Development Experience Sharing of PNF Software Upgrade Use Case

LFN Developer & Testing Forum, Jan 2020

Enbo Wang, Huawei

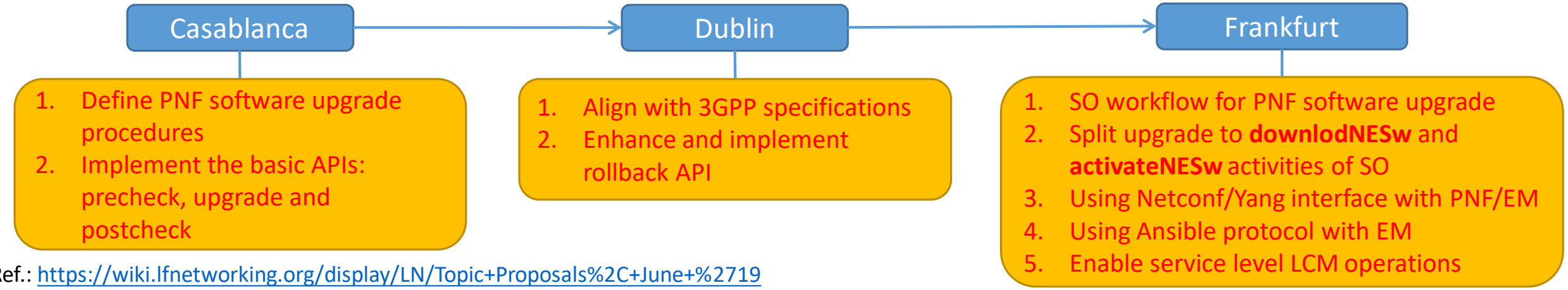
Outline

- Background & Roadmap
- Work done or doing about PNF software upgrade using Ansible in ONAP Frankfurt
  - What is delivered in SDNC for LCM APIs
  - What is done and doing in SO for LCM APIs
- Some experience and lessons
  - Using Minimal Environment for Daily Development and Test
  - Implementation of LCM Client
  - Implementation of Operations downlodNESw and activateNESw
- Next Steps

Background & Roadmap

- Software upgrade is an important part of network elements management and orchestration
- Software upgrade procedures should align with 3GPP specifications
- From SDNC level to E2E level (SO workflow)
- Add general Building Blocks to SO: preCheck, **downlodNESw**, **activateNESw** and postCheck
- Both support SS-API (Netconf/Yang) to PNF/EM and LCM API (Ansible) to EM

