SNMP Démystifié

David Delavennat

CNRS / Centre de Génétique Moléculaire

ANGD *Mathrice*, Novembre 2009

Rev : 106

Plan

Plan

IANA / ICANN

Internet **A**ssigned **N**umbers **A**uthority : 1990

- organisation internationale à but non lucratif
- alloue l'espace des adresses de protocole Internet (IP)
- attribue les identificateurs de protocole (UDP/TCP)
- gére le système de nom de domaine de premier niveau pour les codes génériques (gTLD) et les codes nationaux (ccTLD)
- assure les fonctions de gestion du système de serveurs racines

Internet **C**orporation for **A**ssigned **N**ames and **N**umbers : 1998

- organisation de droit privé à but non lucratfif
- alloue l'espace des adresses de protocole Internet (IP)
- attribue les identificateurs de protocole (UDP/TCP)
- gére le système de nom de domaine de premier niveau pour les codes génériques (gTLD) et les codes nationaux (ccTLD)
- assure les fonctions de gestion du système de serveurs racines
- L'ICANN assume à présent les fonctions de l'IANA

Object IDentifier

- identifiants universels de ressources
- représentés sous la forme d'une suite d'entiers
- organisés sous forme hiérarchique
- assurent l'interopérabilité entre différents logiciels
- seul l'organisme 1.2.3 peut dire quelle est la signification de l'OID 1.2.3.4
- ont été définis dans une recommandation de l'International Telecommunication Union.
- L'IETF a proposé de représenter la suite d'entiers constituant les OID séparés par des points.
- Il est possible d'obtenir un OID, et par conséquence toute une branche, auprès de l'IANA.

OID	Description
0	Branche ITU
1	Branche ISO
2	Branche commune entre l'ITU et l'ISO
2.5	Service X500
2.5.4	Définition des types d'attributs
2.5.6	Définition des classes d'objets
1.3.6.1	Internet OID
1.3.6.1.4	IANA-assigned company OIDs (private MIBs)
1.3.6.1.4.1.4203	OpenLDAP

Abstract Syntax Notation .1

Le langage ASN.1 spécifie un format d'encodage d'informations au sein d'un flux de données ainsi que des types de données.

- celui-ci décrit des syntaxes abstraites (totalement indépendante des syntaxes de transferts)
- le typage des données est explicite (le type transmis est transfèré sur le réseau)
- les définitions des types sont dans des modules (réutilisables ensuite avec importation)
- un type est défini par une collection de valeurs qui peut être distinguée effectivement d'autres valeurs
- à partir de types existants, on peut définir des sous-types, ce qui permet ainsi la définition de types de façon récursive

Les types sont divisés en 4 groupes :

- les types basiques : valeurs caractérisées facilement , indépendamment d'autres types
- les types structurés : types complexes définis à partir d'autre types
- les modules : constituent des paquetages de plusieurs définitions de types
- les tags : permettent de rajouter des informations pour certains types afin de faciliter le codage et décodage

La définition d'un type est constituée de 3 éléments : [Nom type] ::= [Description du type]

- un nom qui représentera le nouveau type
- le symbole '::='
- la description du type

Attribution d'une valeur

- Exemple : [Nom valeur] [Nom type] ::= [Description valeur]

Abstract Syntax Notation .1

Types de base	Domaine de valeur	Exemple
INTEGER	entier pouvant être positif, négatif ou nul. Rappel : Il n'y a pas de limite pour les valeurs possibles car on définit une syntaxe abstraite. On peut associer à chaque valeur un identifiant qui lorsqu'il sera employé correspondra à la valeur entière définie par l'identifiant. -2^{31} à $2^{31}-1$	Hexadecimal ::= INTEGER { 1(1), 2(2), 3(3), 4(4), 5(5), 6(6), 7(7), 8(8), 9(9), A(10), B(11), C(12), D(13), E(14), F(15) }
ENUMERATED	permet d'énumérer un ensemble de valeurs en associant un entier à un identifiant	Ex : Orientation : := ENUMERATED nord(0), sud(1), est(2), ouest(3)
BIT STRING	définition d'une chaine de bits de taille >= 0	Mot32 ::= BIT STRING (SIZE(32)) ;
OCTET STRING	séquence ordonnée de 0 à 65,535 octets	Extension : := OCTET STRING(SIZE(3))
NULL	type ne prenant qu'une seule valeur qui est null. Ce type est utilisé par exemple lorsqu'on l'on doit forcément spécifier un type sans forcément transmettre d'information	
BOOLEAN	TRUE et FALSE	Actif : := BOOLEAN
OID		

Abstract Syntax Notation .1

Types complexes	Domaine de valeur	Exemple
SEQUENCE	collection ordonnée d'éléments distincts, les éléments pouvant être de types différents. Les éléments peuvent être de type OPTIONAL indiquant que la valeur n'est pas nécessairement présente dans la séquence. Le mot clef DEFAULT indique que l'on attribue une valeur par défaut au composant lorsqu'il n'est pas spécifié	
SEQUENCE OF	collection ordonnée d'éléments de même type	
SET	collection non ordonnée d'éléments non distincts qui peuvent être de types différents	
SET OF	collection non ordonnée d'éléments non distincts du même type	
CHOICE	collection de types, chacun étant distinct des autres	
ANY	correspond en fait à tous les types possibles	

Basic Encoding Rules

Utilisé pour transmettre des données entre des systèmes dont l'encodage natif diffère

- Type
- Longueur (en nombre d'octets)
- Donnée

aussi appellé encodage **T**ype-**L**enght-**V**alue

Bit No.	8	7	6	5	4	3	2	1	Implication
	0	0	_	.	.	.	.	.	Universal
	0	1	_	.	.	.	.	.	Application
	1	0	_	.	.	.	.	.	Context
	1	1	_	.	.	.	.	.	Private
	_	_	0	.	.	.	.	.	Primitive Data-type
	_	_	1	.	.	.	.	.	Constructed

Basic Encoding Rules : Data Type

Types	Data-Type	Hex ID	Bin ID
INTEGER	Primitif	0x02	0000 0010
BIT STRING	Primitif	0x03	0000 0011
OCTET STRING	Primitif	0x04	0000 0100
NULL	Primitif	0x05	0000 0101
OBJECT IDENTIFIER	Primitif	0x06	0000 0110
SEQUENCE	Constructed	0x30	0011 0000
IPADDRESS	SNMP Application	0x40	0100 0000
COUNTER, COUNTER32	SNMP Application	0x41	0100 0001
GAUGE, GAUGE32	SNMP Application	0x42	0100 0010
TIMETICKS	SNMP Application	0x43	0100 0011
OPAQUE	SNMP Application	0x44	0100 0100
NSAPADDRESS	SNMP Application	0x45	0100 0101
COUNTER64	SNMP Application	0x46	0100 0110
UINTEGER32	SNMP Application	0x47	0100 0111
getRequest	SNMP Context Constructed	0xA0	1010 0000
getNextRequest	SNMP Context Constructed	0xA1	1010 0001
setRequest	SNMP Context Constructed	0xA2	1010 0010
trap	SNMP Context Constructed	0xA4	1010 0100
getBulkRequest	SNMP Context Constructed	0xA5	1010 0101
informRequest	SNMP Context Constructed	0xA6	1010 0110

