Node, el nodo sera colectado usando solamente ICMP	si	si	no	Se ejecutara	se colectara si está configurado
Service Only Node, el nodo solo tendrá servicios recopilados si están configurados.	si	no	no	Se ejecutara	se colectara si está configurado
El nodo NO está activo y NMIS ignorará principalmente el nodo.	no	N/A	N/A	N/A	N/A

Versiones NMIS posteriores e incluidas NMIS 8.6.8G

The following indicates how NMIS will behave when provided with various configuration options, this is specifically concerned with active, ping and collect.

A continuación se indica cómo se comportará NMIS cuando se le brinden varias opciones de configuración, esto se refiere específicamente a activo, ping y recopilación.

Polling Description	active	ping	collect	collect_snmp	collect_wmi	services
Regular node polling, the node will be ICMP polled and will have SNMP and WMI (if credentials configured) data collected	true	true	true	true	true	will be polled if configured
SNMP or WMI Only Node, the not will have SNMP collected but ICMP polling will not be performed.	true	false	true	true	true	will be polled if configured
Ping Only Node, the node will only be polled using ICMP.	true	true	false	false	false	will be polled if configured
Service Only Node, the node will only have services collected if they are configured.	true	false	false	false	false	will be polled if configured
API Only Node, the node will use plugins to collect data, no other polling will be done, except services if configured.	true	false	false	false	false	will be polled if configured
Node is NOT active and NMIS will mostly ignore the node.	false	N/A	N/A	N/A	N/A	N/A

Model Policy

NMIS 8.6 introduced a new mechanism for adjusting a model's behaviour for particular nodes: the Model Policy system. In version 8.6.0G it allows you to specify flexible rules for adding or removing systemHealth model sections for specific nodes (or groups of nodes).

The Model Policy Document

The installer will install a default model policy document in conf/Model-Policy.nmis. The original/default file will also remain available in the install directory, and contains helpful comments.

The structure of the policy document is quite simple but fairly flexible:

- The policy consists of any number of rules.
- The rules are evaluated in order of their numeric key; fractional numbers are supported to simplify insertion.
- Only the first matching rule is applied.
- Each rule must express what changes to systemHealth should be made.
- The changes are expressed as a list of systemHealth section name plus the desired activation state (true or false).

- systemHealth sections not named are not modified and therefor default to True or on (so check the default section for what would otherwise be disabled).
- Each rule may include any number of filter expressions, which determine whether the rule should be applied to a particular node.
- All given filter expressions must match simultaneously for the rule to be considered a match.
- A filter expression defines a node property or configuration setting to be compared against an explicit list of acceptable values, or a regular expression.
- Node properties are given as node.<propname>, and for configuration settings you'd use conf.<configsetting>.
- All configuration settings are available, using the prefix conf. and the same names as seen in conf/Config.nmis.
- The available node properties are: the static ones from the node configuration, plus the more dynamic ones from the system section in the node's "node info" file (var/<nodename>-node.json).
- A Model Policy document may also include an extra section named _display, which controls in what order the default policy's entries should be shown in the Configuration GUI.
- See the default policy for an example.

It should be noted that as Only the first matching rule is applied and therefor the default rule is not subsequently applied you should include all the relevant "false" sections from the default rules into your rule. For example if you wanted to turn on just one mpls system health section you would set that as true in your rule and you would also include all the other "false" lines which are relevant to your model in the rule.

Example Policy

Here is a partial example policy:

```
%hash = (
         # rule numbers may be fractional numbers (for easy insertion)
         # first matching rule terminates the policy application
         10 => {
             # filter keys: node.xyz or config.abc; node.nodeModel is the (possibly dynamic) current model
             # filter values: string, list of strings,
             # or regexp (=string with //, optional case-insensitive //i)
             IF => { 'node.name' => ['node1','node2'],
                     'node.location' => '/def.*/',
                     'config.auth_ldap_server' => '/192\./', },
             # sections to adjust, only systemHealth supported so far
             systemHealth => {
                 'fanStatus' => 'true',    # add if not present
                 'tempStatus' => 'false', # remove if present
             },
         },
         20 => {
             IF => { 'node.name' => 'embedded' },
             systemHealth => {
                 diskIOTable => 'false' # this node runs off r/o flash disk
             },
         },
         999 => {    # the fallback/defaults, without filter
             systemHealth => {
                 cdp => 'true',
                 lldp => 'true',
                 bgpPeer => 'true',
                 ospfNbr => 'true', }
         } );
```

The first rule applies to at most two particular nodes (because of the given list of node.name values), and only if their location property starts with "def" and only if the NMIS configuration is set up for an LDAP server in the 192.0.0.0/8 network. For all systems that match these restrictions the fanStatus and tempStatus model sections are enabled.

The second rule disables the diskIOTable model section for a specific system that doesn't have real 'disks', just a readonly flash drive.

The last rule does not have any filtering IF clauses, therefore applies to all nodes and thus it serves to set the "default" policy. As mentioned above only the first matching rule is applied, hence the default rule will only apply to nodes where rules 10 and 20 have not matched.