Ref. Wiki: https://wiki.onap.org/display/DW/PNF+software+upgrade+in+R6+Frankfurt

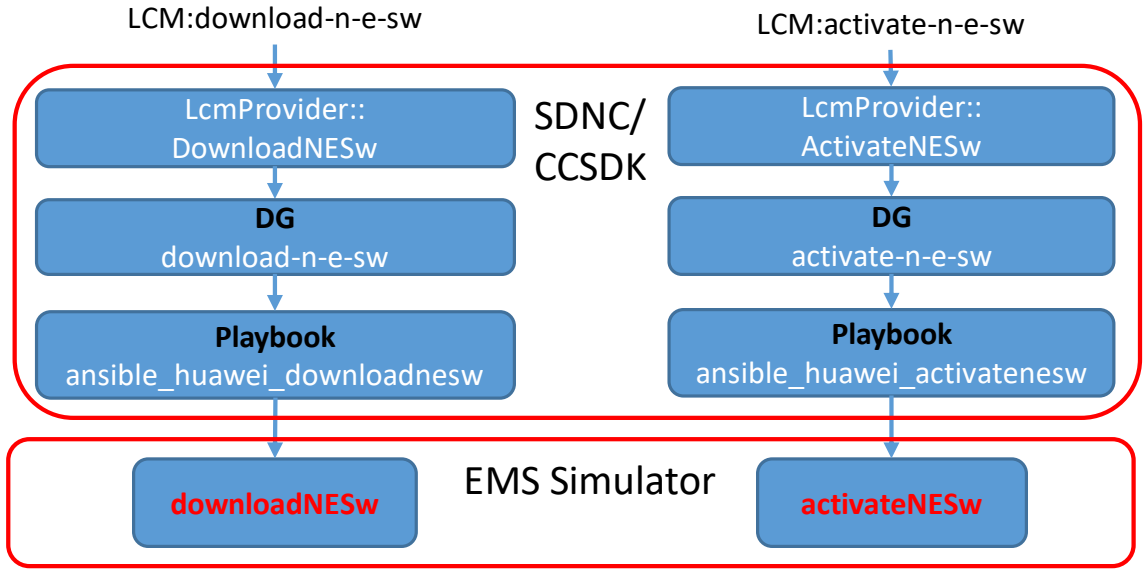


Ref.: https://wiki.lfnetworking.org/display/LN/Topic+Proposals%2C+June+%2719

What is delivered in SDNC for LCM APIs

1. Add northbound APIs to SDNC to interact with *downlodNESw* and *activateNESw* activities of SO
2. The operations are based on 3GPP TS 32.532 (SwM)

Ref. Wiki: https://wiki.onap.org/pages/viewpage.action?pageId=64007357



Yang models of download-n-e-sw and activate-n-e-sw:

```
rpc download-n-e-sw {
  description "An operation to download NE software";
  input {
    uses common-header;
    leaf action {
      type action;
      mandatory true;
    }
    uses action-identifiers;
    leaf payload {
      type payload;
      mandatory true;
    }
  }
  output {
    uses common-header;
    uses status;
    leaf payload {
      type payload;
      mandatory true;
    }
  }
}
```

```
rpc activate-n-e-sw {
  description "An operation to activate NE software";
  input {
    uses common-header;
    leaf action {
      type action;
      mandatory true;
    }
    uses action-identifiers;
    leaf payload {
      type payload;
      mandatory true;
    }
  }
  output {
    uses common-header;
    uses status;
    leaf payload {
      type payload;
      mandatory true;
    }
  }
}
```

Sample of payload field of input of LCM:download-n-e-sw:

"{\"pnf-flag\": \"true\", \"ipaddress-v4-oam\": \"192.168.35.83\", \"playbook-name\": \"ansible_huawei_downloadnesw\", \"pnfId\": \"5gDU0001\", \"swToBeDownloaded\": \"\\\\'[{\\\\\\\"swLocation\\\\\\\": \\\\\\\"http://192.168.35.96:10080/ran_du_pkg1-v2.zip\\\\\\\", \\\\\\\"swFileSize\\\\\\\": 353, \\\\\\\"swFileCompression\\\\\\\": \\\\\\\"GZIP\\\\\\\", \\\\\\\"swFileFormat\\\\\\\": \\\\\\\"binary\\\\\\\"}]\\\\'\"}"

Maybe need to simplify

Sample of payload field of input of LCM:activate-n-e-sw:

"{\"pnf-flag\": \"true\", \"ipaddress-v4-oam\": \"192.168.35.83\", \"playbook-name\": \"ansible_huawei_activatenesw\", \"pnfId\": \"5gDU0001\", \"swVersionToBeActivated\": \"v2\"}"

The playbook-name is optional. If not set the playbook-name, it will use the value in the configuration file */opt/onap/sdnc/data/properties/lcm-dg.properties* of SDNC.

What is done and doing in SO for LCM APIs

- SO API extension to retrieve PNF workflow (SO-2540)
  - Bind PNF model UUID to PNF workflow
  - Add *pnf_resource_to_workflow* table to SO *catalogdb*
  - Add its *select* statements

Done

```
MariaDB [catalogdb]> desc pnf_resource_to_workflow;
+-------------------------+-------------+------+-----+---------+----------------+
| Field                   | Type        | Null | Key | Default | Extra          |
+-------------------------+-------------+------+-----+---------+----------------+
| ID                      | int(11)     | NO   | PRI | NULL    | auto_increment |
| PNF_RESOURCE_MODEL_UUID | varchar(200)| NO   | MUL | NULL    |                |
| WORKFLOW_ID             | int(11)     | NO   | MUL | NULL    |                |
+-------------------------+-------------+------+-----+---------+----------------+
```

Ref. Wiki:
https://wiki.onap.org/display/DW/PNF+software+upgrade+in+R6+Frankfurt

- SO API extension to support PNF Upgrade (SO-2071)
  - Support PNF software upgrade with target software version
  - New added API:
    - /{version:[vV][1]}/serviceInstances/{serviceInstanceId}/pnfs/{pnfId}/workflows/{workflowUuid}