Plan

Management Information Base / Structure of Management Information

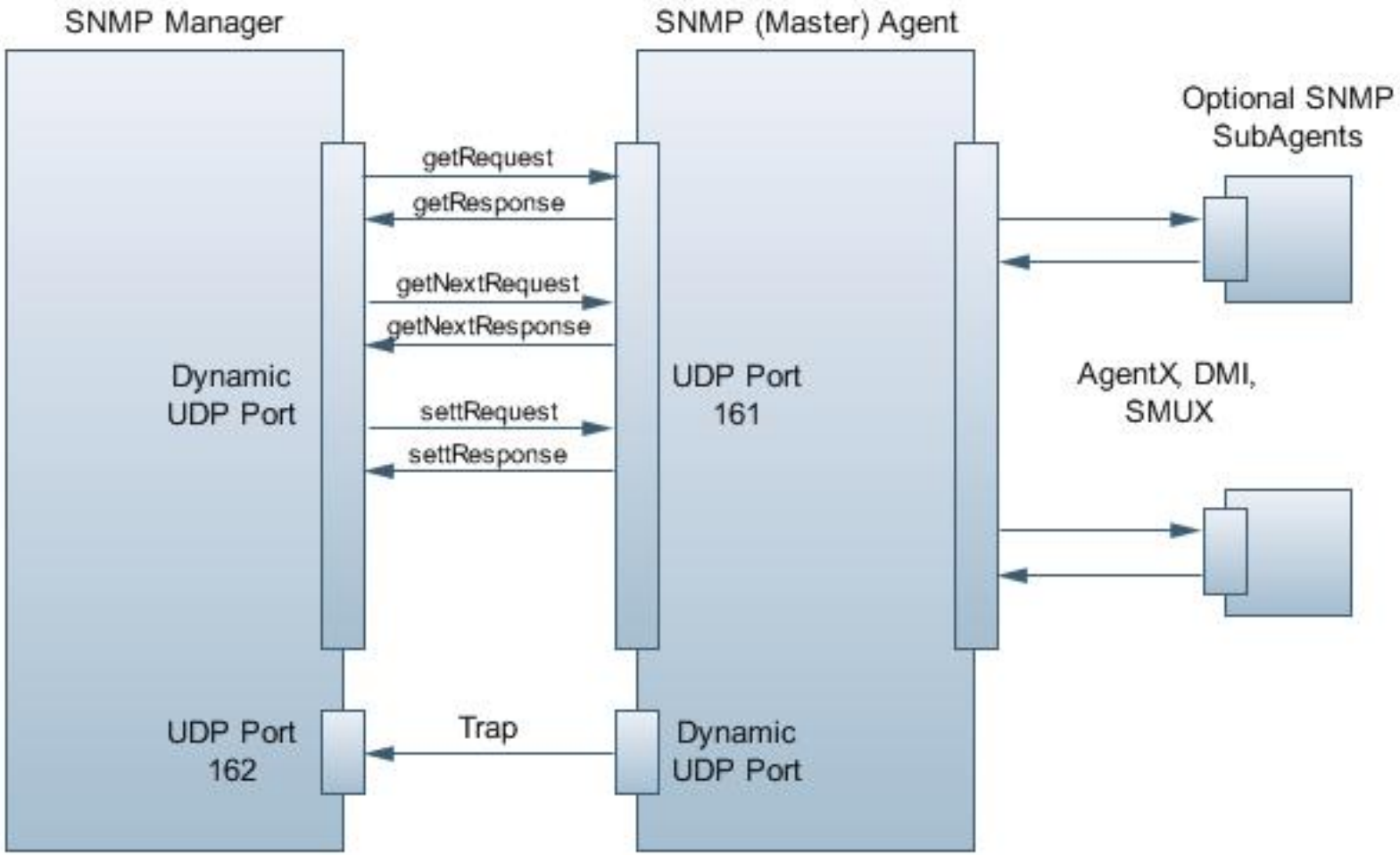
Dans quel but ?

- contient un identifiant textuel pour chaque OID
- un gestionnaire SNMP utilise ces identifiants pour convertir les OID en texte humainement compréhensible
- sans MIB, un message ne contient que des chiffres dénués de sens

Comment ?

- un gestionnaire SNMP importe une MIB SMI (spécification au format ASN.1) puis la ≪ compile ≫
- une compilation convertit la MIB depuis un format ASCII vers un format utilisable nativement par le gestionnaire SNMP

Simple Network Management Protocol



SNMPv1 / SNMPSec

SNMPv1

RFC1155	Mai 1990	Structure and identification of management information for TCP/IP-based internets
RFC1156	Mai 1990	Management Information Base for network management of TCP/IP-based internets
RFC1157	Mai 1990	Simple Network Management Protocol (SNMP)
RFC1213	Mars 1991	Management Information Base for Network Management of TCP/IP-based internets : MIB-II

SNMPSec

RFC1351	Juillet 1992	SNMP Administrative Model
RFC1352	Juillet 1992	SNMP Security Protocols
RFC1353	Juillet 1992	Definitions of Managed Objects for Administration of SNMP Parties

SNMPv1


MESSAGE SNMPv1
Version
02 01 00
Community
04 06
Packet DATA Unit
(getRequest,SetRequest, getNextRequest, getResponse)
A0 30
public
70 75 62 6c 69 63
getRequest
Request ID
02 01 01
Error
02 01 00
Error Index
02 01 00
Varbind List { type: sequence, lenght: }
30 02
Varbind List
Varbind
Object Identifier
Value
02 02
02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02
05 00

SNMPv2

SNMPv2p : party-based

- mise à jour des opérations du protocole
- nouvelles opérations et nouveaux types de données
- sécurité basée sur les groupes de SNMPsec

RFC1441	Avril 1993	Introduction to version 2 of the Internet-standard Network Management Framework
RFC1442	Avril 1993	Structure of Management Information for SNMPv2
RFC1443	Avril 1993	Textual Conventions for SNMPv2
RFC1444	Avril 1993	Conformance Statements for SNMPv2
RFC1445	Avril 1993	Administrative Model for SNMPv2
RFC1446	Avril 1993	Security Protocols for SNMPv2
RFC1447	Avril 1993	Party MIB for SNMPv2
RFC1448	Avril 1993	Protocol Operations for SNMPv2
RFC1449	Avril 1993	Transport Mappings for SNMPv2
RFC1450	Avril 1993	Management Information Base for SNMPv2
RFC1451	Avril 1993	Manager-to-Manager Management Information Base
RFC1452	Avril 1993	Coexistence between version 1 and version 2 of the Internet-standard Network Management Framework

SNMPv2

SNMPv2c : community-based

- aussi appelé " community stringbased SNMPv2 "
- amélioration des opérations de protocole et des types d'opérations de SNMPv2p
- utilise la sécurité par chaine de caractères "community" de SNMPv1

