Developer Days
Network Services Orchestrator

Using Kickers to Support VNF's Upgrade

Ashish Jain
Software Consulting Engineer - CX
16-17 June 2020

“Computers have no idea what goes on outside of them except what humans tell them.”

Ellen Ullman

Quick Introduction

- Ashish Jain
- Software Team – CX India
- 10+ Years of experience in Automation & Orchestration

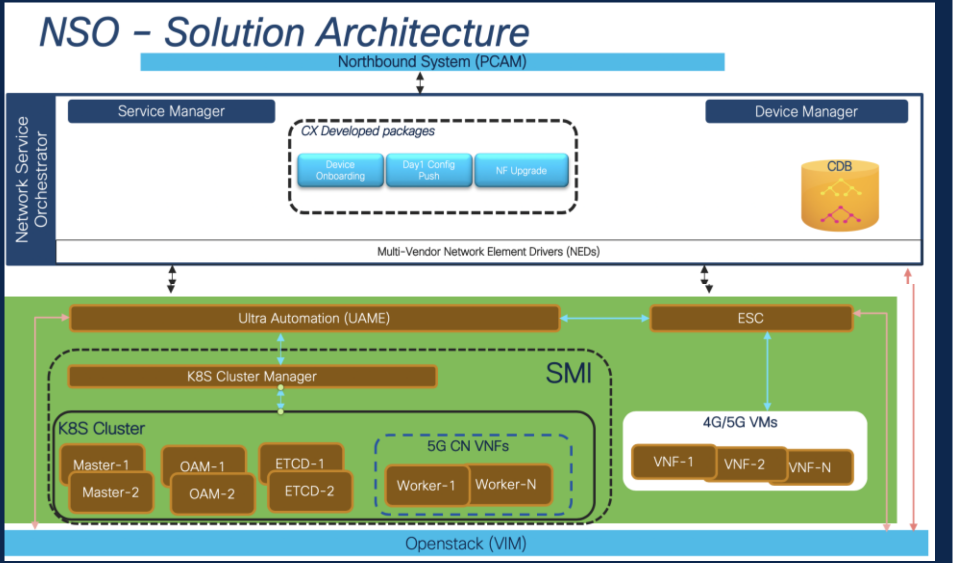
What is Kicker ?

- Declarative notification mechanism to trigger actions
- Actions will be triggered based on certain events like database change or received notifications
- Events and their Kickers are defined separately as data-kicker or notification-kicker
- Kickers are modeled in YANG and Kicker instances stored as configuration data in CDB

What is VNF ?

- A VNF is a function of the virtualized network (NFV- Network function Virtualization)
- It increases network scalability, helps in optimal use of network resources
- The requirement for specific hardware can be removed
- Can run as VMs and deploy any function

Architecture



- NF Upgrade package has been created in NSO
- These package needs to upgrade VNF’s based on certain events from Deployer/SMI
- VNF upgrades needs to show progress and status based on events

Problem Statement

1. NSO needs to start upgrading VNF’s only when Deployer/SMI sends certain kind of event.

- VNF’s are separate entity belonging to a POD and being controlled using orchestrator like deployer i.e. SMI
- NSO send single upgrade action to deployer by specifying the image and VNF name to be upgraded
- Deployer internally triggers image update action to tell VNF to use new image
- Deployer sends notifications to NSO that action sent to it is successful or failed.

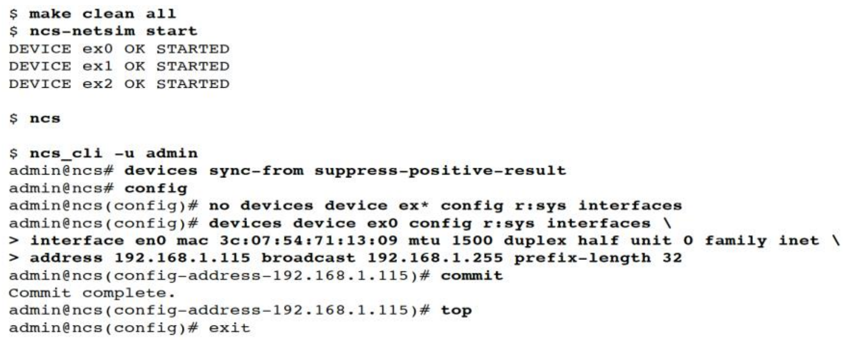
Problem Statement

- 2. VNF Upgrade will be started only when deployer send Success notification ,if its failed then NSO needs to trigger other actions on VNF.

- NSO received success notification from deployer,then it needs to wait for notification from VNF related to upgrade status.
- VNF’s sends notification which contains all information like image-version, upgrade percentage, VNF-state
- NSO needs to take action based on these notifications.
- NSO needs to show end-user the status of action he has triggered for upgrade.

What is the traditional way of taking actions based on notifications received?

