Thomas Fahle

SNMP::Info - SNMP mal ganz einfach

SNMP::Info

• ist eine objektorientierte Schnittstelle für Perl5, um Informationen per SNMP abzufragen oder Einstellungen eines Devices per SNMP zu setzen
• unterstützt sehr viele Netzwerkkomponenten
• bietet eine device orientierte Sicht mit einer gemeinsamen Schnittstelle für alle unterstützten Devices
• verhüllt die Details des nicht ganz so simplen Protokolls SNMP
• unterstützt SNMP v1, v2c und v3
• bietet SNMP Einsteigern einen guten Einstieg
• lässt sich einfach erweitern
• ist gut dokumentiert

SNMP

Simple Network Management Protocol (SNMP) ist ein Netzwerkprotokoll, das von Netzwerk-Management-Systemen (NMS) genutzt wird, um Netzwerkkomponenten (Devices) wie z.B. Router, Server, Switches von einer zentralen Station aus überwachen und steuern zu können.

Netdisco

SNMP::Info ist die Perl-Bibliothek hinter *Netdisco*, einem NMS für mittlere bis große Installationen. Eine Spezialität von Netdisco ist, wie der Name schon andeutet, Network Discovery. Netdisco kann via Layer 2 Discovery eine Topologie des Netzes erstellen und ist z.B. bei der Suche nach `rogue access point` recht nützlich.

SNMP::Info wird aktiv weiterentwickelt und kann ohne Netdisco in eigenen Applikationen genutzt werden.

Unterstützte Devices

SNMP::Info unterstützt zahlreiche Netzwerkkomponenten. Ein detaillierte Übersicht findet sich in der Datei *DeviceMatrix.txt*, die mit SNMP::Info ausgeliefert wird. Die Übersicht steht auch online auf der Website des Projekts bereit.

Installationsvoraussetzungen

SNMP::Info benötigt die `Net-SNMP` Bibliothek mit Perlunterstützung (SNMP::Session) und funktionierende MIB-Module für die zu unterstützenden Netzwerkkomponenten.

Die nicht ganz triviale Installation der benötigten MIB-Module, lässt sich durch die Installation der *netdisco-mibs*, einer gepflegten MIB-Sammlung, erheblich vereinfachen.

Schritt für Schritt

Konstruktor new()

Die Methode `new()` erstellt ein neues Objekt der Klasse SNMP::Info.

Pflichtparameter ist die IP-Adresse oder der DNS-Name des Devices. Wenn kein SNMP Community String angeben wird, wird `public` verwendet.

Die SNMP-Version kann über den Parameter `Version` konfiguriert werden (v2c ist Default).

Der Parameter `AutoSpecify` erlaubt SNMP::Info automatisch die passende Subklasse für das Device auszuwählen. Dazu gleich mehr.

```
#!/usr/bin/perl
use warnings;
use strict;

use SNMP::Info;

my $desthost  = '10.0.0.1';
my $community = 'public';

my $info = new SNMP::Info(

  # Auto Discover more specific Device Class
  AutoSpecify => 1,
  Debug       => 0,

  # The rest is passed to SNMP::Session
  DestHost    => $desthost,
  Community   => $community,
  Version     => 1,
) or die "Can't connect to $desthost!\n";

print "Connected to $desthost.\n";
```

Unter der Haube von SNMP::Info werkelt SNMP::Session, d.h. alle von SNMP::Session unterstützten Parameter können im Konstruktor verwendet werden.

Automatische Auswahl der richtigen Subklasse

SNMP::Info fragt dazu per SNMP auf dem Device `sysServices.0`, `sysDescr.0` und `sysObjectID.0` ab und wählt aus diesen Informationen die passende gerätespezifische Subklasse.

Die Methode `device_type()` liefert den Namen der ausgewählten Subklasse für das Device zurück.

```
print "Detected Device Type: ",
 $info->device_type(), "\n";
```

Ausgabe bei einem Cisco Catalyst 2960S:

```
Detected Device Type:
  SNMP::Info::Layer3::C6500
```

Ausgabe bei einem HP ProCurve J9021A Switch 2810-24G:

```
Detected Device Type:
   SNMP::Info::Layer2::HP
```

Innerhalb der gerätespezifischen Subklassen werden allgemeine Methoden passend für die Netzwerkkomponente überschrieben. SNMP::Info bietet so eine gemeinsame Schnittstelle für alle unterstützten Devices an.

Einfache SNMP Abfragen als Einstieg

Als Einstieg ein paar einfache SNMP-Queries, wie sie in vielen SNMP-Tutorien für Einsteiger verwendet werden.

