Installing Qualcomm test environment

In this tutorial, we instruct how to install a Qualcomm test environment on a host that is running a Ubuntu Linux operating system. This tutorial is valid also for other host operating systems excluding the VirtualBox installation part. We do not need any Qualcomm related software or a connection to the Qualcomm realm in order to complete the installation.

The test environment installed here is based on open-source software that can be used to familiarize ourselves with the concepts in the Qualcomm project before we have access to the Qualcomm realm. The test environment installed here is not used in the Qualcomm realm, but a new environment is installed.

Installing Ubuntu on a VirtualBox

We will install server software which is best run as root user to avoid access privilege issues. To avoid breaking our existing operating system installation, we recommend installing the test environment in a virtual machine. Instructions for installing Ubuntu on a virtual machine are given here: How to run Ubuntu Desktop on a virtual machine using VirtualBox.

All commands given in this tutorial are to be executed in the **terminal of the virtual machine**.

For our purposes, it is enough to use *"Minimal Install"* and check the *"Install third-party software"* checkbox.

Remember to install the *"Guest Additions"* as instructed at the document behind the link for a better copy-paste support between the host machine and the virtual machine.

Upgrading the default packages

To upgrade the default packages:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get autoclean
sudo apt-get autoremove
```

Sysrepo ecosystem

The Qualcomm test environment is effectively a Sysrepo ecosystem that consists of the following modules.

- **Netopeer2 CLI** - Example command-line NETCONF client that can be used as a simple Network Management System (NMS).
- **Netopeer2 NETCONF server** - Complement Sysrepo with the Open Source Netopeer2 NETCONF server to remotely manage applications via NETCONF.
- **Sysrepoctl** - Command line tool for changing YANG schemas of Sysrepo. That is installing, removing and updating schemas.
- **Sysrepocfg** - Command line tool for stored YANG data management. This tool allows importing, exporting and editing configuration that complies with an YANG schema.
- **Sysrepo-plugind** - Simple daemon that groups all available Sysrepo plugins into a single process.

Installing the required dependencies

Following binaries and development libraries need to be installed before we can start to install the Qualcomm test environment.

```
sudo apt-get install git cmake build-essential bison flex libpcre3-dev
libev-dev libavl-dev libprotobuf-c-dev protobuf-c-compiler swig
python-dev lua5.2 pkg-config libpcre++-dev openssl libssl-dev
libcrypto++-dev zlib1g-dev libpcre2-dev libssh-dev uncrustify
libcmocka-dev
```

Checking libssh version

Libnetconf2 requires the libssh version to be 0.9.6. Ubuntu 20.04 has version 0.9.3 installed whereas Ubuntu 21.10 has 0.9.6 version installed. We can check the installed libssh version with

```
sudo apt show libssh-4
```

If we have a version less than 0.9.6 installed we need to build and install the 0.9.6 version from the source code as instructed below. Otherwise we can skip the libshh installation step.

Installing libssh 0.9.6

```
wget \
  https://git.libssh.org/projects/libssh.git/snapshot/libssh-0.9.6.tar.gz
tar -xf libssh-0.9.6.tar.gz
rm libssh-0.9.6.tar.gz
```

```
cd libssh-0.9.6
mkdir build
cd build
cmake ..
make
sudo make install
```

Create installation directory

Switch user to root to avoid access privilege issues when running the installed services.

```
sudo -i
mkdir NetConfServer
cd NetConfServer
```

Installing libyang

```
git clone https://github.com/CESNET/libyang.git
cd libyang
mkdir build
cd build
cmake ..
make && make install
```

Installing sysrepo

```
cd ~/NetConfServer
git clone https://github.com/sysrepo/sysrepo.git
cd sysrepo
mkdir build
cd build
cmake ..
make && make install
```

Installing libnetconf2

```
cd ~/NetConfServer
git clone https://github.com/CESNET/libnetconf2.git
cd libnetconf2
mkdir build
cd build
cmake ..
make && make install
```

Installing Netopeer2

```
cd ~/NetConfServer
ldconfig /usr/local/lib
git clone https://github.com/CESNET/netopeer2.git
cd netopeer2
mkdir build
cd build
cmake ..
make && make install
```

Troubleshooting Netopeer2

In case of dynamic link errors during *make install netopeer2* add already installed libraries to *LD_LIBRARY_PATH*.

