OpenDaylight Controller:Netconf:Testtool

Netconf testtool

Netconf testtool (or netconf device simulator) is a tool that:

- Simulates 1 or more netconf devices
- Is suitable for scale testing
- Uses core implementation of netconf server from ODL controller
- Generates configuration files for controller so that controller distribution (karaf) can easily connect to all simulated devices
- Provides broad configuration options
- Supports notifications

Building testtool

1. Check out latest netconf repository from git
2. Dive into opendaylight/netconf/tools/netconf-testtool/ folder
3. Build testtool using **mvn clean install** command

Downloading testtool

Netconf-testtool is now part of default maven build profile for controller and can be also downloaded from nexus. The executable jar for testtool can be found at: nexus

Running testtool

1. After successful build, dive into opendaylight/netconf/tools/netconf-testtool/target/ folder and there is file netconf-testtool-1.0.0-SNAPSHOT-executable.jar (or if downloaded from nexus just take that jar file)
2. Execute this file using e.g:

```
java -Xmx1G -XX:MaxPermSize=256M -jar netconf-testtool-1.0.0-SNAPSHOT-executable.jar
```

This execution runs the testtool with default for all parameters and you should see this log output from the testtool :

```
  10
  :
  31
  :
  08.206 [main] INFO  o.o.c.n.t.t.NetconfDeviceSimulator - Starting 1, SSH simulated devices starting on port 17830

10
  :
  31
  :
  08.675 [main] INFO  o.o.c.n.t.t.NetconfDeviceSimulator - All simulated devices started successfully from port 17
830 to 17830
```

Default parameters

The default parameters for testtool are:

- Use SSH
- Run 1 simulated device
- Device port is 17830
- Yang modules used by device are only: ietf-netconf-monitoring, ietf-yang-types, ietf-inet-types (these modules are required for device in order to support netconf monitoring and are included in the netconf-testtool)
- Connection timeout is set to 30 minutes (quite high, but when testing with 10000 devices it might take some time for all of them to fully establish a connection)
- Debug level is set to false
- No distribution is modified to connect automatically to the netconf testtool

Verifying testtool

To verify that the simulated device is up and running, we can try to connect to it using command line ssh tool. Execute this command to connect to the device:

```
ssh admin@localhost -p 17830 -s netconf
```

Just accept the server with yes (if required) and provide any password (Testtool accepts all users with all passwords). You should see the hello message sent by simulated device.

Testtool help

```
usage: netconf testtool [
  -h] [
  --device-count DEVICES-COUNT] [
  --schemas-dir SCHEMAS-DIR] [
  --notification-file NOTIFICATION-FILE] [
  --starting-port STARTING-PORT]
                          [
  --generate-config-connection-timeout GENERATE-CONFIG-CONNECTION-TIMEOUT] [
  --generate-config-address GENERATE-CONFIG-ADDRESS]
                          [
  --generate-configs-batch-size GENERATE-CONFIGS-BATCH-SIZE] [
  --distribution-folder DISTRO-FOLDER] [
  --ssh SSH] [
  --exi EXI] [
  --debug DEBUG]

Netconf device simulator. Detailed info can be found at https:
  //wiki.opendaylight.org/view/OpenDaylight_Controller:Netconf:Testtool#Building_testtool

optional arguments:
  -h, --help             show this help message and exit
  --device-count DEVICES-COUNT
                         Number of simulated netconf devices to spin
  --schemas-dir SCHEMAS-DIR
                         Directory containing yang schemas to describe simulated devices. Some schemas e.g. netconf
 monitoring and inet types are included by default
  --notification-file NOTIFICATION-FILE
                         Xml file containing notifications that should be sent to clients after create
subscription is called
  --starting-port STARTING-PORT
                         First port for simulated device. Each other device will have previous+
  1 port number
  --generate-config-connection-timeout GENERATE-CONFIG-CONNECTION-TIMEOUT
                         Timeout to be generated in initial config files
  --generate-config-address GENERATE-CONFIG-ADDRESS
                         Address to be placed in generated configs
  --generate-configs-batch-size GENERATE-CONFIGS-BATCH-SIZE
                         Number of connector configs per generated file
  --distribution-folder DISTRO-FOLDER
                         Directory where the karaf distribution for controller is located
  --ssh SSH              Whether to use ssh for transport or just pure tcp
  --exi EXI              Whether to use exi to transport xml content
  --debug DEBUG          Whether to use debug log level instead of INFO
```