RFC1901	Janvier 1996	Introduction to Community-based SNMPv2
RFC1902	Janvier 1996	Structure of Management Information for SNMPv2
RFC1903	Janvier 1996	Textual Conventions for SNMPv2
RFC1904	Janvier 1996	Conformance Statements for SNMPv2
RFC1905	Janvier 1996	Protocol Operations for SNMPv2
RFC1906	Janvier 1996	Transport Mappings for SNMPv2
RFC1907	Janvier 1996	Management Information Base for SNMPv2
RFC1908	Janvier 1996	Coexistence between SNMPv1 and SNMPv2

SNMPv2

SNMPv2u : user-based

- utilise les opérations, les types de données de SNMPv2c
- sécurité basée sur les usagers

RFC1009	Février 1996	An Administrative Infrastructure for SNMPv2
RFC1010	Février 1996	User-based Security Model for SNMPv2

SNMPv2*

- combine les meilleures parties de SNMPv2p et SNMPv2u.
- documents décrivant cette version jamais publiés

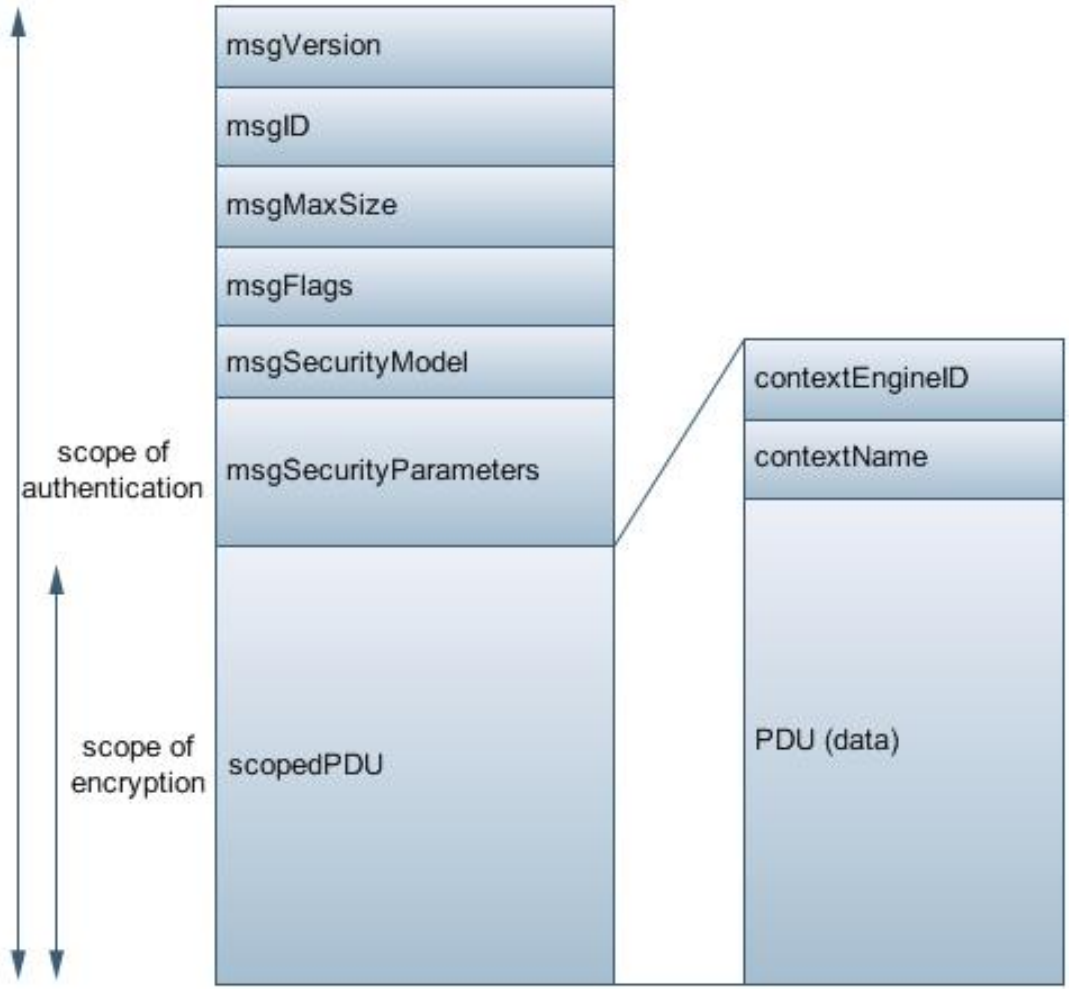
SNMPv3

- sécurité basée sur les usagers (SNMPv2u et SNMPv2*)
- types et opérations de SNMPv2p

SNMPv3

RFC2576	March 2000	Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework
RFC2578	Avril 1999	Structure of Management Information Version 2 (SMIv2)
RFC2579	April 1999	Textual Conventions for SMIv2
RFC2580	April 1999	Conformance Statements for SMIv2
RFC3410	December 2002	Introduction and Applicability Statements for Internet-Standard Management Framework
RFC3411	December 2002	An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks
RFC3412	December 2002	Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)
RFC3413	December 2002	Simple Network Management Protocol (SNMP) Applications
RFC3414	December 2002	User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)
RFC3415	December 2002	View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)
RFC3416	December 2002	Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)
RFC3417	December 2002	Transport Mappings for the Simple Network Management Protocol (SNMP)
RFC3418	December 2002	Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)

SNMPv3


msgVersion
msgID
msgMaxSize
msgFlags
msgSecurityModel
msgSecurityParameters
scopedPDU
contextEngineID
contextName
PDU (data)
scope of authentication
scope of encryption

Security Model : CSM

Community-based Security Model (SNMPv1 / SNMPv2c)

- nom de communauté, en clair
- public
- private

```
1  #
2  # NET-SNMP
3  #
4  rocommunity   COMMUNITY [SOURCE [OID]]
5  rwcommunity   COMMUNITY [SOURCE [OID]]
6  rocommunity6  COMMUNITY [SOURCE [OID]]
7  rwcommunity6  COMMUNITY [SOURCE [OID]]
8
9  rocommunity   COMMUNITY SOURCE -V VIEW
10 rwcommunity   COMMUNITY SOURCE -V VIEW
11 rocommunity6  COMMUNITY SOURCE -V VIEW
12 rwcommunity6  COMMUNITY SOURCE -V VIEW
```