Alle Methoden geben einen skalaren Wert zurück.

```
print "sysUpTime: ",  $info->uptime(), "\n";
print "sysContact: ", $info->contact(),"\n";
print "sysName: ",    $info->name(),   "\n";
print "sysLocation: ",
        $info->location(), "\n";
print "MAC Root Port: ", $info->mac(), "\n";

print "OS: ",     $info->os(),     "\n";
print "Vendor: ", $info->vendor(), "\n";
print "Model: ",  $info->model(),  "\n";
print "Software Version: ",
        $info->os_ver(), "\n";
```

Ausgabe bei einem Cisco Catalyst 2960S:

```
sysUpTime: ddddddddd
sysContact: XXXXXXX
sysName: XXXXXXXXXX
sysLocation: XXXXXX
MAC Root Port: xx:xx:xx:xx:xx:xx
OS: ios
Vendor: cisco
Model: 29xxStack
Software Version: 12.2(55)SE3
```

Ausgabe bei einem HP ProCurve J9021A Switch 2810-24G:

```
sysUpTime: ddddddddd
sysContact: XXXXXXX
sysName: XXXXXXXXXX
sysLocation: XXXXXX
MAC Root Port: xx:xx:xx:xx:xx:xx
OS: hp
Vendor: hp
Model: 2810-24G
Software Version: N.11.25
```

Hardwarestatus

Dank der gemeinsamen Schnittstelle für alle Devices, läÃŸt sich der Status eingebauter Hardware, wie Netzteile, Ventilatoren oder RAM, sehr einfach abfragen. Alle aufgezeigten Methoden geben einen skalaren Wert zurück.

```
# Fan and Power supplies
print "Fan status: ",
  $info->fan(), "\n";
print "Power supply 1 status: ",
  $info->ps1_status(), "\n";
print "Power supply 2 status: ",
  $info->ps2_status(), "\n";

# Returns bytes of total memory
print "Speicher gesamt: ",
  $info->mem_total(), " Bytes\n";

# Returns bytes of free memory
print "Freier Speicher: ",
  $info->mem_free(), " Bytes\n";

# Returns bytes of used memory
print "Speicher genutzt: ",
  $info->mem_used(), " Bytes\n";
```

Ausgabe bei einem Cisco Catalyst 2960S:

```
Fan status: ok
Power supply 1 status: ok
Power supply 2 status: other
Speicher gesamt: 89697640 Bytes
Freier Speicher: 50696124 Bytes
Speicher genutzt: 39001516 Bytes
```

Ausgabe bei einem HP ProCurve J9021A Switch 2810-24G:

```
Fan status: good
Power supply 1 status: good
Power supply 2 status: notPresent
Speicher gesamt: 24034944 Bytes
Freier Speicher: 16851344 Bytes
Speicher genutzt: 7183600 Bytes
```

SNMP::Info wählt hier automatisch für jedes Device die passenden SNMP-Abfragen.

So wird beispielsweise zur Ermittlung des freien Speichers bei Cisco `ciscoMemoryPoolFree` abgefragt, bei HP hingegen `hpGlobalMemFreeBytes`.

Tabellarische Daten

Oft liegen Daten, die per SNMP ermittelt werden, als Tabellen vor.

Zur Abbildung von Tabellen nutzt SNMP::Info *parallele Hashes* mit identischen Schlüsseln als Datenstruktur. Dazu mehr in den nächsten Abschnitten.

Interface Daten

Um Interfacedaten, wie Beschreibung, Typ oder Geschwindigkeit, eines Devices zu ermitteln, benötigt man zunächst eine Liste der Interfaces, die man über die Methode `interfaces()` erhält.

```
my $interfaces = $info->interfaces();
```

`$interfaces` ist eine Referenz auf einen Hash, der als Schlüssel Interface-IDs (iid) enthält.

Gekürzte Data::Dumper Ausgabe

```
$VAR1 = {
    '10127' => '10127',
    '10131' => '10131',
 };
```

Im nächsten Schritt werden die gewünschten Interfacedaten geholt, z.B. die Geschwindigkeit via `i_speed()`.

```
my $interfaces_speed = $info->i_speed();
```

`$interfaces_speed` ist ebenfalls eine Referenz auf einen Hash, dessen Schlüssel, wie oben, Interface-IDs enthalten; die jeweiligen Geschwindigkeiten werden als Werte gespeichert.

Gekürzte Data::Dumper Ausgabe

```
$VAR1 = {
    '10127' => '1.0 Gbps',
    '10131' => '100 Mbps',
 };
```

Dieser Vorgang wird für weitere Interfacedaten beliebig oft wiederholt.

So liefert beispielsweise `i_mac()` die MAC-Adresse (sofern vorhanden) und `i_description()` eine Beschreibung des Interfaces.