Note: When using SSH for netconf, the testtool is quite resource demanding so more memory has to be reserved for the testtool (run testtool with these addition arguments e.g. -Xmx1G -XX:MaxPermSize=256M)
Note2: Default starting port is 17830

Supported operations [edit]

Testtool supported operations:

1. get-schema - returns yang schemas loaded from user specified directory,
2. edit-config - always returns OK and stores the xml from the input in a local variable available for get-config and get rpc. Every edit-config replaces the previous data,
3. commit - always returns OK, but does not actually commit the data,
4. get-config - returns local xml stored by edit-config,
5. get - returns local xml stored by edit-config with netconf-state subtree, but also supports filtering.
6. (un)lock - returns always ok with no lock guarantee
7. create-subscription - returns always ok and after the operation is triggered, provided netconf notifications (if any) are fed to the client. No filtering or stream recognition is supported.

Note: when operation="delete" is present in the payload for edit-config, it will wipe its local store to simulate the removal of data.

Notification support

Testtool supports notifications via the *--notification-file* switch. To trigger the notification feed, create-subscription operation has to be invoked.
The xml file provided should look like this example file:

```
  <?xml version=
  '1.0' encoding=
  'UTF-8' standalone=
  'yes'
  ?>
<notifications>

<!-- Notifications are processed in the order they are defined in XML -->

<!-- Notification that is sent only once right after create-subscription is called -->
<notification>
                <!-- Content of each notification entry must contain the entire notification with event time. Event
 time can be hardcoded, or generated by testtool if XXXX is set as eventtime in this XML -->
                <content><!
  [CDATA[
                        <notification xmlns=
  "urn:ietf:params:xml:ns:netconf:notification:1.0"
  >
                            <eventTime>
  2011
  -01-04T12:
  30
  :
  46
  </eventTime>
                            <random-notification xmlns=
  "http://www.opendaylight.org/netconf/event:1.0"
  >
                                <random-content>single no delay</random-content>
                            </random-notification>
                        </notification>
                ]
  ]
  ></content>
</notification>

<!-- Repeated Notification that is sent 5 times with 2 second delay inbetween -->
<notification>
                <!-- Delay in seconds from previous notification -->
                <delay>
  2
  </delay>
                <!-- Number of times this notification should be repeated -->
                <times>
  5
  </times>
                <content><!
  [CDATA[
                        <notification xmlns=
  "urn:ietf:params:xml:ns:netconf:notification:1.0"
  >
                            <eventTime>XXXX</eventTime>
                            <random-notification xmlns=
  "http://www.opendaylight.org/netconf/event:1.0"
  >
                                <random-content>scheduled 5 times 10 seconds each</random-content>
                            </random-notification>
                        </notification>
                ]
  ]
  ></content>
</notification>
```

```
<!-- Single notification that is sent only once right after the previous notification -->
<notification>
                <delay>
  2
  </delay>
                <content><!
  [CDATA[
                        <notification xmlns=
  "urn:ietf:params:xml:ns:netconf:notification:1.0"
  >
                            <eventTime>XXXX</eventTime>
                            <random-notification xmlns=
  "http://www.opendaylight.org/netconf/event:1.0"
  >
                                <random-content>single with delay</random-content>
                            </random-notification>
                        </notification>
                ]
  ]
  ></content>
</notification>

</notifications>
```

Connecting testtool with controller karaf distribution

This part describes the usage of the testtool with ODL controller. The usage differs slightly between Helium and Lithium.

- Testtool + Helium distribution of controller

Auto connect for controller

It is possible to make the controller distribution auto connect to the simulated devices spawned by testtool (so user does not have to post a configuration for every netconf connector via Restconf). The testtool is able to modify the ODL distribution to auto connect to the simulated devices after feature "odl-netconf-connector-all" is installed.

When running testtool, issue this command(just point the testool to the distribution) :

```
java -Xmx1G -XX:MaxPermSize=256M -jar netconf-testtool-0.3.0-SNAPSHOT-executable.jar --device-count 10 --distribut
ion-folder ~/distribution-karaf-0.2.0-Helium/ --debug true
```

With the distribution-folder parameter, the testtool will modify the distribution to include configuration for netconf connector to connect to all simulated devices. So there is no need to spawn netconf connectors via Restconf.

