PRACTICAL EXERCISE 2 (W1_PE2): OUR FIRST SNMP AGENT

This second practical exercice consists in presenting an overview of the considered architecture, and then building and executing our first SNMP agent.

1. Lab Architecture

In this lab, we will use a particular MIB specification (written with the standard SMI language) designed specifically for pedagogical purposes. This MIB models a simple coffee machine with its status variables such as temperature, water level etc. and of course a control switch allowing to "boil" simulated coffee (Note : nous considérons ici à titre illustratif la modélisation d'une machine à café avec un interrupteur permettant de faire chauffer/bouillir le café).

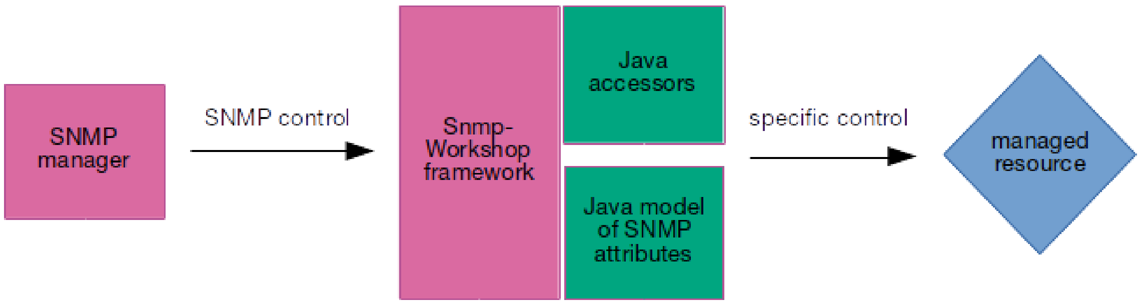
This approach will allow us to:

- get rid of the complexity of existing industrial MIBs, allowing us to focus on essentials;
- build step by step the complete SNMP control chain from Manager requests to Agent application code.

The goal of this lab is therefore to build a realistic coffee machine simulator based on a fully manageable SNMP agent and on a suitable manager. The coffee machine agent will provide the following properties:

- Monitoring of ground coffee, available water, water temperature and others;
- The ability to make coffee on demand through some SNMP requests;
- Choice of coffee options;
- Capabilities for problem reporting.

The following figure gives a summary of the complete coffee pot system. In fact it is a reminder of the previous execution phase figure.



The coffee pot system

2. The Agent

2.1 The MIB of the Agent

The first step of the lab consists in the development of an SNMP agent for the coffee pot. To this purpose, a suitable MIB specification is provided in the file: `mibs/CoffeePot.mib`.

Please read and understand this specification, as this is the first important moment of the lab. Take your time, load the MIB file with the editor of your choice (mousepad, xcoral, vi ...) and study its structure and details. (note : vous pouvez simplement utiliser la commande mousepad pour afficher/éditer la MIB, comme donné dans l'exemple ci-dessous.)

```
$ cd /home/user/snmp-lab/SnmpWorkShop-nojava
$ mousepad ./mibs/CoffeePot.mib &
```

QUESTION W1.PE2.1: MIB UNDERSTANDING (1/1 point)

The CoffeePot.mib specification defines a series of fictitious OID nodes in order to avoid collisions with existing MIBs.

Check every proposal that applies: (NA=4)

- [✓] 1.3.6.1.4.1 is the standard OID prefix for PRIVATE nodes
- [ ] 1.3.6.1.4.1 is a randomly chosen OID prefix in order to avoid collisions.
- [✓] 1.3.6.1.4.1.domestic is a fictitious node.
- [✓] 1.3.6.1.4.1.domestic.householdElectricals designates a fictitious node for all electric machines at home.
- [✓] All OIDs of the coffee pot MIB will begin with 1.3.6.1.4.1.9999.1.2

✓

Correct:
Correct! This is an IETF standard.
Yes!
Yes!
Yes!

One of the main properties of this specification is that coffee is generated by a SNMP SET request to the value "on", of the attribute CoffeePotControl. Indeed, SNMP does not provide any procedural action call request in itself. This limitation is overridden by our scheme that gives a particular semantics to the "on" value of this attribute (i make some coffee).

2.2 Desktop Organization

Prepare the desktop of your virtual machine with two terminals, both situated at the root of the lab tree (SnmpWorkshop folder). The terminals will have following distinct roles:

- on one hand, a normal user terminal that will serve for edition, compilation, and that will be used for launching the Wireshark network analyzer tool.
- on the other hand, a root-owned terminal that will be used to execute the Agent.