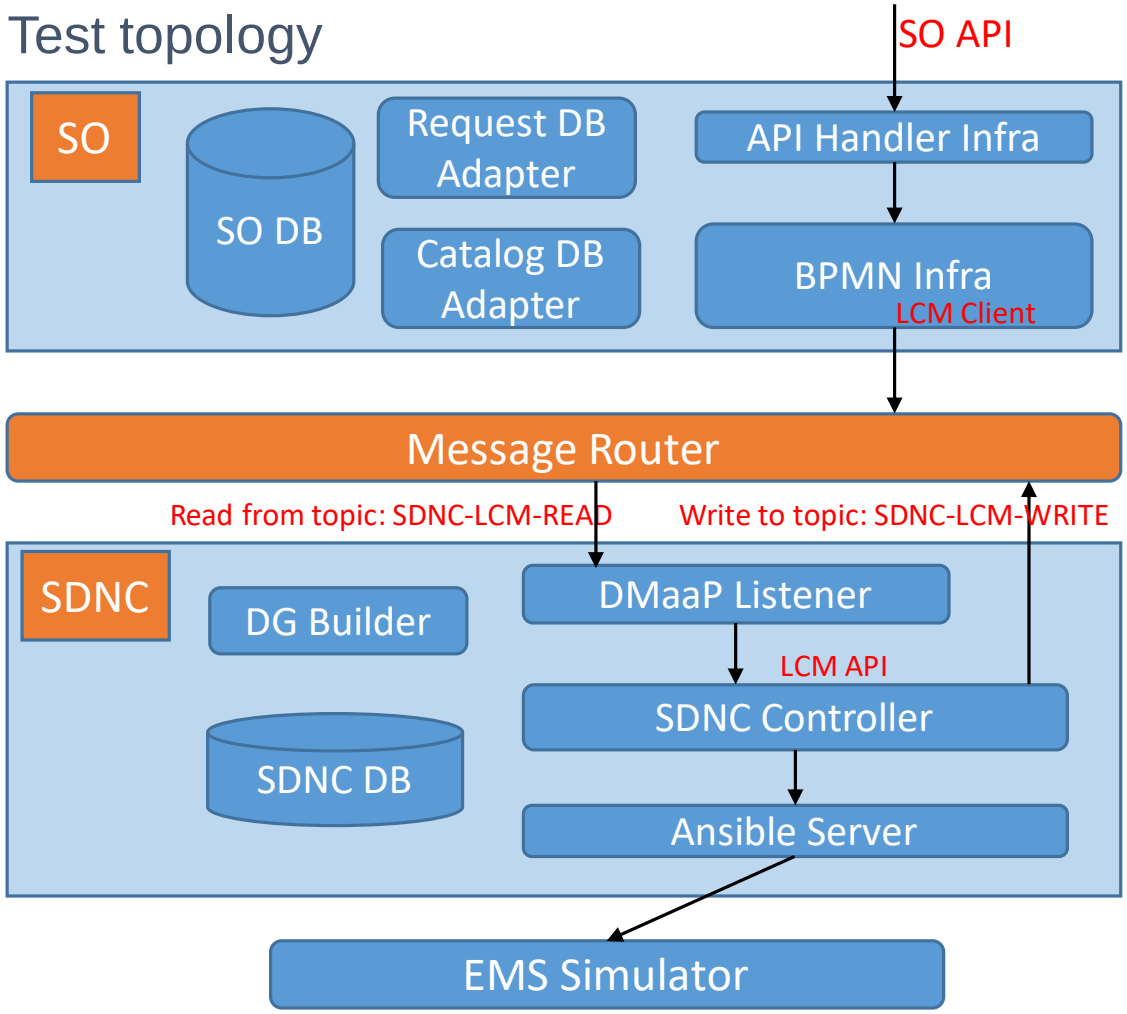
Done

- a generic decision points for API (SO-2070) WIP
- Create a new SO building block – preCheck (SO-2089) WIP
- SO LCM client extension to support PNF SW Upgrade (SO-2588) WIP

1 Using Minimal Environment for Daily Development and Test

Deploy the minimal environment used by daily development and test for SO and LCM APIs

Test topology



Docker containers

SO containers
so-api-handler-infra
so-bpmn-infra
so-request-db-adapter
so-catalog-db-adapter
so-db

SDNC containers
sdnc-controller
sdnc-dmaap-listener
sdnc-ansible-server
sdnc-dgbuilder
sdnc-db

