NED Migration in NSO 5

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

CHAPTER 1

Overview

Summary

In NSO 5 changing the ned-id for a device that has configuration stored in CDB is NOT allowed. The reason for this is that changing the ned-id is an implicit data migration.

Instead the new procedure is to load the new NED and then run the new **migrate** action. This action will migrate data and change the ned-id.

History

Prior to NSO 5 there could only be one instance of a certain namespace stored at any time. This prevented any procedure or management that required several backward incompatible versions of a namespace to be accessible. For devices managed by NSO this was a problem since devices of a certain family needed to share one and the same NED. So if one device added or firmware upgraded required a new NED version with backward incompatible namespaces then all devices using that NED needed to upgrade its stored configuration to that NED version.

The work associated with changing a NED version for device would also range from very easy to very hard. For instance if some external upgrade code or manual intervention was needed. If services where affected by the changes and specific steps like un-deploys and/or re-deploys of services had to be explored and executed on all services.

A second problem correlated to the NED version upgrades was that the pre-upgrade analysis was at best a manual procedure. Also this is important to do since without it and the above preparations the upgrade could in worst case end with data loss and inconsistent services.

With the introduction of NSO 5 this has all changed. There are defined processes and tools for changing a device NED and identify/manage the affected services. Before we go into these details we need to step back and discuss how NSO 5 can handle several backward incompatible versions of the same namespace.

CHAPTER 2

Introduction

With the introduction of NSO 5 the administration of NEDs and the loading of NED YANG modules has changed. To separate NED versions with backward incompatible YANG modules these NEDs need to be compiled with a unique identified that separates the versions at load time in NSO. This unique identifier is the "ned-id". The device YANG module compilation is performed with a new ncsc compiler directive --ncs-ned-id.

In releases prior to NSO 5 the ned-id was only used to identify the single NED that handled a group of devices of the same type. It was a sort of "family" identifier that had the form like "cisco-ios" or "alu-sr". With NSO 5 a ned-id needs to include both the type (netconf/cli/gen/snmp) and major version. So the look like "cisco-ios-cli-11.1" and "alu-sr-cli-12.3". As before these ned-ids are YANG identities. The family identities still exists are are defined as abstract YANG identities that are base identity for all ned-id versions for a specific type of devices. This way queries of a system of which NED versions exist for a certain family can be made.

With NSO 5 every NED now needs a ned-id. This also include netconf NEDS, which is a big difference from earlier NSO releases where the default "netconf" ned-id could be used for all netconf NEDs. NSO 5 requires also the netconf NEDS to have unique ned-ids in the form like "juniper-junos-nc-13.1" and be based on a family id like "juniper-junos".

Note Before NSO 5 the NED packages all resided in a directory named as old ned-id. The new ned-id has changed and so has the directory name. If the NED tarball is unpacked the unique and qualified name needs to be there or there would be problems adding new NED versions.

It is safe to say that the importance of the ned-id has grown with NSO 5. Internally NSO will, for a device path, lookup the ned-id to find the unique YANG modules for that device. Also, as soon as any configuration is stored for the device, then the ned-id cannot be changed or deleted. This would otherwise imply a data loss. Changing the ned-id is equivalent to a device configuration data migration or "NED migration". To be able to perform a NED migration a new action has been implemented called "migrate". This is the only way a ned-id can be changed for a device which already has stored configuration data.

CHAPTER 3

Migration

The "migrate" action can migrate device configuration between backwards incompatible NED versions. The action are designed to give structured analysis of which paths will change between two NED versions and visibility into the scope of the potential impact that a change in the NED will drive in the service code.

The action allows for a usage-based analysis of which parts of the NED data model (and instance tree) that a particular service has written to. This will give the user an (at least opportunistic) sense of which paths must changed in service code.

These features aims to significantly reduce the amount of uncertainty and side-effects that NED-upgrades comes with historically and lower the barrier of upgrading NEDs.

By using the `/ncs:devices/device/migrate` action one can change the NED version of a device. The action migrates all configuration and service meta-data. As a side-effect it will read and commit the actual device configuration. The action can also be executed in parallel on a device group or on all devices matching a NED identity.

The action reports what paths have been modified and the services affected by those changes. This information can then be used to prepare the service code to handle the new NED version. If the `verbose` option is used, all service instances are reported instead of just the service points. If the `dry-run` option is used, the action simply reports what it would do. This gives you the chance to do analysis before any actual change is performed.