So if you do not have an active terminal (but you should, after the previous exercise) just start a new terminal with the "terminal emulator" icon of the dock at the bottom of the screen, go to the lab folder and load the environment variables, with the following commands:

```
$ cd /home/user/snmp-lab/SnmpWorkShop-nojava
$ source set_envt.sh
```

Repeat the same with **another terminal**, but when in the SnmpWorkshop folder, become root and then load the environment variables. (note : vous ouvrez un deuxième terminal que vous utiliserez en tant que root).

```
$ sudo su            (note: you become root)
# cd /home/user/snmp-lab/SnmpWorkShop-nojava
# source set_envt.sh
```

2.3 Hollow Agent Generation

Our next step is to build a "hollow" Agent i an Agent obtained without specific coding at all, just by compiling the MIB file and linking to the SnmpWorkshop framework. The result is a complete working SNMP V1 Agent, however its behaviour is limited to the defaults generated by the `smic` compiler.

So this step consists of following actions:

- test and validation of the SNMP Agent.

We will now proceed to the actions one by one. Go to our working folder in the normal user terminal (note : c'est-à-dire en utilisant le premier terminal):

```
$ cd agent-step1
```

2.3.1 MIB Compilation

The MIB specification of the coffee pot (file `CoffeePot.mib` in the `mibs` folder) is compiled by following command:

```
$ smic ../mibs/CoffeePot.mib
```

This step generates new files for SNMP attribute handling as well as a default GET/SET behaviour class. Even if youre not a Java specialist, it is useful to check these files!

2.3.2 Building the Hollow Agent

The generated Java code can now be Java-compiled by the command:

```
$ build_all
```

You normally now have a complete operational SNMP Agent with "hollow" default behaviour. Let's test it!

2.3.3 Execution and Test of the Agent

To this purpose we will now execute the Agent and explore its mechanics using two tools:

- `wireshark`, which is a network analyser software allowing to decode in particular SNMP packets;
- `browser`, which is a graphical MIB-browsing SNMP Manager application.

In the user terminal start Wireshark as background task with the & character:

```
$ wireshark &
```

Configure Wireshark to capture packets from the "Loopback: lo" interface and to filter SNMP packets (note : il suffit de double cliquer sur "Loopback: lo" pour démarrer la capture sur cette interface.)

analyze packet contents. If necessary, consult appropriate tutorials for more details.

On the root terminal, go to the working folder agent-step1 and launch your Agent with the start_agent command:

```
# cd agent-step1
# start_agent 161 public
```

Note: The two parameters of the `start_agent` command mean the following: 161 is the *port* on which the Agent will listen (and 161 is the standard SNMP port); "public" is the SNMP password (also called "community" in SNMP speak) that will allow our requests to be accepted by the Agent. Of course, the Manager will have to honor these two conventions as well, in order for its SNMP requests to be passed to the Agent.

If everything went well, the Agent should start and indicate its activity on the terminal. It should also list all its known attributes along with their OIDs.

Now in the non-root terminal, still available, launch the MIB browser software as background task:

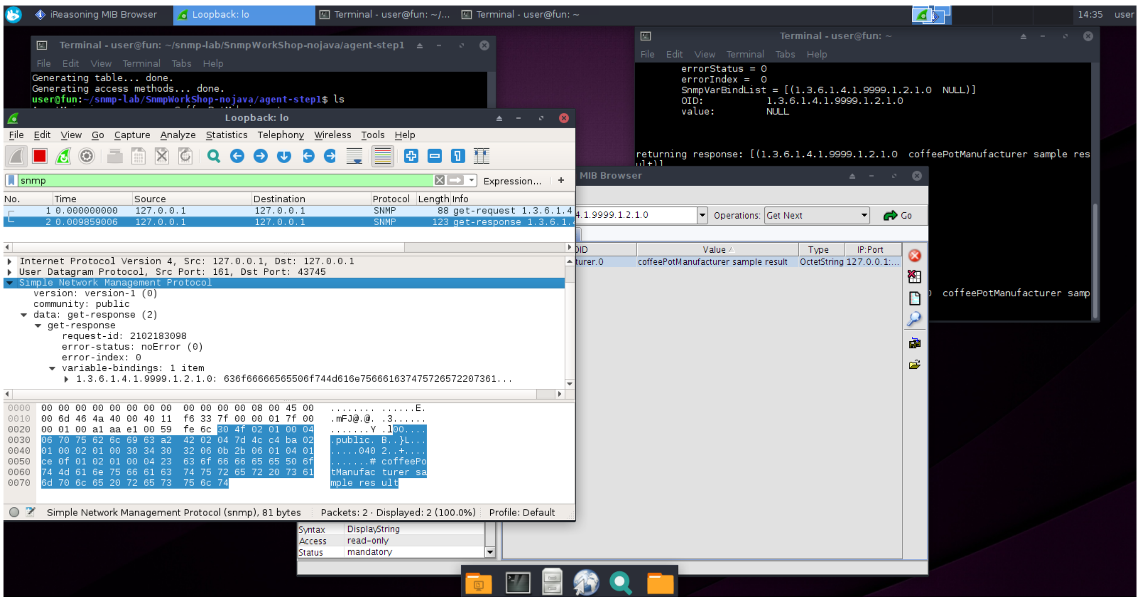
```
$ browser &
```

The first step is to load our coffeepot MIB file with the "File | Load Mibs" menu. Also, configure the address (127.0.0.1 or localhost) of the Agent as well as its port (161) and community (public as read/write community) through the "Advanced..." menu.

Now test a first GET request on any of the attributes of the MIB: just right-click on an attribute of the list on the left, and choose the GET option. Try different attributes as well as SET requests on the writeable attributes. Everything should work seamlessly! But of course, note the always similar responses given by the Agent ("909090"): this is due to the default code generated by the smic compiler, of course.

Note: if your SNMP requests do not yield immediate responses (and a Timeout notification appears), check the following:

- is the Agent running?
- has the Agent been started with port 161 and `public` community?



```
Terminal - user@fun: ~/snmp-lab/SnmpWorkShop-nojava/agent-step1
Generating table... done.
Generating access methods... done.
user@fun:~/snmp-lab/SnmpWorkShop-nojava/agent-step1$ ls
```

```
Loopback: lo
snmp
No.   Time          Source      Destination   Protocol  Length Info
  1 0.000000000    127.0.0.1   127.0.0.1     SNMP          88 get-request 1.3.6.1.4
  2 0.009859006    127.0.0.1   127.0.0.1     SNMP         123 get-response 1.3.6.1.

▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ User Datagram Protocol, Src Port: 161, Dst Port: 43745
▾ Simple Network Management Protocol
    version: version-1 (0)
    community: public
  ▾ data: get-response (2)
    ▾ get-response
        request-id: 2102183098
        error-status: noError (0)
        error-index: 0
      ▾ variable-bindings: 1 item
        ▸ 1.3.6.1.4.1.9999.1.2.1.0: 636f66666565506f744d616e7566616374757265722073 61...

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 6d 46 4a 40 00 40 11  f6 33 7f 00 00 01 7f 00   .mFJ@.@. .3......
0020  00 01 00 a1 aa e1 00 59  fe 6c 30 4f 02 01 00 04   .......Y .l0O....
0030  06 70 75 62 6c 69 63 a2  42 02 04 7d 4c c4 ba 02   .public. B..}L...
0040  01 00 02 01 00 30 34 30  32 06 0b 2b 06 01 04 01   .....040 2..+....
0050  ce 0f 01 02 01 00 04 23  63 6f 66 66 65 65 50 6f   .......# coffeePo
0060  74 4d 61 6e 75 66 61 63  74 75 72 65 72 20 73 61   tManufac turer sa
0070  6d 70 6c 65 20 72 65 73  75 6c 74                  mple res ult

Simple Network Management Protocol (snmp), 81 bytes   Packets: 2 · Displayed: 2 (100.0%)   Profile: Default
```

```
Terminal - user@fun: ~
        errorStatus = 0
        errorIndex =  0
        SnmpVarBindList = [(1.3.6.1.4.1.9999.1.2.1.0  NULL)]
        OID:            1.3.6.1.4.1.9999.1.2.1.0
        value:          NULL


returning response: [(1.3.6.1.4.1.9999.1.2.1.0  coffeePotManufacturer sample res
ult)]
```

MIB Browser — 4.1.9999.1.2.1.0 — Operations: Get Next — Go

OID	Value	Type	IP:Port
turer.0	coffeePotManufacturer sample result	OctetString	127.0.0.1:...

Syntax	DisplayString
Access	read-only
Status	mandatory

Your desktop should look like this screenshot: you see Wireshark at the front analyzing a SNMP response packet, in the middle the MIB browser and behind those, the running Agent in a terminal.

QUESTION W1.PE2.2: SNMP PACKET ANALYZIS (1/1 point)

On the Wireshark UI, select a SNMP packet of your choice and develop all its layers.

Check all that apply. (NA=1)

- ☐ fortunately, the SNMP password (community) does not appear on the capture!
- ☐ the SNMP password is encrypted.
- ☑ the SNMP password is transmitted in clear text.

✓

Correct:
Of course, the capture bears all data of the packet, including the password.
Unfortunately it is not.
Indeed, and that explains why many criticized SNMP V1 security at that time.