Message Router containers
message-router
message-router-kafka
message-router-zookeeper

Initial Configuration:

1. Add two topics to Message Router:
   I. SDNC-LCM-READ
   II. SDNC-LCM-WRITE
2. Add Ansible Inventory to Ansible Server
3. Add Ansible playbook name to lcm-dg.properties of SDNC Controller

2 Implementation of LCM Client

- SO imports the appc-client as LCM Client
- LCM actions for PNF SW upgrade are implemented in SDNC, so do the new added downlodNESw and activateNESw
- That means, there are two Yang models to define LCM APIs: SDNC and APPC
  - Both SDNC/CCSDK and APPC implement LCM API provider
  - Only APPC implements LCM client, and used by SO
  - When add downlodNESw and activateNESw to SDNC, the appc-client of APPC doesn't have them
- Options:
  - Option 1: Add downlodNESw and activateNESw to APPC
  - Option 2: Implement sdnc-lcm-client in SDNC/CCSDK
  - Some others: so-sdnc-adaptor?

WIP

Need to discuss

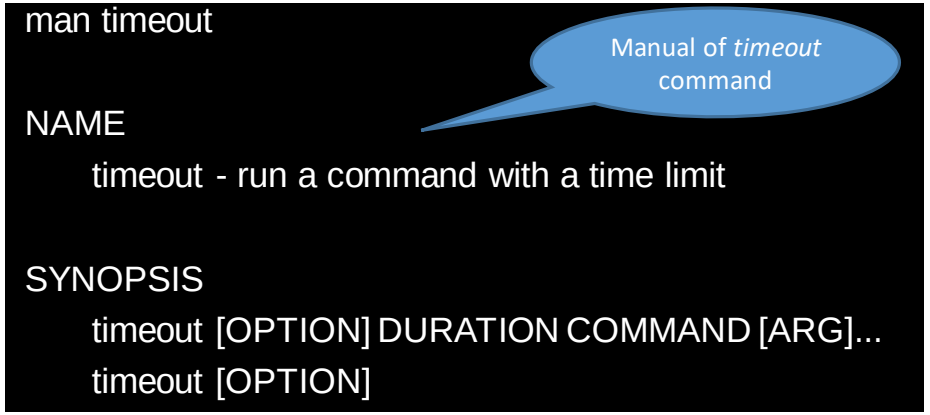
	Option 1	Option 2	Notes
Code Reusability	√		Need to align LCM models between APPC and SDNC
Maintainability		√	LCM models are independent

3.1 Implementation of downlodNESw and activateNESw Operations

- Using Ansible playbooks to implement downlodNESw and activateNESw operations between ONAP (ansible-server) and EMS Simulator
- In current implementation, downlodNESw or activateNESw returns until it has done the downloading or activating job
- ansible-server: Check is_alive() of a *Process* object in *multiprocessing* of Python to determine whether the operation is done

```
# check if playbook is still running
while ActiveProcess[input_data['Id']].is_alive():
    cherrypy.log("*** Playbook running returning PENDING for " + str(input_data['Id']))
    time.sleep(3)
```

Ref.: https://gerrit.onap.org/r/gitweb?p=ccsdk/distribution.git;a=blob;f=ansible-server/src/main/ansible-server/RestServer.py



In the *Process* object, run *ansible-playbook* command by *timeout* command, as:

