NDM Negotiations

# Basic Flow of Operations

1. OFCP connects to capable switch and learns the following through NETCONF capabilities exchange:
   1. If the capable implements NDMs through the optional presence of the NDN YANG module identifiers (URIs)
   2. A list of NDMs that the capable switch implements through their respective unique identifier (URIs)
2. Very simple clients (e.g. shell scripts) may ignore the information provided in the capabilities exchange and perform a query for the subtree that would contain the basic NDM parameters performing a 'get' on the following node:  
     
   /capable-switch/resources/ndm  
     
   If that subtree does not exist, the client can safely assume that the capable switch does not implement NDMs.
3. If the capable switch does indeed implement NDMs as discovered in (1), the client can proceed by querying the capable switch for available NDMs performing a 'get' operation on the following node:  
     
   /capable-switch/resources/ndm/available-ndms  
     
   The capable switch responds with a list of available NDMs as a list with the following leafs:  
     
   ndm:authority string  
   ndm:name string  
   ndm:type enumeration (ttp, fpmod)  
   ndm:version string  
     
   The capable switch may respond with an empty list meaning that while it does implement NDM functionality there are currently no NDMs available for use.
4. At this point, the client may have found NDMs for which it doesn't currently have access to the associated YANG module.  
     
   The client may then use the (optional) *get-schema* operation as described in RFC 6022, Section 3.1 to fetch the YANG modules for specific NDMs.  
     
   The client now knows:
   1. Which NDMs the capable switch supports through the capabilities exchange in (1. above) or the query in (2. above).
   2. The formal set of configuration parameters (including default values) and negotiation RPCs for all NDMs as defined in the corresponding YANG modules.
5. The client may now proceed by instantiating (parameterizing) an NDM on any of the available logical switches either according to the data definitions in the YANG module (5.a below); or by first collecting suggested parameters from the capable switch using the suggest-ndm-parameters RPC if present (5.b below)  
   1. The client uses basic NETCONF *edit-config* operations to create an instantiated (parameterized) NDM on any of the logical switches
   2. The client uses the *suggest-ndm-parameters* RPC with NDM-specific input values to query the switch for suggested parameters. The switch responds with a complete and valid set of data for the client to use in creating an NDM. The OFCP proceeds as described in (5.1)   
        
      The *suggest-ndm-parameters* RPC may be called for several NDMs to collect suggested parameters before deciding on which NDM to create as described in (6. above)

# Example

This section contains an informal set of terminal and protocols dumps corresponding to the general steps above to provide an example. We make use of the *netconf-console* tool to query the NETCOFN server.

The command below initiates a capabilities-exchange with the server to find out which protocol capabilities the server supports and which YANG modules are implemented. We can see below that the server supports of-config, ndm, and two specific ndm-modules l2l3 and l2l3acl.

calle@macbook:openflow $netconf-console --hello

<?xml version="1.0" encoding="UTF-8"?>

<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">

<capabilities>

<capability>urn:ietf:params:netconf:base:1.0</capability>

<capability>urn:ietf:params:netconf:base:1.1</capability>

<capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>

<capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>

[...]

<capability>urn:onf:of111:config:yang?module=of-config&amp;revision=2011-12-07</capability>  
<capability>urn:opennetworking.org:yang:ndm?module=ndm&amp;revision=2013-04-30</capability>

<capability>urn:opennetworking.org:yang:ndm:l2l3?module=l2l3</capability>

<capability>urn:opennetworking.org:yang:ndm:l2l3acl?module=l2l3acl</capability>

</capabilities>

<session-id>13</session-id>

</hello>

The command below queries the server for configuration and operational data under the /capable-switch/resources/ndm node. The server returns data (in this case the name, type and version of the two supported NDMs). The client can now safely assume that the server supports NDMs.

calle@macbook:openflow $netconf-console --get -x /capable-switch/resources/ndm

<?xml version="1.0" encoding="UTF-8"?>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">

<data>

<capable-switch xmlns="urn:onf:of111:config:yang">

<resources>

<ndm xmlns="urn:opennetworking.org:yang:ndm">

<available-ndms>

<name>l2l3acl</name>

<type>ttp</type>

<version>draft</version>

</available-ndms>

<available-ndms>

<name>l2l3</name>

<type>ttp</type>

<version>draft</version>

</available-ndms>

</ndm>

</resources>

</capable-switch>

</data>

</rpc-reply>

calle@macbook:openflow $

This is an example where the server has no support for NDMs:

calle@macbook:openflow $netconf-console --get -x /capable-switch/resources/xlndm

<?xml version="1.0" encoding="UTF-8"?>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">

<data/>

</rpc-reply>

The below shows how to fetch a specific YANG module inband using the get-schema and the name of the module we want to fetch (l2l3acl). Module content edited for space.

calle@macbook:openflow $netconf-console --get-schema=l2l3acl

<?xml version="1.0" encoding="UTF-8"?>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">

<data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring"><![CDATA[module l2l3acl {

namespace "urn:opennetworking.org:yang:ndm:l2l3acl";

prefix "l2l3acl";

import of-config {

prefix ofc;

}

import ndm {

prefix ndm;

}

grouping l2l3acl-params {

leaf ingress-vlan-table-size {

type uint32;

}

[...]