If the `no-networking` option is used, no southbound traffic is generated towards the devices. Only the device configuration in CDB is used for the migration. If used, NSO can not know if the device is in sync. To determine this, the **compare-config** or the **sync-from** action must be used. The `no-networking` option will give you an opportunity to perform an migration and verify that the changes work in a local environment without the need for actual devices.

It is recommended to **re-deploy** all affected services touching the device after the device migration, even though there are no backwards incompatible data model changes affecting the service. When reading the reverse/forward diffset of a service, NSO will detect changes to the NED identity of a device touched by the service and migrate the diffset on the fly. Thus the diffsets are still valid, but until the new diffset is written (typically through a re-deploy) this migration procedure will add extra time in handling the reverse/forward diffset (e.g. when using the **get-modifications** action).

CHAPTER 4

Migration Example

The example `examples.ncs/getting-started/developing-with-ncs/26-ned-migration` in the NSO examples collection illustrates how to migrate devices between different NED versions using the **migrate** action. This chapter illustrates how to use the **migrate** in this example.

The example starts of with having two NEDS, router-nc-1.0 and router-nc-1.1 with the following incompatible change:

```
$ diff packages/router-nc-1.0/src/yang packages/router-nc-1.1/src/yang
78a79,89
>   typedef syslogOption {
>     type enumeration {
>       enum "cons";
>       enum "ndelay";
>       enum "nowait";
>       enum "odelay";
>       enum "perror";
>       enum "pid";
>     }
>   }
>
106,109c117,118
<         container option {
<           leaf pid {
<             type empty;
<           }
---
>         leaf-list option {
>           type syslogOption;
```

To build the example, do:

```
$ make all
```

To start the ncs-netsim network, do:

```
$ ncs-netsim start
DEVICE ex0 OK STARTED
DEVICE ex1 OK STARTED
```

To run the example we do:

```
$ ncs
```

This will start NSO and load the packages.

We now start a CLI to begin our work:

```
$ ncs_cli -u admin
```

We need an initial sync-from devices:

```
admin@ncs> request devices sync-from
sync-result {
    device ex0
    result true
}
sync-result {
    device ex1
    result true
}
```

We create a service instance touching configuration on the devices that we are about to migrate:

```
admin@ncs> configure
admin@ncs% set services syslog devices [ ex0 ex1 ]
admin@ncs% set services syslog server 1.2.3.4 enabled selector 1 facility [ all ] level info
admin@ncs% set services syslog server 1.2.3.4 enabled selector 1 comparison same
admin@ncs% commit
```

View the service meta-data:

```
admin@ncs% show devices device ex0..1 config r:sys syslog server | display xml
<config xmlns="http://tail-f.com/ns/config/1.0">
  <devices xmlns="http://tail-f.com/ns/ncs">
  <device>
    <name>ex0</name>
      <config>
      <sys xmlns="http://example.com/router">
      <syslog>
      <server refcounter="1"  backpointer="[ /ncs:services/sls:syslog ]" >
        <name>1.2.3.4</name>
        <enabled refcounter="1" >true</enabled>
        <selector refcounter="1"  backpointer="[ /ncs:services/sls:syslog ]" >
          <name>1</name>
          <comparison refcounter="1" >same</comparison>
          <level refcounter="1" >info</level>
          <option>
            <pid refcounter="1"  backpointer="[ /ncs:services/sls:syslog ]" />
          </option>
          <facility refcounter="1"  backpointer="[ /ncs:services/sls:syslog ]" >all</facility>
        </selector>
      </server>
      <server>
        <name>10.3.4.5</name>
        <enabled>true</enabled>
        <selector>
          <name>8</name>
          <facility>auth</facility>
          <facility>authpriv</facility>
          <facility>local0</facility>
        </selector>
      </server>
      </syslog>
      </sys>
      </config>
  </device>
  <device>
    <name>ex1</name>
      <config>
      <sys xmlns="http://example.com/router">
```

```
      <syslog>
      <server refcounter="1"  backpointer="[ /ncs:services/sls:syslog ]" >
        <name>1.2.3.4</name>
        <enabled refcounter="1" >true</enabled>
        <selector refcounter="1"  backpointer="[ /ncs:services/sls:syslog ]" >
          <name>1</name>
          <comparison refcounter="1" >same</comparison>
          <level refcounter="1" >info</level>
          <option>
            <pid refcounter="1"  backpointer="[ /ncs:services/sls:syslog ]" />
          </option>
          <facility refcounter="1"  backpointer="[ /ncs:services/sls:syslog ]" >all</facility
        </selector>
      </server>
      <server>
        <name>10.3.4.5</name>
        <enabled>true</enabled>
        <selector>
          <name>8</name>
          <facility>auth</facility>
          <facility>authpriv</facility>
          <facility>local0</facility>
        </selector>
      </server>
      </syslog>
      </sys>
      </config>
  </device>
  </devices>
</config>
```