Traditionally notifications from devices/nodes choosen to be stored in CDB as operational data and a separate CDB subscriber was used to act on the received notifications. With the use of notification-kicker the CDB subscriber can be removed and there is no longer any need to store the received notification in CDB.

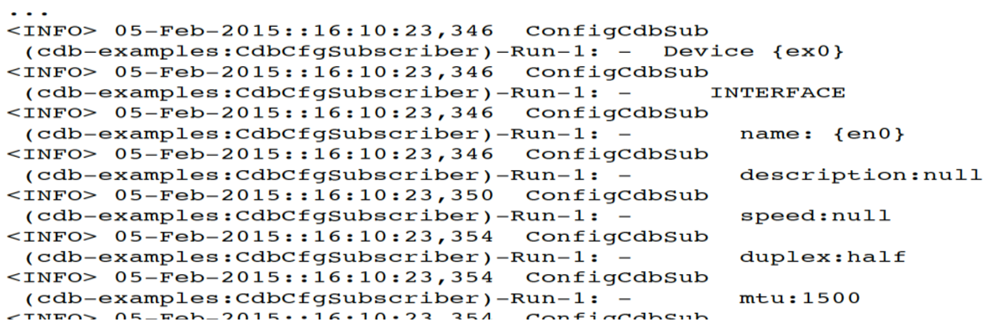


```
$ make clean all
$ ncs-netsim start
DEVICE ex0 OK STARTED
DEVICE ex1 OK STARTED
DEVICE ex2 OK STARTED

$ ncs

$ ncs_cli -u admin
admin@ncs# devices sync-from suppress-positive-result
admin@ncs# config
admin@ncs(config)# no devices device ex* config r:sys interfaces
admin@ncs(config)# devices device ex0 config r:sys interfaces \
> interface en0 mac 3c:07:54:71:13:09 mtu 1500 duplex half unit 0 family inet \
> address 192.168.1.115 broadcast 192.168.1.255 prefix-length 32
admin@ncs(config-address-192.168.1.115)# commit
Commit complete.
admin@ncs(config-address-192.168.1.115)# top
admin@ncs(config)# exit
```

Subscriber Output



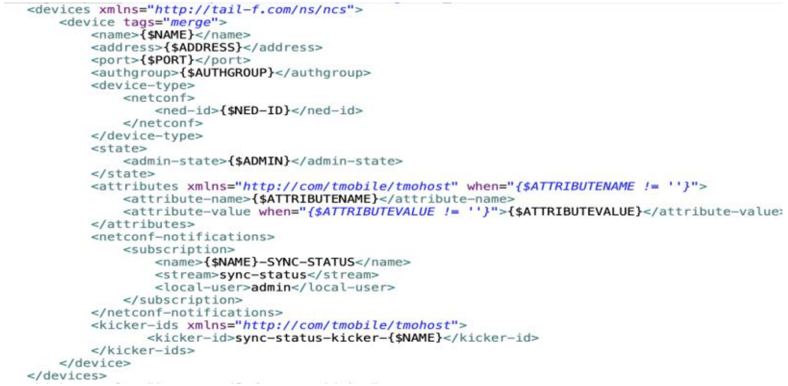
```
...
<INFO> 05-Feb-2015::16:10:23,346  ConfigCdbSub
 (cdb-examples:CdbCfgSubscriber)-Run-1: -  Device {ex0}
<INFO> 05-Feb-2015::16:10:23,346  ConfigCdbSub
 (cdb-examples:CdbCfgSubscriber)-Run-1: -     INTERFACE
<INFO> 05-Feb-2015::16:10:23,346  ConfigCdbSub
 (cdb-examples:CdbCfgSubscriber)-Run-1: -       name: {en0}
<INFO> 05-Feb-2015::16:10:23,346  ConfigCdbSub
 (cdb-examples:CdbCfgSubscriber)-Run-1: -       description:null
<INFO> 05-Feb-2015::16:10:23,350  ConfigCdbSub
 (cdb-examples:CdbCfgSubscriber)-Run-1: -       speed:null
<INFO> 05-Feb-2015::16:10:23,354  ConfigCdbSub
 (cdb-examples:CdbCfgSubscriber)-Run-1: -       duplex:half
<INFO> 05-Feb-2015::16:10:23,354  ConfigCdbSub
 (cdb-examples:CdbCfgSubscriber)-Run-1: -       mtu:1500
<INFO> 05-Feb-2015::16:10:23,354  ConfigCdbSub
```

Solution Kicker Provided

- Notification Kickers are triggered by the arrival of notifications from any device subscription.These subscriptions are defined under the /devices/device/netconf-notification/ subscription path.
- Storing the received notifications in CDB is optional and not part of the notification kicker functionality.
- The kicker invocations are serialized under a certain subscription i.e. kickers are invoked in the same sequence as notifications are received for the same subscription. This means that invocations are queued up and executed as quickly as the action permits.