Zwecks Ausgabe der Werte, läuft man in einer Schleife über die Schlüssel (iid) der Intefaceliste und holt mit eben diesem Schlüssel in den parallelen Hashes die Werte.

```
my $interfaces = $info->interfaces();
my $interfaces_description =
  $info->i_description();
my $interfaces_mac   = $info->i_mac();
my $interfaces_speed = $info->i_speed();

foreach my $iid ( sort { $a <=> $b }
  keys %$interfaces )
{
  my $description =
    $interfaces_description->{$iid}
    || 'No interface description.';
  my $mac = $interfaces_mac->{$iid}
    || 'Keine MAC-Adresse gefunden.';
  my $speed = $interfaces_speed->{$iid}
    || 'Unbekannte Geschwindigkeit.';

  printf( "%20s %10s\n",
    $description, $speed );
}
```

Beispiel Ausgabe:

```
Vlan1                    1.0 Gbps
GigabitEthernet1/0/47     10 Mbps
GigabitEthernet1/0/48    100 Mbps
GigabitEthernet1/0/49    1.0 Gbps
```

SNMP::Info zeigt die Geschwindigkeiten in einem menschenlesbarem Format an und wählt im Hintergrund die passende SNMP-Abfrage für das jeweilige Gerät, z.B. `ifSpeed` oder `ifHighSpeed`.

Interface Statistiken

Statistische Interfacedaten, wie empfangene und gesendete Daten oder Übertragungsfehler, lassen sich mit den Methoden `i_octet_in()`, `i_octets_out()`, `i_errors_in()` und `i_errors_out()` einfach ermitteln.

```
my $interfaces = $info->interfaces();
my $interface_descriptions =
  $info->i_description();

my $i_octet_in   = $info->i_octet_in();
my $i_octets_out = $info->i_octets_out();
my $i_errors_in  = $info->i_errors_in();
my $i_errors_out = $info->i_errors_out();

foreach my $iid ( sort { $a <=> $b }
  keys %$interfaces )
{
  my $description =
    $interface_descriptions->{$iid}
    || 'No description';
  my $octets_in = $i_octet_in->{$iid}
    || 0;
  my $octets_out = $i_octets_out->{$iid}
    || 0;
  my $errors_in = $i_errors_in->{$iid}
    || 0;
  my $errors_out = $i_errors_out->{$iid}
    || 0;

  printf(
    "%20s: %10s %10s %10s %10s\n",
    $description, $octets_in, $octets_out,
    $errors_in, $errors_out
  );
}
```

Beispiel Ausgabe:

```
GigabitEthernet1/0/48: 2017710952  0 0  0
GigabitEthernet1/0/49: 3239869357  0 0  0
```

Network Discovery (CDP und LLDP)

Das nachfolgende Beispiel, das ich der SNMP::Info Dokumention entnommen habe, ermittelt die IP-Adressen und physischen Portnamen auf den direkt angeschlossen Devices (Neighborhood) mittels Cisco Discovery Protocol (CDP) oder Link Layer Discovery Protocol (LLDP).

SNMP::Info verhüllt, wie gewohnt, die internen Details hinter einer allgemeinen Schnittstelle und schaltet intern je nach Device auf das entsprechende Protokoll um.

Als Mapping Tabelle zwischen den Interface-IDs der verschiedenen Devices dient `c_if()`. Der zurückgegebene Hash enthält die remote Interface-IDs als Keys und die lokalen Interface-IDs als Werte.

```
my $interfaces = $info->interfaces();

# Get CDP Neighbor info

# Returns the mapping
# to the SNMP Interface Table
my $c_if   = $info->c_if();

# returns remote IPV4 address
my $c_ip   = $info->c_ip();

# Returns remote port ID
my $c_port = $info->c_port();

# Print out data per port
foreach my $iid ( keys %$interfaces ) {

  # Print out physical port name,
  # not snmp iid
  my $port = $interfaces->{$iid};

  # The CDP Table has table entries
  # different than the interface
  # tables.
  # So we use c_if to get the map
  # from cdp table to interface
  # table.

  my %c_map = reverse %$c_if;
  my $c_key = $c_map{$iid};
  unless ( defined $c_key ) {
    next;
  }
  my $neighbor_ip   = $c_ip->{$c_key};
  my $neighbor_port = $c_port->{$c_key};

  if ( defined $neighbor_ip ) {
    print "Local port '$port'
          connected to '$neighbor_ip'
          remote port '$neighbor_port'\n";
  }
}
```

Beispiel Ausgabe:

```
Local port 'GigabitEthernet1/0/48'
       connected to '172.xx.xxx.x'
       remote port 'GigabitEthernet1/0/48'
Local port 'GigabitEthernet1/0/49'
       connected to '172.xx.xx.xx'
       remote port '9/3'
```

SNMP::Info kann noch viel mehr

SNMP::Info kann deutlich mehr als hier gezeigt. Eine vollständige Übersicht würde den Rahmen dieses Artikels deutlich sprengen.

Ich empfehle einen Blick ins umfangreiche, gut gepflegte Handbuch für mehr Informationen.