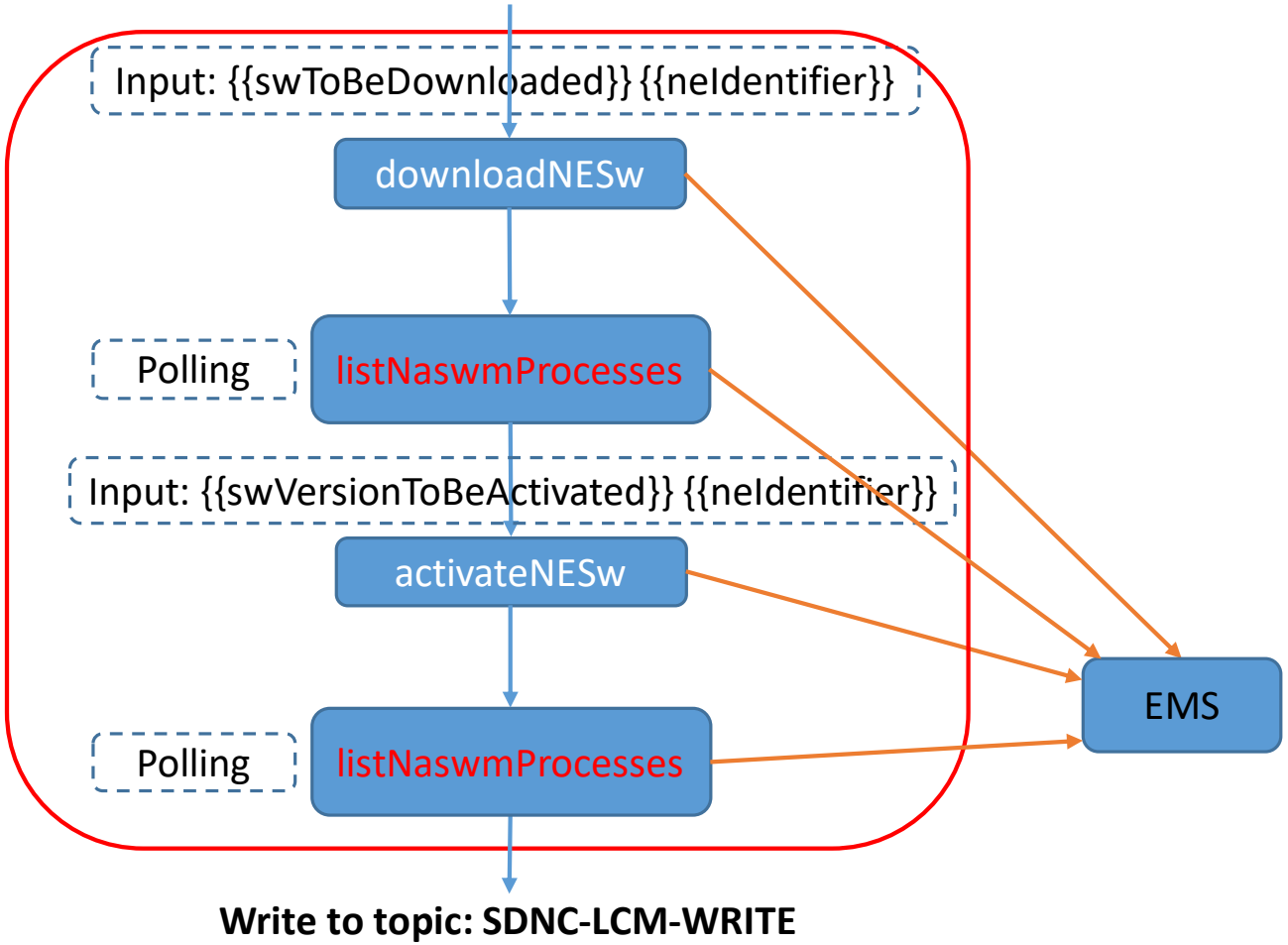
'timeout -s KILL -t ' + str(timeout) + ' ansible-playbook -v --timeout ' + str(timeout) + ...

Ref.: https://gerrit.onap.org/r/gitweb?p=ccsdk/distribution.git;a=blob;f=ansible-server/src/main/ansible-server/AnsibleModule.py

1. In practice, the *timeout parameter* is very difficult to be set a proper value.
2. How to know the progress?

3.2 Propose Option 1 to Enhance SW Upgrade

Option 1: Add new operation: listNaswmProcesses (3GPP TS 32.532)



Ref. 3GPP TS 32.532 (SwM)

Output of downloadNESw	Output of activateNESw
downloadProcessId result reason listOfStepNumbersAndDurations	activateProcessId result reason listOfStepNumbersAndDurations

