pyats-03 - Connecting to a device

Syllabus

- installation
- understand topology
- understand testbed file
- `unicon` module

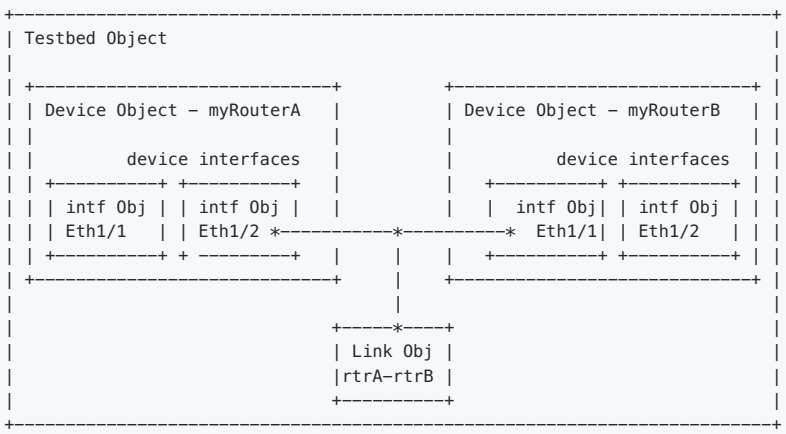
Presentation

Installation

```
pip install pyats
# See:
#   pyats.connections
#   pyats.topology
#   unicon
```

Understand topology

Objects' concept



Classes

```
from pyats.topology import Testbed, Device, Interface, Link
help(Testbed)
help(Device)
help(Interface)
help(Link)
```

Sample topology

```
from pyats import topology


testbed = topology.loader.load('''
devices:
    vm:
      os: 'linux'
      tacacs:
        username: pyast
      passwords:
        linux: pyastpyast
      connections:
        linux:
          protocol: ssh
          ip: 192.168.242.44
      type: 'linux'
''')

device = testbed.devices['vm']

device.connect()
assert device.connected

output = device.execute('hostname')
print(output)

device.disconnect()
```

Understand testbed file

A YAML testbed file

```
# test-env.yaml
devices:
    vm:
      os: 'linux'
      type: "%{self.os}"
      connections:
        defaults:
          via: main
          alias: default
        main:
          username: pyast
          protocol: ssh
          ip: 192.168.242.44
          password: pyastpyast
        test:
          username: pyast
          protocol: ssh
          ip: 192.168.242.44
          password: pyastpyast
```

> See full list of configuration options on https://pubhub.devnetcloud.com/media/pyats/docs/topology/schema.html.

Testbed validation

```
pyats --help
pyats validate testbed test-env.yaml
```

> Read more on https://pubhub.devnetcloud.com/media/pyats/docs/cli/pyats.html#

A test

```
# connections.py
from pyats.topology import loader, Device
from pyats.aetest import Testcase, test,  main


class Smoke(Testcase):

    @test
    def defaut_connection(self, device: Device):
        device.connect() # == self.vm.connectionmgr.connect()
        device.execute('hostname')
        device.disconnect()

    @test
    def test_connection(self, device: Device):
        device.connect(via='test', alias='test')
        device.test.execute('hostname')
        device.test.disconnect()

    @test
    def multiple_connections(self, device: Device):
        device.connect(via='main', alias='main')
        device.main.ping('108.177.119.103')

        device.connect(via='test', alias='test')
        device.test.execute('hostname')

        device.disconnect_all()

if __name__ == '__main__':
    testbed = loader.load('test-env.yaml')
    main(device=testbed.devices['vm'])


# job.py
from pyats.easypy.tasks import run


def main(runtime):
    run(testscript='connections.py', device=runtime.testbed.devices['vm'])
```

Execution

```
python connections.py
easypy job.py -testbed_file test-env.yaml
```

Custom connections

```
# connections creation
from pyats.connections import BaseConnection
help(BaseConnection)
help(BaseConnection.locked)

# hooks creation
from pyats.connections.hooks import ServiceHook
help(ServiceHook)
```

`unicon` module

`unicon`'s connection is a default one for the `topology` module.

Sample connection

```
from unicon import Connection
help(Connection)
vm = Connection(hostname='vm424583', start=['ssh 192.168.242.44'],
                username='pyast', password='pyastpyast', os='linux')
vm.connect()
vm.connected
version = vm.execute('cat /etc/issue')
version
vm.disconnect()
```

Main features:

- agnostic

- support CLI proxies
- support SSH tunnels
- interactive commands via Expect Abstraction Library (EAL)

Overview of interactive commands

```
# initiate interactive session
from unicon.eal.expect import Spawn
help(Spawn)
# Spawn("telnet 1.2.3.4 1000")


from unicon.eal.dialogs import Statement
help(Statement)
# Statement(pattern=r'password:',
#               action=lambda spawn, password: spawn.sendline(password)
#               args={'password': 10},
#               loop_continue=True,
#               continue_timer=False)

from unicon.eal.dialogs import  Dialog
help(Dialog)
# Dialog(Statement(...), Statement(...), Statement(...)).process(Spawn(....))
```

Additional materials

- http://yaml.org/
- https://pubhub.devnetcloud.com/media/pyats/docs/topology/introduction.html
- https://pubhub.devnetcloud.com/media/pyats/docs/topology/concept.html
- https://pubhub.devnetcloud.com/media/pyats/docs/topology/schema.html
- https://pubhub.devnetcloud.com/media/pyats/docs/connections/index.html
- https://pubhub.devnetcloud.com/media/pyats-packages/docs/unicon/user_guide/introduction.html
- https://pubhub.devnetcloud.com/media/pyats-packages/docs/unicon/user_guide/connection.html
- https://pubhub.devnetcloud.com/media/pyats-packages/docs/unicon/user_guide/eal.html
- https://pubhub.devnetcloud.com/media/pyats/docs/utilities/file_transfer_utilities.html

Control

Task description

There is a linux VM (`192.168.242.44` , user: `pyast` , pass: `pyastpyast`) which is available only from Softserve's network. You need to create the following tests using given VM:

- copy file from the local PC into the VM
- copy file from the VM to the local PC
  You have to create testbed file which will describe your device. The tests have to be executed with `easypy` .

Please use standard multiprotocol file transfer while implement the task.

Review items

Please send for review:

- testbed file, Python's modules
- installation instructions
- execution instructions