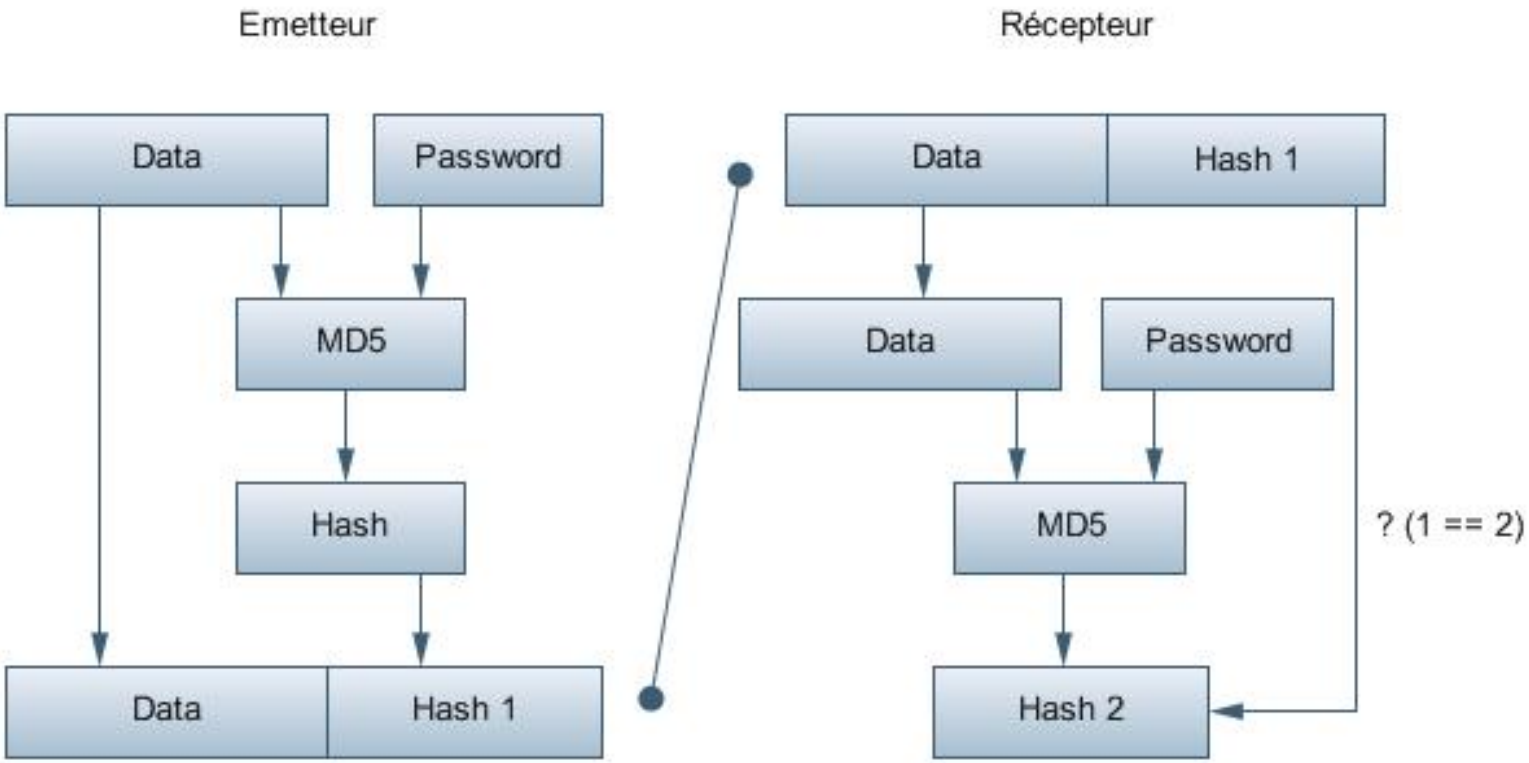
Security Model : USM

User-based Security Model (SNMPv3)

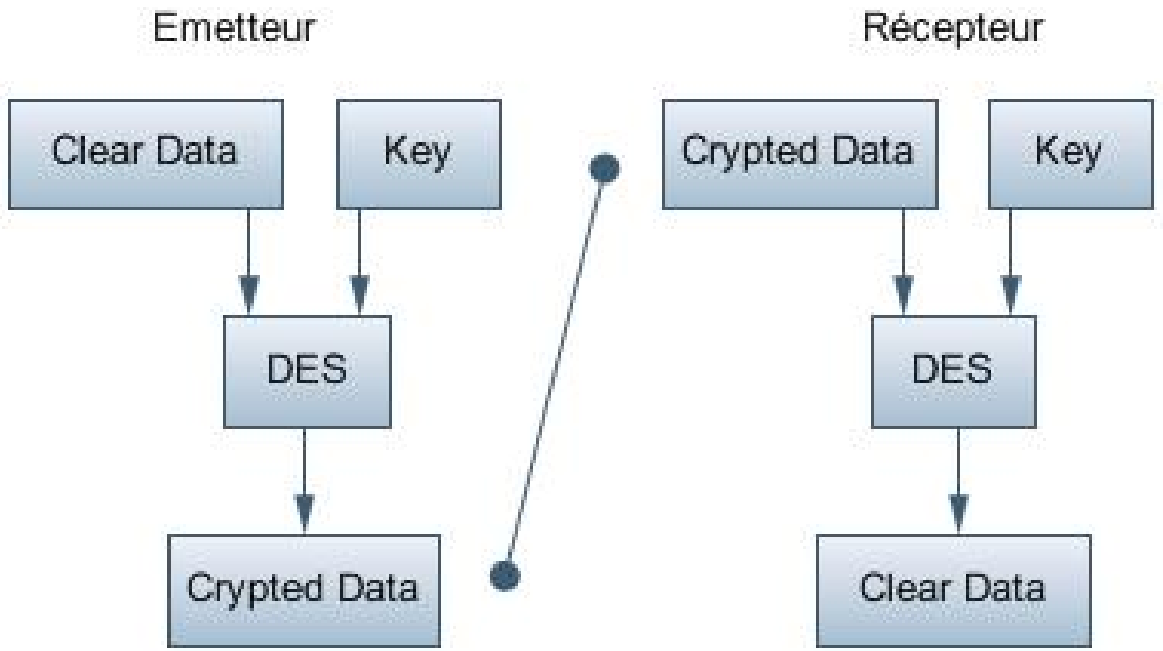
- authentification : empêche quelqu'un de changer le paquet en cours de route
- cryptage : empêche quiconque de lire les informations de gestion contenues dans un paquet
- estampillage du temps : empêche la réutilisation d'un paquet

```
1  #
2  # NET-SNMP
3  #
4  rouser USER [noauth|auth|priv [OID]]
5  rwuser USER [noauth|auth|priv [OID]]
6
7  rouser USER noauth|auth|priv -V NAME
8  rwuser USER noauth|auth|priv -V NAME
```