To illustrate the migration to a different device model we need to make some changes to our simulated device. In another shell, copy the device model from the package we are about to migrate to:

```
$ cp packages/router-nc-1.1/netsim/router.fxs netsim/ex/ex0/router.fxs

$ ncs-netsim stop ex0
DEVICE ex0 STOPPED
$ ncs-netsim start ex0
DEVICE ex0 OK STARTED

$ ncs-netsim cli ex0

admin@ex0> configure
admin@ex0% set sys syslog server 1.2.3.4 selector 1 option [ pid ]
admin@ex0% commit
admin@ex0% exit
admin@ex0> exit
```

The migrate action reports what paths have been modified and the services affected by those changes. The verbose option makes the action report all service instances instead of just the service points. If the dry-run option is used, the action simply reports what it would do. If the no-networking option is used, no southbound traffic is generated towards the devices. Only the device configuration in CDB is used for the migration.

We can now do a migrate dry-run to check what would happen if we migrated the device to the new NED:

```
admin@ncs% request devices device ex0 migrate new-ned-id router-nc-1.1 dry-run verbose
modified-path {
    path /devices/device[name='ex0']/config/r:sys/syslog/server/selector/option/pid
    info sub-tree has been deleted
}
```

```
modified-path {
    path /devices/device[name='ex0']/config/r:sys/syslog/server/selector/option
    info node type has changed from non-presence container to leaf-list
}
affected-services-with-changes [ /services/sls:syslog ]
affected-services [ /services/sls:syslog ]
affected-services-with-changes [ /services/sls:syslog ]
affected-services [ /services/sls:syslog ]
```

This tells us that there pid leaf has changed and that changes on the device will affect us in the /services/sls:syslog service.

So we need to fix the service code so that it can handle the change in the NED model. This service is implemented using a template. If we list the template we will see that the template code already handles the two incompatible versions of the NED. So we are in a good spot to run the migration.

```
$ cat packages/syslog/templates/syslog.xml
```

```
      ...
      <config>
        <?if-ned-id
          router-nc-1.0:router-nc-1.0?>
          <sys xmlns="http://example.com/router">
            <syslog>
              <server foreach="{/server}">
                ...
                <selector foreach="{selector}">
                  ...
                  <option>
                    <pid/>
                  </option>
                  ...
                </selector>
              </server>
            </syslog>
          </sys>
        <?elif-ned-id
          router-nc-1.1:router-nc-1.1?>
          <sys xmlns="http://example.com/router">
            <syslog>
              <server foreach="{/server}">
                ...
                <selector foreach="{selector}">
                  ...
                  <option>pid</option>
                  ...
                </selector>
              </server>
            </syslog>
          </sys>
        <?end?>
      </config>
      ...
```

In the NSO CLI it is time to migrate the device:

```
admin@ncs% request devices device ex0 migrate new-ned-id router-nc-1.1 verbose
modified-path {
    path /devices/device[name='ex0']/config/r:sys/syslog/server/selector/option/pid
    info sub-tree has been deleted
}
modified-path {
    path /devices/device[name='ex0']/config/r:sys/syslog/server/selector/option
```

```
    info node type has changed from non-presence container to leaf-list
}
affected-services-with-changes [ /services/sls:syslog ]
affected-services [ /services/sls:syslog ]
```