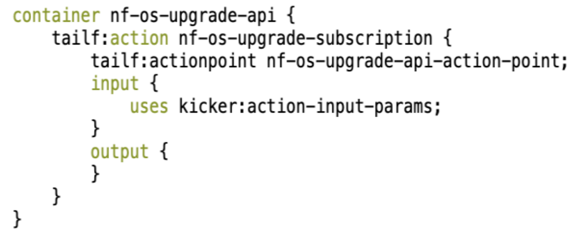
Kicker Configurations

Device Subscriptions



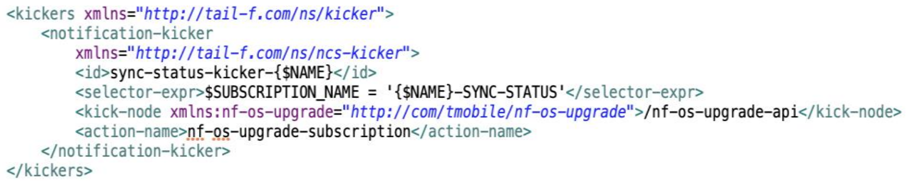
```xml
<devices xmlns="http://tail-f.com/ns/ncs">
    <device tags="merge">
        <name>{$NAME}</name>
        <address>{$ADDRESS}</address>
        <port>{$PORT}</port>
        <authgroup>{$AUTHGROUP}</authgroup>
        <device-type>
            <netconf>
                <ned-id>{$NED-ID}</ned-id>
            </netconf>
        </device-type>
        <state>
            <admin-state>{$ADMIN}</admin-state>
        </state>
        <attributes xmlns="http://com/tmobile/tmohost" when="{$ATTRIBUTENAME != ''}">
            <attribute-name>{$ATTRIBUTENAME}</attribute-name>
            <attribute-value when="{$ATTRIBUTEVALUE != ''}">{$ATTRIBUTEVALUE}</attribute-value
        </attributes>
        <netconf-notifications>
            <subscription>
                <name>{$NAME}-SYNC-STATUS</name>
                <stream>sync-status</stream>
                <local-user>admin</local-user>
            </subscription>
        </netconf-notifications>
        <kicker-ids xmlns="http://com/tmobile/tmohost">
               <kicker-id>sync-status-kicker-{$NAME}</kicker-id>
        </kicker-ids>
    </device>
</devices>
```

Yang Definition



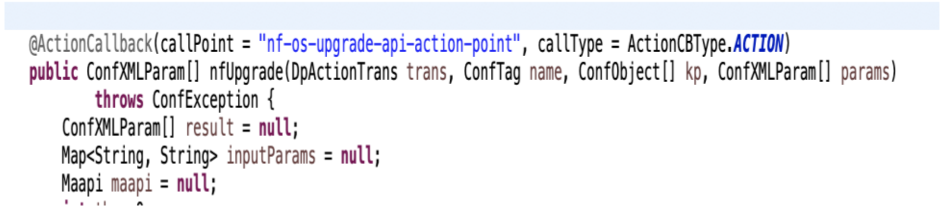
```
container nf-os-upgrade-api {
    tailf:action nf-os-upgrade-subscription {
        tailf:actionpoint nf-os-upgrade-api-action-point;
        input {
            uses kicker:action-input-params;
        }
        output {
        }
    }
}
```

Kicker Config



```xml
<kickers xmlns="http://tail-f.com/ns/kicker">
    <notification-kicker
        xmlns="http://tail-f.com/ns/ncs-kicker">
        <id>sync-status-kicker-{$NAME}</id>
        <selector-expr>$SUBSCRIPTION_NAME = '{$NAME}-SYNC-STATUS'</selector-expr>
        <kick-node xmlns:nf-os-upgrade="http://com/tmobile/nf-os-upgrade">/nf-os-upgrade-api</kick-node>
        <action-name>nf-os-upgrade-subscription</action-name>
    </notification-kicker>
</kickers>
```

Java Code Snippet



```java
@ActionCallback(callPoint = "nf-os-upgrade-api-action-point", callType = ActionCBType.ACTION)
public ConfXMLParam[] nfUpgrade(DpActionTrans trans, ConfTag name, ConfObject[] kp, ConfXMLParam[] params)
        throws ConfException {
    ConfXMLParam[] result = null;
    Map<String, String> inputParams = null;
    Maapi maapi = null;
```

Problem Statement Resolved

- Kickers helps in solving the VNF upgrade issue providing simple actions point after receiving the success/failure notifications from deployer.
- As soon as Deployer generates Success/Failure subscription, kicker action will be called immediately and subsequent actions will be taken related to same.
- VNF kickers will be triggered as soon as individual notifications being generated and kicker action will take care of showing the progress, upgrade status and individual VNF state.

cisco