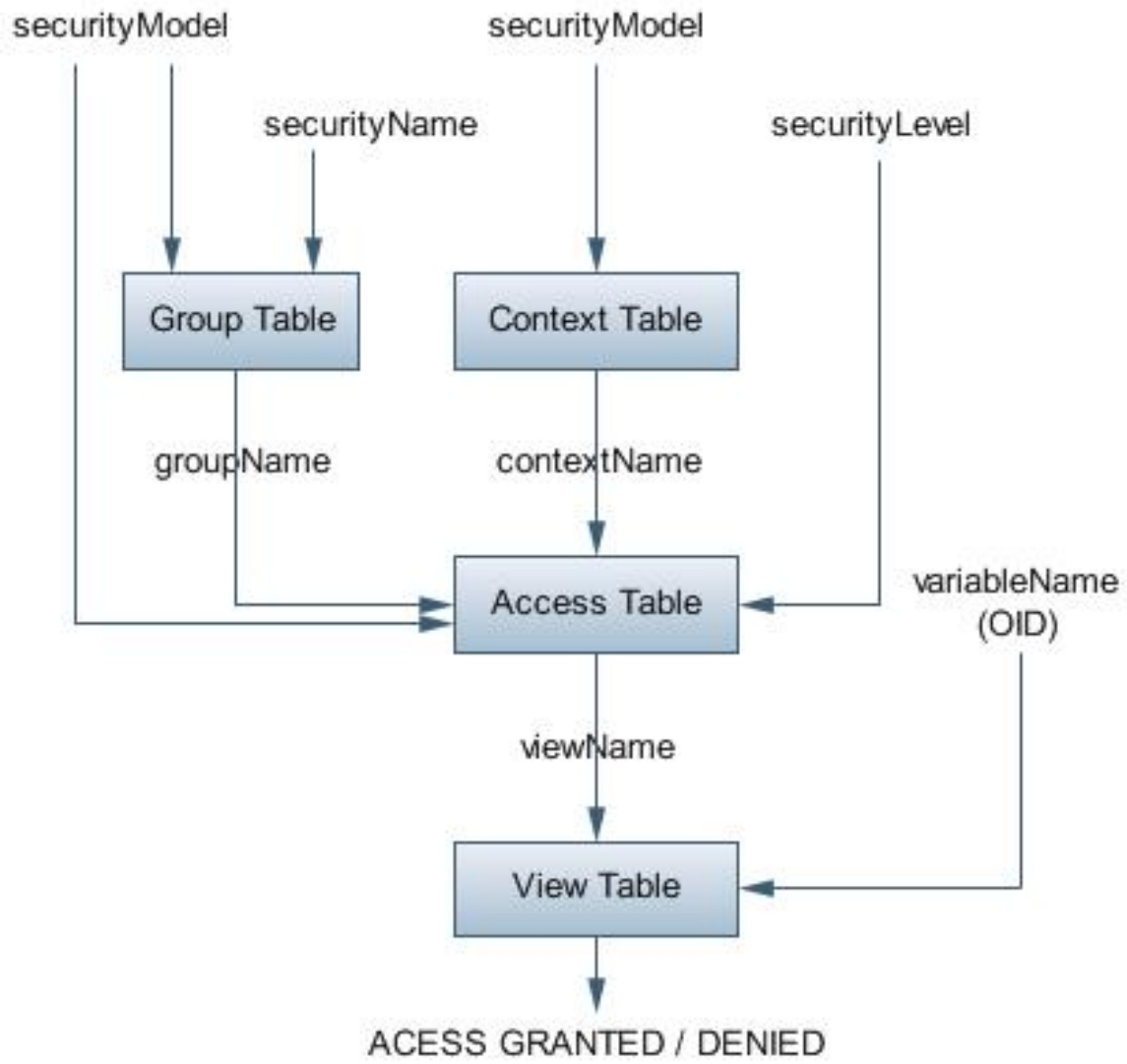
Security Model : USM - auth



Security Model : USM - priv



View-based Access Control Model : Algorithme



View-based Access Control Model : Exemple

```
1  # http://www.net-snmp.org/wiki/index.php/Vacm
2  # First, map the community name (COMMUNITY) into a security name
3  # (local and mynetwork, depending on where the request is coming from):
4  #       sec.name   source          community
5  com2sec local      localhost       secret42
6  com2sec custom_sec 192.168.1.0/24  public
7  # Second, map the security names into group names:
8  #                  sec.model sec.name
9  group custom_grp  v1         custom_sec
10 group custom_grp  v2c        custom_sec
11 group incremental usm        myuser      # SNMPv3 username == sec.name
12 # Third, create a view for us to let the groups have rights to:
13 #              incl/excl subtree                 mask
14 view all       included  .1
15 view custom_v  excluded  .1
16 view custom_v  included  sysUpTime.0
17 view custom_v  included  interfaces.ifTable
18 view mini_view excluded  .1                      80
19 view mini_view included  sysUpTime.0
20 view if_view   excluded  .1                      80
21 view if_view   included  sysUpTime.0
22 view if_view   included  ifTable
23 # Finally, grant the groups access to their views:
24 #                  context sec.model sec.level match read      write  notif
25 access MyRWGroup   ""      any       noauth    exact all       all    none
26 access custom_grp  ""      any       noauth    exact cust1_v   none   none
27 access incremental ""      usm       noauth    exact mini_view none   none
28 access incremental ""      usm       auth      exact if_view   none   none
29 access incremental ""      usm       priv      exact all_view  none   none
```

TRAP / INFORM

Description

- port UDP 162
- requête SNMP
- non acquité (TRAP) / acquité (INFORM, uniquement SNMPv2c et SNMPv3)
- transporte une notification SNMP

Système Réactif

- un évènement survient sur un noeud
- l'agent associé émet une requête SNMP notifiant l'évènement à un gestionnaire SNMP

Plan

Bibliothèques

- Perl : http ://search.cpan.org/perldoc ?Net : :SNMP
- Python : http ://pysnmp.sourceforge.net/
- Ruby : http ://snmplib.rubyforge.org/
- PHP : php5-snmp

Exemples

- getRequest

```
# ./getRequest sysDescr.0 sysName.0
sysDescr.0=FreeBSD david-server 7.2-RELEASE-p3
FreeBSD 7.2-RELEASE-p3 #0: Thu Aug 27 20:47:52 UTC 2009
root@angd.mathrice.fr:/usr/obj/usr/src/sys/ANGD amd64
sysName.0=david-server
#
```

- setRequest

```
# ./setRequest '1.3.6.1.2.1.1.5.0' 'snmp test server'
snmp before='david-server'
setting '1.3.6.1.2.1.1.5.0' to 'snmp test server'
snmp after='snmp test server'
#
```

- walk

```
# ./walk ifDescr
STRING: bge0
STRING: bge1
STRING: lo0
#
```

getRequest

- Ruby

```ruby
#!/usr/bin/env ruby
require 'rubygems'
require 'snmp'
SNMP::Manager.open(:Host=>'localhost') {|manager|
  ARGV.each {|varbind_name|
    manager.get(varbind_name).each_varbind {|varbind|
      print "#{varbind_name}=#{varbind.value.asn1_type}: #{varbind.value}\n"
    }
  }
}
```

- PHP

```php
#!/usr/bin/env php
<?php
$varbinds=$argv;
array_shift($varbinds);
foreach ($varbinds as $varbind_name) {
  $varbind_value=snmpget('localhost','public',$varbind_name);
  print "{$varbind_name}={$varbind_value}\n";
}
?>
```

setRequest

- Ruby

```ruby
#!/usr/bin/env ruby
require 'rubygems'
require 'snmp'

SNMP::Manager.open(:Host => 'localhost',:Community=>'private'){|manager|
  puts "snmp before='#{manager.get_value(ARGV[0])}'"
  puts "setting #{ARGV[0]} to '#{ARGV[1]}'"
  varbind = SNMP::VarBind.new(ARGV[0], SNMP::OctetString.new(ARGV[1]))
  manager.set(varbind)
  puts "snmp after='#{manager.get_value(ARGV[0])}'"
}
```

