Configuring NETCONF

NETCONF is a network configuration and management protocol based on XML.

NETCONF protocol uses XML for configuration data and protocol message encoding, using RPC and Client/Server mechanism to update, install or delete the relevant part of the device configuration or all the management information.

Currently, we support <get>, <get-config>, <get-schema> and <edit-config>.

Enable NETCONF on switch:

```
admin@XorPlus# set protocols netconf
admin@XorPlus# commit
Commit OK.
Save done.
admin@XorPlus#
```

Delete NETCONF configuration on switch:

```
admin@XorPlus# delete protocols netconf
Deleting:
    netconf {
    }
OK
admin@XorPlus# commit
Commit OK.
Save done.
```

YANG is a data modeling language used to model configuration and state data manipulated by NETCONF.
You can find the YANG module file of different modules on your switch under the directory "/pica/bin/netconf/data-models".

```
admin@XorPlus$pwd
/pica/bin/netconf/data-models
admin@XorPlus$
admin@XorPlus$ls -lt *.yang
-rw-r--r-- 1 root xorp  2075 Jul 13  2016 arp.yang
-rw-r--r-- 1 root xorp  1143 Jul 13  2016 bfd.yang
-rw-r--r-- 1 root xorp  7767 Jul 13  2016 bgp.yang
-rw-r--r-- 1 root xorp  5612 Jul 13  2016 cos.yang
-rw-r--r-- 1 root xorp  2160 Jul 13  2016 dhcp.yang
-rw-r--r-- 1 root xorp  3030 Jul 13  2016 dot1x.yang
-rw-r--r-- 1 root xorp 13460 Jul 13  2016 firewall.yang
-rw-r--r-- 1 root xorp 16760 Jul 13  2016 ietf-inet-types.yang
-rw-r--r-- 1 root xorp 18034 Jul 13  2016 ietf-yang-types.yang
-rw-r--r-- 1 root xorp  1529 Jul 13  2016 igmp.yang
-rw-r--r-- 1 root xorp  2657 Jul 13  2016 igmpsnooping.yang
-rw-r--r-- 1 root xorp 41585 Jul 13  2016 interface.yang
-rw-r--r-- 1 root xorp  4991 Jul 13  2016 ipfix.yang
-rw-r--r-- 1 root xorp  1147 Jul 13  2016 lacp.yang
-rw-r--r-- 1 root xorp  3432 Jul 13  2016 lldp.yang
-rw-r--r-- 1 root xorp   781 Jul 13  2016 mfea.yang
-rw-r--r-- 1 root xorp 18188 Jul 13  2016 mstp.yang
-rw-r--r-- 1 root xorp  4673 Jul 13  2016 neighbour.yang
-rw-r--r-- 1 root xorp  7835 Jul 13  2016 ospfv2.yang
-rw-r--r-- 1 root xorp  5512 Jul 13  2016 ospfv3.yang
-rw-r--r-- 1 root xorp  3408 Jul 13  2016 ovsdb.yang
-rw-r--r-- 1 root xorp  4202 Jul 13  2016 pim.yang
-rw-r--r-- 1 root xorp  8203 Jul 13  2016 policy.yang
-rw-r--r-- 1 root xorp  3954 Jul 13  2016 rip.yang
-rw-r--r-- 1 root xorp  3031 Jul 13  2016 ripng.yang
-rw-r--r-- 1 root xorp  4624 Jul 13  2016 sflow.yang
-rw-r--r-- 1 root xorp  1207 Jul 13  2016 snmp.yang
-rw-r--r-- 1 root xorp  2666 Jul 13  2016 static-routes.yang
-rw-r--r-- 1 root xorp  1670 Jul 13  2016 stm.yang
-rw-r--r-- 1 root xorp  1666 Jul 13  2016 udld.yang
-rw-r--r-- 1 root xorp  2747 Jul 13  2016 vlan-interface.yang
-rw-r--r-- 1 root xorp  6063 Jul 13  2016 vlans.yang
-rw-r--r-- 1 root xorp  1747 Jul 13  2016 vrrp.yang
-rw-r--r-- 1 root xorp  6423 Jul 13  2016 vxlans.yang
-rw-r--r-- 1 root xorp  4186 Jul 13  2016 xovs.yang
```

