Leaf-list support for NSO

NSO Feature Demo

Cisco Tail-f
Stockholm, Sweden
2017-09-13

NSO 4.5

Backwards compatible: NO

New Representation of Yang leaf-lists for NSO

Context/background of why this feature/enhancement has been developed

The new representation simplifies the usage of YANG leaf-lists in service implementation. Many NSO customers are requesting this.

High level description of feature/enhancement

The internal representation of leaf-lists has changed. Leaf-lists now behave similar to lists in terms of operations, i.e., individual elements can now be created and deleted instead of setting/deleting the entire leaf-list as a whole entity. Attributes can be set on individual leaf-list elements which removes previous limitations on how service code can be written.

The feature/enhancement resolves the following problem

The lack of possibility to use shared leaf-list elements, which especially has been a problem for using FASTMAP.

Leaf-list illustration

If having leaf-list a { type string; } in the YANG model of the device, suppose there are two service instances s1 and s2 that both intend to set the same leaf-list element xyz.

Service instances:
- s1
- s2

After s1 and s2 are deployed the configuration of the device is a[xyz].

Old leaf-list implementation:
Deleting s1 removes a[xyz] from the device which is not correct as it still belongs to s2

New leaf-list implementation (Storing attributes on individual leaf-list elements is allowed):
Deleting s1 does not remove a[xyz] from the device as it still belongs to s2

Leaf-list Java service example

Java service code that uses NavuLeaf.set() or Maapi.setElem() (or the shared version of the same methods) to handle leaf-lists should be updated to use the new API. The recommended API to use now is NavuLeafList.sharedCreate() or Maapi.sharedCreate(). It is also possible to delete() a single leaf-list element. The vlan service in this example is the same as in examples.ncs/getting-started/developing-with-ncs/4-rfs-service, and the following diff illustrates the change that was made to migrate (from red to green lines)to the new API:

```
-------------------------------------------------------------------------------------------------------
                    unit.leaf(" enabled" ).sharedSet(new ConfBool(true));
                    unit.leaf(" description" ).sharedSet(
                    vlan.leaf(" description" ).value());-

Before migration:
                    ConfValue arp = vlan.leaf(" arp" ).value();
                    if (arp != null)
                            unit.leaf(" arp" ).sharedSet(arp);
After migration:
                    for (ConfValue arpValue : vlan.leafList(" arp" )) {
                            unit.leafList(" arp" ).sharedCreate(arpValue);
                    }
                }
        } catch (Exception e) {
            throw new DpCallbackException(" Could not instantiate service" , e);
-------------------------------------------------------------------------------------------------------
```

For a more details see examples.ncs/getting-started/developing-with-ncs/23-new-leaf-list-upgrade

Good to know

- Information about leaf-list can be found in the API documentation that is included in the NSO delivery package
- Service code must be modified in order to get the benefits of the new leaf-list representation
- It is still possible to use the old leaf-list representation (see next page for details)
- diffIterate will report create/delete for leaf-list elements instead of set/delete for the whole leaf-list (see notes on backwards compatibility).
- Data providers will get invocation of iterator()/existsOptional()/create()/remove()/numInstances()/moveAfter() data callbacks for leaf-list elements instead of getElem()/setElem()/remove() for the whole leaf-list (see notes on backwards compatibility).
- Entries in audit.log will now show individual create and delete for leaf-list elements.

Backward compatibility

Certain operations can be made to run in a backwards compatibility mode, making them behave as before, i.e., handling leaf-lists as single elements. The flags for controlling this behavior are deprecated, meaning that they will be removed in a future release.

- For the iteration functions in the CDB and MAAPI APIs, the flag ITER_WANT_LEAF_LIST_AS_LEAF can be used to get the old behavior, see the JavaDoc for com.tailf.conf.ConfIterateFlags and com.tailf.conf.DiffIterateFlags.

- For data providers, annotation @DpFlags (leafListAsLeaf = true) can be used to get the old behavior, see the JavaDoc for com.tailf.dp.annotations.DpFlags.

Limitations

- There is no option to downgrade to a previous version of NSO. See ncs-backup(1) regarding backup.
- CDB Subscribers on slaves cannot use the backwards compatibility flag for iteration
- If transactions on leaf-lists conflict with each other, iteration might produce unexpected results due to a set/delete on the whole list actually being composed of individual create/delete operations
- It is not possible to see attributes on leaf-list elements in the CLI using | display service-meta-data, but is is possible by using | display xml

Recommendation

- Always do a backup before upgrading. See ncs-backup(1)

- We recommend to use the new leaf-list mode for iteration as the backward compatibility mode is deprecated from the start and will be removed in the future

- Look at examples in
    - 4-rfs-service
    - 18-simple-service-erlang
    - 23-new-leaf-list-upgrade


CISCO