- PHP

```php
#!/usr/bin/env php
<?php
array_shift($argv);
$object_oid=$argv[0];
$object_value=$argv[1];
$before=snmpget('localhost','public',$object_oid);
print("snmp before=$before\n");
print("setting $object_oid to '$object_value'\n");
snmpset('localhost','private',$object_oid,'s',$object_value);
$after=snmpget('localhost','public',$object_oid);
print("snmp after=$after\n");
?>
```

walk

- Ruby

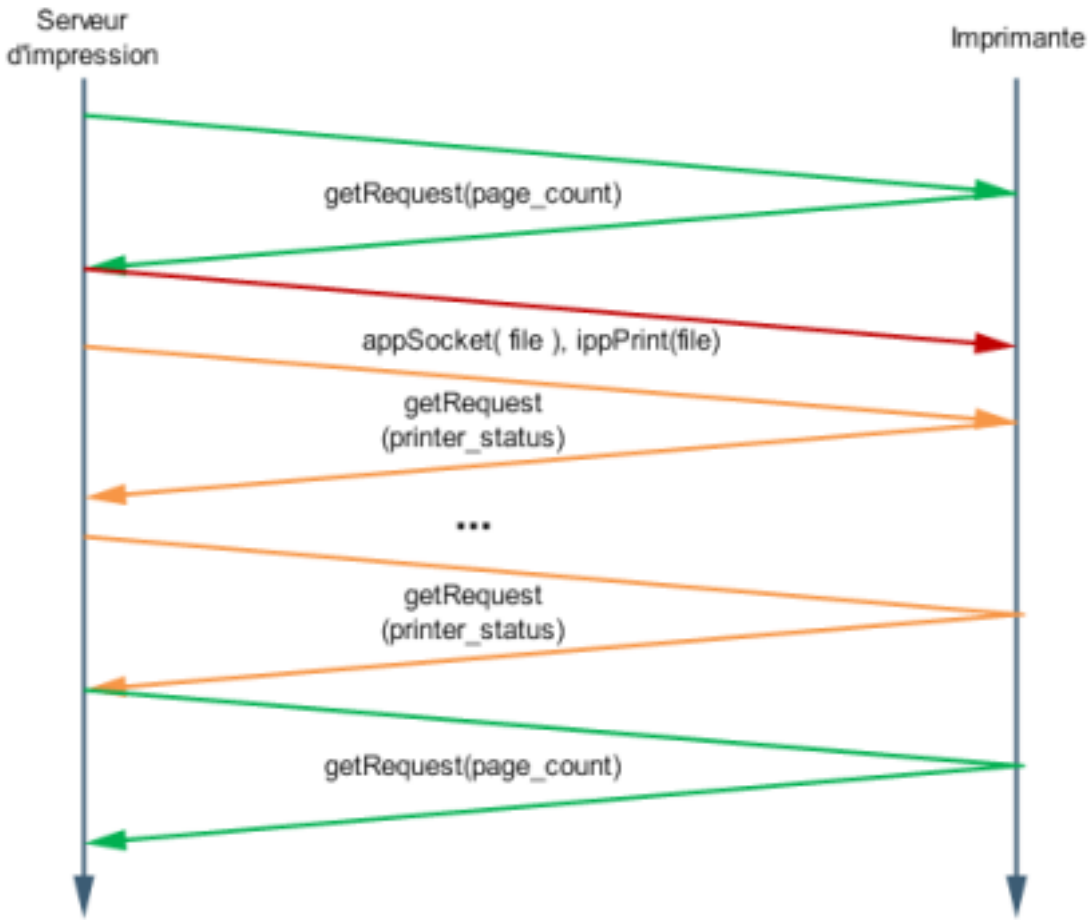
```ruby
#!/usr/bin/env ruby
require 'rubygems'
require 'snmp'
SNMP::Manager.open(:Host=>'localhost') { | manager |
  ARGV.each{|object_oid|
    manager.walk(object_oid){|response_objects|
      response_objects.each{|response_object|
        print "#{response_object.value.asn1_type}: #{response_object.value}\n";
      }
    }
  }
}
```

- PHP

```php
#!/usr/bin/env php
<?php
$objects_oid=$argv;
array_shift($objects_oid);
foreach ($objects_oid as $object_oid){
  $response_objects=snmpwalk('localhost','public',$object_oid);
  foreach ($response_objects as $response_object){
    print "$response_object\n";
  }
}
?>
```

Plan

Principe


Serveur d'impression
Imprimante
getRequest(page_count)
appSocket(file), ippPrint(file)
getRequest (printer_status)
...
getRequest (printer_status)
getRequest(page_count)

PRINTER-MIB

Printer Status	hrDeviceStatus	hrPrinterStatus	hrPrinterDetectedErrorState
Normal	running(2)	idle(3)	none set
Busy / Temporarily Unavailable	running(2)	printing(4)	
Non Critical Alert Active	warning(3)	idle(3) or printing(4)	could be : lowPaper, lowToner, or serviceRequested
Critical Alert Active	down(5)	other(1)	could be : jammed, noPaper, noToner, coverOpen, or serviceRequested
Unavailable	down(5)	other(1)	
Moving off-line Off-line	warning(3) down(5)	idle(3) or printing(4) other(1)	offline offline
Moving on-line	down(5)	warmup(5)	
Standby	running(2)	other(1)	

Résultat attendu : accounting.2009.3.log

Date	Printer	User	Page count
[04/*Mar*/2009 : 17 : 40 : 08 + 0100]	p26black	babar	1
[04/*Mar*/2009 : 18 : 10 : 21 + 0100]	p26color	mafalda	8
[05/*Mar*/2009 : 14 : 30 : 16 + 0100]	p26color	mafalda	1
[05/*Mar*/2009 : 18 : 08 : 25 + 0100]	p26black	mafalda	1
[06/*Mar*/2009 : 09 : 46 : 45 + 0100]	p26black	haddoc	10

Code : accounting.rb

```
1 require 'cups/backend'
2
3 backend=CUPS::Backend::Accounting.new
4 backend.print
```

Code : cups_backend_print.rb

```
1  module CUPS
2    module Backend
3      class Accounting
4        def print
5          pre_print_page_count = self.printer.page_count
6          backend_return_code  = self.printer.print()
7          printing_in_progress = true
8          while printing_in_progress
9            self.printer.get_status
10           if (self.printer.is_idle?) then
11             self.printer.status_stability-=1
12             if (self.printer.status_idle_is_stable?) then
13                 printing_in_progress = false
14             end
15           else
16             self.printer.status_stability = 5
17           end
18           sleep 1
19         end
20         post_print_page_count = self.printer.page_count
21         self.printer.printed_page_count = post_print_page_count -
               pre_print_page_count
22         self.account
23         exit backend_return_code
24       end
25     end
26   end
27 end
```

Code : cups_backend.rb

```ruby
require 'cups/printer'

module CUPS
  module Backend
    class Accounting
      def initialize
        @log_dir = ...
        @printer = Printer.new( ... )
      end
      attr_reader :printer, :log_dir
    end
  end
end
```