Running testtool and ODL on different machines

The testtool binds by default to 0.0.0.0 so it should be accessible from remote machines. However you need to set the parameter "generate-config-address" (when using autoconnect) to the address of machine where testtool will be run so ODL can connect. The default value is localhost.

Testtool + Controller base karaf distribution

You can test netconf not only with a downloaded distribution but also with local opendaylight-karaf distribution inside controller project:

- Build the whole controller project with latest code
- Start the testtool with following parameters(assuming running the testtool from controller/opendaylight/netconf/netconf-testtool):

```
java -jar netconf-testtool-0.3.0-SNAPSHOT-executable.jar --device-count 10 --distribution-folder ../../../distribu
tion/opendaylight-karaf/target/assembly/ --debug true
```

Executing operations via Restconf on a mounted simulated device

Simulated devices support basic rpcs for editing their config. This part shows how to edit data for simulated device via Restconf.

Test yang schema

The controller and Restconf assume that the data that can be manipulated for mounted device is described by a yang schema. For demonstration, we will define a simple yang model:

```
module test {
    yang-version 1
  ;
    namespace "urn:opendaylight:test"
  ;
    prefix "tt"
  ;

    revision "2014-10-17"
  ;


   container cont {

        leaf l {
            type string;
        }
   }
}
```

Save this schema in file called **test@2014-10-17.yang** and store it a directory called test-schemas/ in e.g. home folder.

Editing data for simulated device

- Start the device with following command:

```
java -Xmx1G -XX:MaxPermSize=256M -jar netconf-testtool-0.3.0-SNAPSHOT-executable.jar --device-count 10 --distribut
ion-folder ~/distribution-karaf-0.2.0-Helium/ --debug true --schemas-dir ~/test-schemas/
```

- Start helium distribution
- Install odl-netconf-connector-ssh feature
- Install odl-restconf feature\
- Check that you can see config data for simulated device by Executing GET request to

```
http:
  //localhost:8181/restconf/config/opendaylight-inventory:nodes/node/17830-sim-device/yang-ext:mount/
```

- The data should be just and empty data container
- Now execute edit-config request by executing a POST request to:

```
http:
  //localhost:8181/restconf/config/opendaylight-inventory:nodes/node/17830-sim-device/yang-ext:mount
```

with headers:

```
Accept application/xml
Content-Type application/xml
```

and payload:

```
  <cont xmlns=
  "urn:opendaylight:test"
  >
  <l>Content</l>
</cont>
```

- Check that you can see modified config data for simulated device by Executing GET request to

```
http:
  //localhost:8181/restconf/config/opendaylight-inventory:nodes/node/17830-sim-device/yang-ext:mount/
```

- Check that you can see the same modified data in operational for simulated device by Executing GET request to

```
http:
  //localhost:8181/restconf/operational/opendaylight-inventory:nodes/node/17830-sim-device/yang-ext:mount/
```

Known problems

Slow creation of devices on virtual machines

When testtool seems to take unusually long time to create the devices use this flag when running it:

```
  -Dorg.apache.sshd.registerBouncyCastle
  =
  false
```

Too many files open

When testtool or ODL starts to fail with TooManyFilesOpen exception, you need to increase the limit of open files in your OS. To find out the limit in linux execute:

```
ulimit -a
```

Example sufficient configuration in linux:

```
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (
  -e) 0
file size               (blocks, -f) unlimited
pending signals                 (
  -i) 63338
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files                      (
  -n) 500000
pipe size            (
  512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (
  -r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (
  -u) 63338
virtual memory          (kbytes, -v) unlimited
file locks                      (
  -x) unlimited
```

To set these limits edit file: /etc/security/limits.conf e.g:

```
  *         hard    nofile      500000
*         soft    nofile      500000
root      hard    nofile      500000
root      soft    nofile      500000
```

"Namespace urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf is not owned by a module"

error parsing input: Namespace urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf is not owned by a module The netconf-node-topology model could be missing. Fix by restarting ODL

"Killed"

The testtool might end unexpectedly with a simple message: "Killed". This means that the OS killed the tool due to too much memory consumed or too many threads spawned. To find out the reason on linux you can use following command:

```
dmesg | egrep -i -B100 'killed process'
```

Also take a look at this file: /proc/sys/kernel/threads-max. It limits the number of threads spawned by a process. Sufficient(but probably much more than enough) value is e.g. 126676