We will now view how the service data has been affected on the device:

```
admin@ncs% show devices device ex0 config r:sys syslog server | display xml
<config xmlns="http://tail-f.com/ns/config/1.0">
  <devices xmlns="http://tail-f.com/ns/ncs">
  <device>
    <name>ex0</name>
      <config>
      <sys xmlns="http://example.com/router">
      <syslog>
      <server refcounter="1"  backpointer="[ /ncs:services/sls:syslog ]" >
        <name>1.2.3.4</name>
        <enabled refcounter="1" >true</enabled>
        <selector refcounter="1"  backpointer="[ /ncs:services/sls:syslog ]" >
          <name>1</name>
          <comparison refcounter="1" >same</comparison>
          <level refcounter="1" >info</level>
          <option>pid</option>
          <facility refcounter="1"  backpointer="[ /ncs:services/sls:syslog ]" >all</facility
        </selector>
      </server>
      <server>
        <name>10.3.4.5</name>
        <enabled>true</enabled>
        <selector>
          <name>8</name>
          <facility>auth</facility>
          <facility>authpriv</facility>
          <facility>local0</facility>
        </selector>
      </server>
      </syslog>
      </sys>
      </config>
  </device>
  </devices>
</config>

admin@ncs% request services syslog get-modifications
cli {
    local-node {
        data  devices {
                    device ex0 {
                        config {
                            r:sys {
                                syslog {
              +                     server 1.2.3.4 {
              +                         enabled;
              +                         selector 1 {
              +                             comparison same;
              +                             level info;
              +                             facility [ all ];
              +                         }
              +                     }
                                }
                            }
                        }
                    }
```

```
                    device ex1 {
                        config {
                            r:sys {
                                syslog {
                +                   server 1.2.3.4 {
                +                       enabled;
                +                       selector 1 {
                +                           comparison same;
                +                           level info;
                +                           option {
                +                               pid;
                +                           }
                +                           facility [ all ];
                +                       }
                +                   }
                                }
                            }
                        }
                    }
                }

    }
}
```

As you could see the service still works but does not longer own the data with the backwards incompatible change. That data now looks like it has been manually configured below the service data. Since this is not desired we use the action:

```
admin@ncs% request services syslog re-deploy reconcile { discard-non-service-config }
```

Now the service once again owns the data affected by the NED migration. Let's view the service data on the device once again:

```
admin@ncs% show devices device ex0 config r:sys syslog server | display xml
<config xmlns="http://tail-f.com/ns/config/1.0">
  <devices xmlns="http://tail-f.com/ns/ncs">
  <device>
    <name>ex0</name>
      <config>
      <sys xmlns="http://example.com/router">
      <syslog>
      <server refcounter="1"  backpointer="[ /ncs:services/sls:syslog ]" >
        <name>1.2.3.4</name>
        <enabled refcounter="1" >true</enabled>
        <selector refcounter="1"  backpointer="[ /ncs:services/sls:syslog ]" >
          <name>1</name>
          <comparison refcounter="1" >same</comparison>
          <level refcounter="1" >info</level>
          <option refcounter="1"  backpointer="[ /ncs:services/sls:syslog ]" >pid</option>
          <facility refcounter="1"  backpointer="[ /ncs:services/sls:syslog ]" >all</facility>
        </selector>
      </server>
      <server>
        <name>10.3.4.5</name>
        <enabled>true</enabled>
        <selector>
          <name>8</name>
          <facility>auth</facility>
          <facility>authpriv</facility>
          <facility>local0</facility>
        </selector>
      </server>
```

```
      </syslog>
      </sys>
      </config>
  </device>
  </devices>
</config>

admin@ncs% request services syslog get-modifications
cli {
    local-node {
        data  devices {
                    device ex0 {
                        config {
                            r:sys {
                                syslog {
              +                     server 1.2.3.4 {
              +                         enabled;
              +                         selector 1 {
              +                             comparison same;
              +                             level info;
              +                             option [ pid ];
              +                             facility [ all ];
              +                         }
              +                     }
                                }
                            }
                        }
                    }
                    device ex1 {
                        config {
                            r:sys {
                                syslog {
              +                     server 1.2.3.4 {
              +                         enabled;
              +                         selector 1 {
              +                             comparison same;
              +                             level info;
              +                             option {
              +                                 pid;
              +                             }
              +                             facility [ all ];
              +                         }
              +                     }
                                }
                            }
                        }
                    }
                }

    }
}
```

It is important to re-deploy all affected services touching the device after the device migration, even though there are no backwards incompatible data model changes affecting the service. When reading the reverse/forward diffset of a service, NSO will detect changes to the NED identity of a device touched by the service and migrate the diffset on the fly. Thus the diffsets are still valid, but until the new diffset is written (typically through a re-deploy) this migration procedure will add extra time in handling the reverse/forward diffset (e.g. when using the get-modifications action).

To stop the example do:

```
$ ncs --stop
```

```
$ ncs-netsim stop
```