Code : cups_printer.rb

```ruby
require 'cups/snmp'

module CUPS
  class Printer
    def initialize(h={})
      @snmp = CUPS::Snmp.new
      @status_code = {
      :by_name  => {
        :other    => 1, :unknown  => 2,
        :idle     => 3, :printing => 4,
        :warmup   => 5 },
      :by_code  => {
        1 => 'other', 2 => 'unknown',
        3 => 'idle', 4 => 'printing',
        5 => 'warmup' }
      }
      @status      = @status_code[:other]
      @uri         = h[:uri]
      @accounting = h[:accounting]
      @printed_page_count = 0
      @status_stability = 5
      @job         = JOB.new
    end
    attr_reader   :status_code, :status, :uri, :snmp, :job, :accounting
    attr_accessor :printed_page_count, :status_stability
  end
end
```

Code : cups_printer_misc.rb

```ruby
module CUPS
  class Printer
    def page_count
      SNMP::Manager.open(:Host=>self.uri.host,:Version=>self.snmp.version,:
          Community=>self.snmp.community){|manager|
        manager.get_value(self.snmp.oid[:page_count]).to_i
      }
    end
    def get_status
      SNMP::Manager.open(:Host=>self.uri.host,:Version=>self.snmp.version,:
          Community=>self.snmp.community){|manager|
        @status=manager.get_value(self.snmp.oid[:printer_status]).to_i
      }
    end
    def is_printing?
      self.status == self.status_code[:by_name][:printing]
    end
    def is_idle?
      self.status == self.status_code[:by_name][:idle]
    end
    def status_idle_is_stable?
      self.status_stability == 0
    end
  end
end
```

Code : cups_snmp_job.rb

```ruby
module CUPS
  class Snmp
    def initialize
      @version   = :SNMPv1
      @community = 'public'
      @oid       = {
        # cf http://www.oidview.com/mibs/0/Printer-MIB.html
        :page_count     => '1.3.6.1.2.1.43.10.2.1.4.1.1',
        :printer_status => '1.3.6.1.2.1.25.3.5.1.1.1'
      }
    end
    attr_reader :version, :community, :oid
  end
  class JOB
    attr_accessor :jid, :file, :user_name, :title, :copies, :options, :file_name
  end
end
```

Code : cups_uri.rb

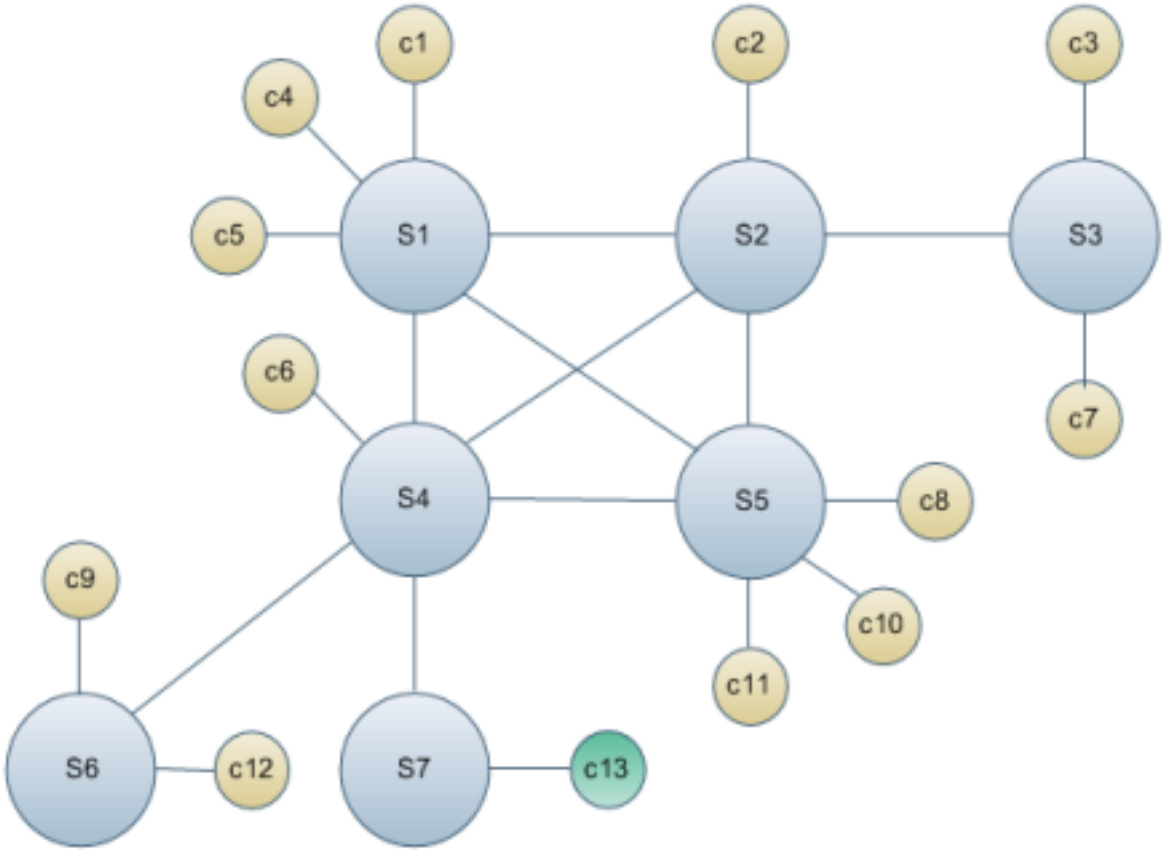
```ruby
module CUPS
 class URI
  def initialize(uri,backend)
   @backend,@host,@port = nil,nil,nil
   if Regexp.new("^#{backend}://([^:]+)://([^:]+):?([0-9]*)$").match(uri).nil?
       then
    STDOUT.puts "URI format error:"
    STDOUT.puts " expected accounting://<cups_uri>"
    STDOUT.puts " got #{uri}"
   else
    @backend = $1
    @host    = $2
    @port    = $3
    @uri     = %Q{#{@backend}://#{@host}#{(@port.nil?) ? "" : ":#{@port}"; }}
   end
  end
  attr_reader :backend, :host, :port, :uri
 end
end
```

Code : cups_backend_accounting_account.rb

```ruby
module CUPS
  module Backend
    class Accounting
      def account
        "#time}" =~/\S+\s\S+\s\S+\s\S+\s(\S+)\s\S+/
        time_zone=$1
        time_string=time.strftime("%d/%b/%Y:%X")
        accounting_filename=[
          "#{self.log_dir}/",
          "#{self.backend_name}.",
          "#{time.year}.",
          "#{time.strftime('%m')}.",
          "log"
        ].join
        accounting_line=[
          "[#{time_string} #{time_zone}]",
          "#{self.printer.uri.host}",
          "#{self.printer.job.user_name}",
          "#{self.printer.printed_page_count}"
        ].join(' ')
        File.open(accounting_filename, File::WRONLY|File::APPEND|File::CREAT){|
            accounting_file|
          accounting_file.puts accounting_line
        }
      end
    end
  end
end
```