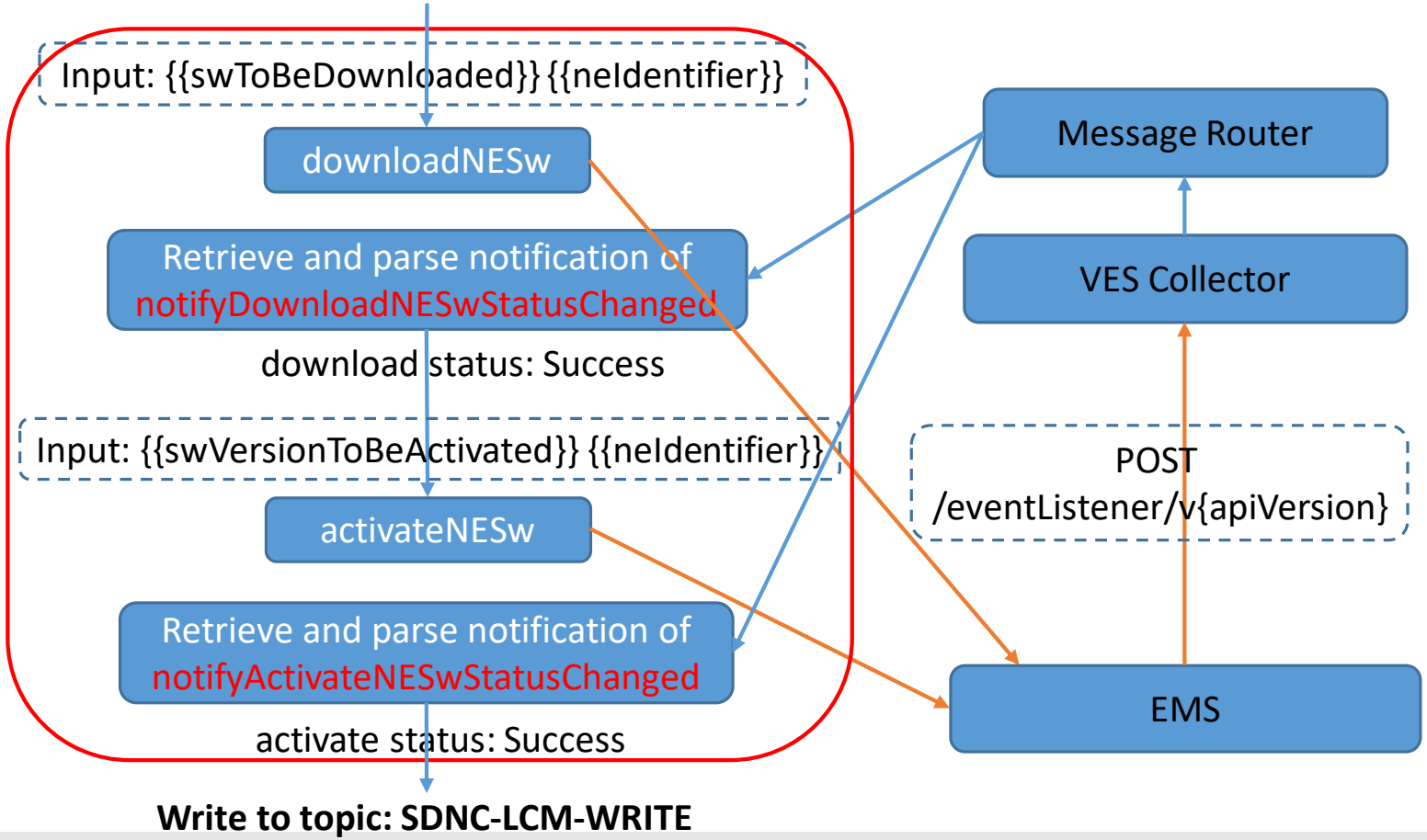
Input of listNaswmProcesses	Output of listNaswmProcesses
naswmProcessId naswmOperationType	naswmProcessList result reason

3.3 Propose Option 2 to Enhance SW Upgrade

Option 2: Add new VES notifications based on 3GPP TS 32.532:

notifyDownloadNESwStatusChanged

notifyActivateNESwStatusChanged



Ref. 3GPP TS 32.532 (SwM)

Output of downloadNESw	notifyDownloadNESwStatusChanged
downloadProcessId result reason listOfStepNumbersAndDurations	objectClass objectInstance notificationId eventTime systemDN notificationType downloadProcessId downloadOperationStatus downloadedNESwInfo failedSwInfo

Output of activateNESw	notifyActivateNESwStatusChanged
activateProcessId result reason listOfStepNumbersAndDurations	objectClass objectInstance notificationId eventTime systemDN notificationType activateProcessId activateOperationStatus swVersion failureReason

Next Steps

- Implementation of SO Building Blocks for LCM operations
- Discuss detailed parameters of SO API
  - e.g. does support the optional “playbook-name”?
- Simplify parameters of payload for LCM API
  - How to use the “*pnf-id*” in the *action-identifiers* of request of LCM API?

Thank You!