```
export LD_LIBRARY_PATH=”$LD_LIBRARY_PATH:/usr/local/lib”
```

Or update lib cache and dynamic links:

```
sudo ldconfig /usr/local/lib
```

In case of shared memory issues (this happens if you’re building libraries as a normal user), this was suggested as a workable solution (apparently it’s about some specific kernel configuration in Debian/Ubuntu, that changes behaviour of shared memory access permissions):

```
sudo sysctl fs.protected_regular=0
```

Testing the build

List YANG modules in sysrepo.

```
sysrepoctl -l
```

Display Netopeer2 help.

```
netopeer2-server -h
```

See Sample NETCONF Session

Testing the sysrepo oven plugin

See Plugin Example for the description of the oven plugin. Section *"Trying it out"* has information on how to run commands directly with sysrepo-plugind.

Setting up the oven plugin

As root, command:

```
cd /root/NetConfServer/sysrepo/examples/plugin
mkdir -p /usr/local/lib/sysrepo-plugind/plugins
sysrepoctl -i oven.yang
cd /root/NetConfServer/sysrepo/build/examples
cp oven.so /usr/local/lib/sysrepo-plugind/plugins
```

Set root password

Set root password, netopeer2 client prompts for it.

```
passwd
```

Setup netopeer2-server, netopeer2-cli and sysrepo-plugind

Open **four** terminals as root. In the first three command:

1. `sysrepo-plugind -d -v4`
2. `netopeer2-server -d -v3`
3. `netopeer2-cli -v3`
4. This terminal is used for sysrepocfg, no commands to be issued yet.

In netopeer2 client terminal:

View help:

```
help
```

Connect to netopeer2 server, give root password when prompted. Netopeer2-server should acknowledge the login attempt.

```
connect
```

Get the running netopeer2 server configuration.

```
get-config --source running
```

Test the oven plugin by executing a user remote procedure call. Command user-rpc opens a text editor where the XML is to be inserted. After the XML is saved and the editor is exited, sysrepo-plugind terminal should print some informational output.

```
user-rpc
```

```
<insert-food xmlns="urn:sysrepo:oven">
    <time>on-oven-ready</time>
</insert-food>
```

In the fourth, currently unused terminal, edit the running sysrepo configuration to turn the oven on. We can use any text editor that is installed on the machine. The sysrepocfg command can be issued from any directory.

```
sysrepocfg --edit=vim --datastore running
```

This opens an XML file. Add the following lines at the end of the XML, se the file and exit.

```
<oven xmlns="urn:sysrepo:oven">
    <turned-on>true</turned-on>
    <temperature>200</temperature>
</oven>
```

After ~4 seconds there should be a notification of the food being put into the oven in sysrepo-plugind terminal.

To remove the food from the oven, use netopeer2-cli terminal.

```
user-rpc
```

```
<remove-food xmlns="urn:sysrepo:oven"/>
```

Note that we can also insert and remove food from the oven also from sysrepocfg terminal with:

```
sysrepocfg --rpc=vim
```

Finally, disconnect the client from the server. In netopeer2-cli terminal.

```
disconnect
```

Setting up Call Home for Oven example

In a NETCONF Call Home procedure a NETCONF server initiates a secure connection to a NETCONF client. This is the reverse of the normal process.

See It's Time to Call Home for more information on Call Home.

In sysrepocfg terminal, export the running configuration into file named running_conf.xml.

```
sysrepocfg -Xrunning_conf.xml
```

View file SSH Call Home example configuration file

```
/root/NetConfServer/netopeer2/example_configuration/ssh_callhome.xml
```

And copy the <call-home> … </call-home> XML block to clipboard or any place from where it c an be pasted.

Open the running_conf.xml for editing in any text editor. Remove the <listen> … </listen> XML block and replace it with the <call-home> … </call-home> XML block that we copied earlier. Save the running_conf.xml file and exit.

Import the modified configuration in sysrepocfg terminal.

```
sysrepocfg -Irunning_conf.xml
```

In netopeer2-cli terminal listen for SSH Call Home. Note that we must have disconnected the client first.

```
listen
```

We should now see line (with varying session id):

```
[INF]: LN: Call Home client "default-client" session 9 established.
```

in the netopeer2 server terminal. We can now use the netopeer2-cli to insert or remove food from the oven as instructed earlier, only this time the session was established using Call Home.