Plan

Principe



CISCO-CDP-MIB.mib

```
1  ciscoCdpMIB    MODULE-IDENTITY     ::= { ciscoMgmt 23 }
2
3  ciscoCdpMIBObjects OBJECT IDENTIFIER ::= { ciscoCdpMIB 1 }
4
5  cdpInterface    OBJECT IDENTIFIER ::= { ciscoCdpMIBObjects 1 }
6  cdpCache        OBJECT IDENTIFIER ::= { ciscoCdpMIBObjects 2 }
7  cdpGlobal       OBJECT IDENTIFIER ::= { ciscoCdpMIBObjects 3 }
8
9  cdpInterfaceTable OBJECT-TYPE
10     SYNTAX      SEQUENCE OF CdpInterfaceEntry
11     MAX-ACCESS not-accessible
12     STATUS      current
13     DESCRIPTION
14             "The (conceptual) table containing the status of CDP on
15             the device's interfaces."
16     ::= { cdpInterface 1 }
17
18 cdpInterfaceEntry OBJECT-TYPE
19     SYNTAX      CdpInterfaceEntry
20     MAX-ACCESS not-accessible
21     STATUS      current
22     DESCRIPTION
23             "An entry (conceptual row) in the cdpInterfaceTable,
24             containing the status of CDP on an interface."
25     INDEX       { cdpInterfaceIfIndex }
26     ::= { cdpInterfaceTable 1 }
```

CISCO-CDP-MIB.yaml

```
1  ---
2  ciscoCdpMIB:              1.3.6.1.4.1.9.9.23
3  ciscoCdpMIBObjects:       1.3.6.1.4.1.9.9.23.1
4  cdpCache:                 1.3.6.1.4.1.9.9.23.1.2
5  cdpCacheTable:            1.3.6.1.4.1.9.9.23.1.2.1
6  cdpCacheEntry:            1.3.6.1.4.1.9.9.23.1.2.1.1
7  cdpCacheAddress:          1.3.6.1.4.1.9.9.23.1.2.1.1.4
8  cdpCacheVersion:          1.3.6.1.4.1.9.9.23.1.2.1.1.5
9  cdpCacheDeviceId:         1.3.6.1.4.1.9.9.23.1.2.1.1.6
10 cdpCacheDevicePort:       1.3.6.1.4.1.9.9.23.1.2.1.1.7
11 cdpCachePlateform:        1.3.6.1.4.1.9.9.23.1.2.1.1.8
12 cdpCachePhyLocation:      1.3.6.1.4.1.9.9.23.1.2.1.1.23
```

Résultat attendu

name	ip	contact	location	description
C3550	172.16.0.40	s.info@	Bat 26 salle machine	...12.2(50)SE3...
C2950-26-A-4	172.16.0.3	s.info@	Bat 26 armoire A	...12.1(22)EA11...
C2950-26-SM-1	172.16.0.41	s.info@	Bat 26 salle machine	...12.1(22)EA9...
C3548-26-A-1	172.16.0.17	s.info@	Bat 26 armoire A	...12.0(5)WC17...
C3548-26-A-2	172.16.0.18	s.info@	Bat 26 armoire A	...12.0(5)WC17...
C3548-26-B-1	172.16.0.12	s.info@	Bat 26 armoire B	...12.0(5)WC17...
C2950-26-B-3	172.16.0.8	s.info@	Bat 26 armoire B	...12.1(22)EA11...
C3548-26-B-2	172.16.0.16	s.info@	Bat 26 armoire B	...12.0(5)WC17...
C2950-26-B-4	172.16.0.4	s.info@	Bat 26 armoire B	...12.1(22)EA11...
C2950-26-A-3	172.16.0.7	s.info@	Bat 26 armoire A	...12.1(22)EA11...
C2960G-26-SM-2	172.16.0.42	s.info@	Bat 26 salle machine	...12.2(40)SE...

Code : cdp_node_snmpget.rb

```ruby
require 'pp'
require 'snmp'
class Node
  def snmpget(host,oid)
    value=nil
    begin
      SNMP::Manager.open(
        :Host=>host,
        :Version=>:SNMPv1,
        :Community=>'public'
      ){|manager|
        value=manager.get_value(oid)
      }
    rescue Exception => _exception_
      pp _exception_
    end
    return value
  end
end
```

Code : cdp_node.rb

```ruby
require 'cdp_node_snmpget'
class Node
  def initialize(h={})
    @ip = h[:ip]
  end
  attr_accessor :ip
  def sysName()
    return self.snmpget(ip,'sysName.0')
  end
  def sysDescr()
    return self.snmpget(ip,'sysDescr.0')
  end
  def sysLocation()
    return self.snmpget(ip,'sysLocation.0')
  end
  def sysUptime()
    return self.snmpget(ip,'sysUpTime.0')
  end
  def sysContact()
    return self.snmpget(ip,'sysContact.0')
  end
  def commit
    @@db.execute('INSERT INTO devices(name,ip,description,contact,location,
        uptime,last_update) VALUES (?,?,?,?,?,?,?)',
      sysName,ip,sysDescr,sysContact,sysLocation,sysUptime,Time.now )
  end
end
```

Code : cdp_topology_discover_loop.rb

```
1  require 'pp'
2  require 'snmp'
3  class Topology
4    def discover_loop(root_node)
5      begin
6        neighbors = {}
7        SNMP::Manager.open(:Host=>root_node.ip,:Version=>:SNMPv1,:Community=>'
           public',:MibModules=>["CISCO-CDP-MIB"]){|manager|
8          manager.walk('cdpCacheAddress'){|result|
9            neighbor_ip=hexa_to_decimal(result.value)
10           neighbor = nodes[neighbor_ip]
11           if neighbor.nil? then
12             neighbor = Node.new(:ip=>neighbor_ip)
13             nodes[neighbor_ip]=neighbor
14           end
15           neighbors[neighbor_ip]=neighbor
16         }
17       }
18       nodes_already_treated.push root_node.ip
19       neighbors.each{|_,neighbor|
20         begin
21           neighbor.commit
22           discover_loop(neighbor)
23         end unless nodes_already_treated.include? neighbor.ip
24       }
25     rescue Exception => _exception_
26       pp _exception_
27     end
28   end
29 end
```

Code : cdp_topology.rb

```ruby
require 'cdp_node'
require 'cdp_topology_discover_loop'
class Topology
  def initialize(h={})
    @root = Node.new(:ip=>h[:root_node_ip])
    @root.commit
    @nodes_already_treated = []
    @nodes                 = {}
  end
  attr_reader :root, :nodes, :nodes_already_treated
  def hexa_to_decimal(hexa_string)
    return hexa_string.unpack("C*").map{|n| "%d" % n}.join('.')
  end
  def discover
    discover_loop(root)
  end
end
```

Code : cdp.rb

```
1  require 'sqlite3'
2  require 'cdp_topology'
3
4  @@db = SQLite3::Database.new('topology.sqlite')
5  @@db.execute('DELETE FROM devices')
6  topology=Topology.new(:root_node_ip=>'157.136.84.40')
7  topology.discover
```

Merci