Example of VLAN configuration via NETCONF use <edit-config>:

Step 1: Create an XML file according the vlan.yang for RPC request to create VLAN136:

```
<vlans xmlns="http://pica8.com/xorplus/vlans">
    <vlan-id>
      <id>136</id>
      <description/>
      <vlan-name>default</vlan-name>
      <l3-interface>vlan136</l3-interface>
    </vlan-id>
  </vlans>
```

Step 2: Display the configuration on switch after the client sending an RPC request.

```
The configuration has been changed by user root
DELTAS:
    vlans {
        vlan-id 136 {
            description: ""
            vlan-name: "default"
            l3-interface: "vlan136"
        }
    }
admin@XorPlus# show | display set
    set protocols netconf
    set vlans vlan-id 136 l3-interface "vlan136"
```

Now, we only support get the system's version information and vxlan information via NETCONF <get> function.

Example of get the system's version information via NETCONF use <get>:

Display the RPC reply after the client sending an RPC request.

```
    <version xmlns="http://pica8.com/xorpplus/version">
      <mac_address>48:0f:cf:af:70:3b</mac_address>
      <hardware_mode>HP5712</hardware_mode>
      <system_version>2.8.0/aeec598</system_version>
      <system_released_date>10/13/2016</system_released_date>
      <L2_L3_version>2.8.0/aeec598</L2_L3_version>
      <L2_L3_released_date>10/13/2016</L2_L3_released_date>
    </version>
```

NETCONF client

About NETCONF client, you can use ncclient which is python lib now.

If you use ncclient, you must modify the rpc.py : add two lines codes to work with pica8 switch.

Edit the `rpc.py` file to contain the followings before the statement ‘self._session.send(req)’:

req = req.replace('nc:','')

req = req.replace(':nc','')

Get .yang or .yin File

The administrator can use get-schema operation to retrieve the **.yang** or **.yin** data file information on the PICA8 switch. For details about get-schema operation, see RFC6022 YANG Module for NETCONF Monitoring.

In the following example, the user builds the **testgetschema.py** script on ncclient. The script uses the get-schema operation to get the information from the **vlans.yang** file on the PICA8 switch.

```
[ncclient] $ vi  testgetschema.py
from ncclient import manager
import sys

host=sys.argv[1]
mgr = manager.connect(host=host, port=830, username='admin', password='pica8', hostkey_verify=False)

elem = mgr.get_schema(identifier='vlans')
with open("%s.xml" % host, 'w') as f:
    f.write(str(elem))
mgr.close_session()
```

Run the **testgetschema.py** script on ncclient. By issuing the get-schema command and receiving the reply from the PICA8 switch, we can get the **vlans.yang** module file information displayed as follows:

```
module vlans {
    namespace "http://pica8.com/xorplus/vlans";
    prefix vlans;
    // import some basic types
    import ietf-yang-types {
        prefix yang;
    }
    organization "PICA8, Inc";
    description
        "This module is data model for vlans configuration";
    revision 2015-12-25 {
        description "Initial revision.";
    }
    container vlans {
        description
            "Vlan configuration.";
        list vlan-id {
            description
                "VLAN tag identifier, range 1-4094, e.g. 2,3,5-100.";
            key "id";
            leaf id {
                type string;
            }
            leaf description {
                description
                    "Vlan description.";
                type string;
                default "";
            }
            leaf vlan-name {
                description
                    "VLAN name, up to 32 alphanumeric characters in length.";
                type string;
                default "default";
            }
            leaf l3-interface {
                description
                    "Associate a Layer 3 interface with an existing VLAN.";
                type string;
                default "";
            }
            leaf open-flow-enable {
                description
                    "Vlan will be used by open flow, maximum of 200 vlans enabled.";
                type boolean;
                default 'false';
            }
        }
........
  }
}
```