}]]></data>

</rpc-reply>

calle@macbook:openflow $

The below shows how to add a parameterized L2L3 NDM to the logical switch. First we’ll take a look at the content of the NDM we’re planning to add:

calle@macbook:openflow $more add-l2l3acl-implementation.xml

<capable-switch xmlns="urn:onf:of111:config:yang"

xmlns:ndm="urn:opennetworking.org:yang:ndm"

xmlns:l2l3acl="urn:opennetworking.org:yang:ndm:l2l3acl">

<logical-switches>

<switch>

<id>LogicalSwitch6</id>

<resources>

<ndm:parameterized-ndm>

<l2l3acl:l2l3acl>

<l2l3acl:ingress-vlan-table-size>128</l2l3acl:ingress-vlan-table-size>

<l2l3acl:acl-table-size>128</l2l3acl:acl-table-size>

<l2l3acl:router-mac-table-size>128</l2l3acl:router-mac-table-size>

<l2l3acl:l3-table-size>128</l2l3acl:l3-table-size>

<l2l3acl:l2-table-size>128</l2l3acl:l2-table-size>

<l2l3acl:egress-vlan-table-size>128</l2l3acl:egress-vlan-table-size>

</l2l3acl:l2l3acl>

</ndm:parameterized-ndm>

</resources>

</switch>

</logical-switches>

</capable-switch>

The next step is to add it using the edit-config operation:

calle@macbook:openflow $netconf-console --edit-config=add-l2l3acl-implementation.xml

<?xml version="1.0" encoding="UTF-8"?>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">

<ok/>

</rpc-reply>

And the next steps is to take a look at the newly parametrized NDM in the configuration using the get-config operation:

calle@macbook:openflow $netconf-console --get -x "/capable-switch/logical-switches/switch[id='LogicalSwitch6']/resources/parameterized-ndm"

<?xml version="1.0" encoding="UTF-8"?>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">

<data>

<capable-switch xmlns="urn:onf:of111:config:yang">

<logical-switches>

<switch>

<id>LogicalSwitch6</id>

<resources>

<parameterized-ndm xmlns="urn:opennetworking.org:yang:ndm">

<l2l3acl xmlns="urn:opennetworking.org:yang:ndm:l2l3acl">

<ingress-vlan-table-size>128</ingress-vlan-table-size>

<acl-table-size>128</acl-table-size>

<router-mac-table-size>128</router-mac-table-size>

<l3-table-size>128</l3-table-size>

<l2-table-size>128</l2-table-size>

<egress-vlan-table-size>128</egress-vlan-table-size>

</l2l3acl>

</parameterized-ndm>

</resources>

</switch>

</logical-switches>

</capable-switch>

</data>

</rpc-reply>

calle@macbook:openflow $

The below shows how to use the suggest-ndm-parameters to ask the switch for a reasonable set of parameters to use. First we take a look at the input parameters suggested by the client:

calle@macbook:openflow $more action.xml

<suggest-ndm-parameters xmlns="urn:opennetworking.org:yang:ndm">

<l2l3acl xmlns="urn:opennetworking.org:yang:ndm:l2l3acl">

<ingress-vlan-table-size>128</ingress-vlan-table-size>

<router-mac-table-size>128</router-mac-table-size>

</l2l3acl>

</suggest-ndm-parameters>

Then we execute the RPC and capture the output. The output parameters (in the rpc-reply below) can be immediately used in a edit-config to add a parametrized NDM according to server suggestions.

calle@macbook:openflow $netconf-console --rpc=action.xml

<?xml version="1.0" encoding="UTF-8"?>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">

<l2l3acl xmlns="urn:opennetworking.org:yang:ndm:l2l3acl">

<ingress-vlan-table-size>128</ingress-vlan-table-size>

<acl-table-size>128</acl-table-size>

<router-mac-table-size>128</router-mac-table-size>

<l3-table-size>128</l3-table-size>

<l2-table-size>128</l2-table-size>

<egress-vlan-table-size>128</egress-vlan-table-size>

</l2l3acl>

</rpc-reply>

calle@macbook:openflow $