Thinking of PNF Software Upgrade Using Ansible

ONAP DDF, Jun 2019

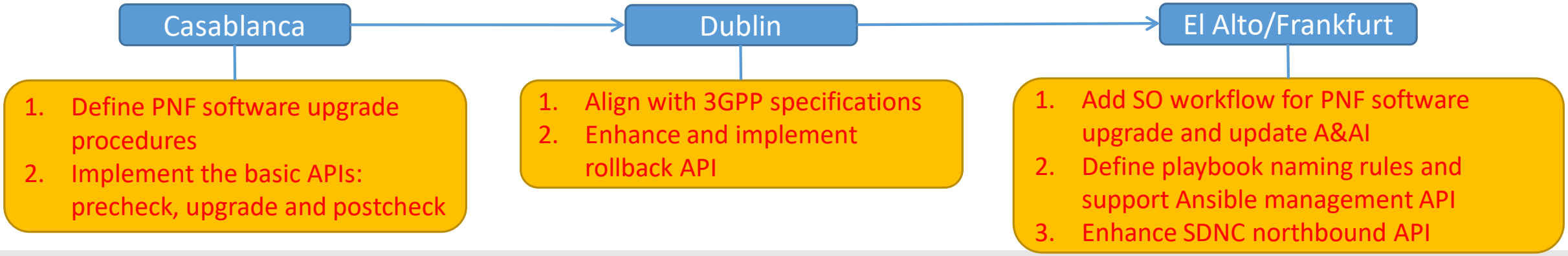
Enbo Wang, Huawei

Outline

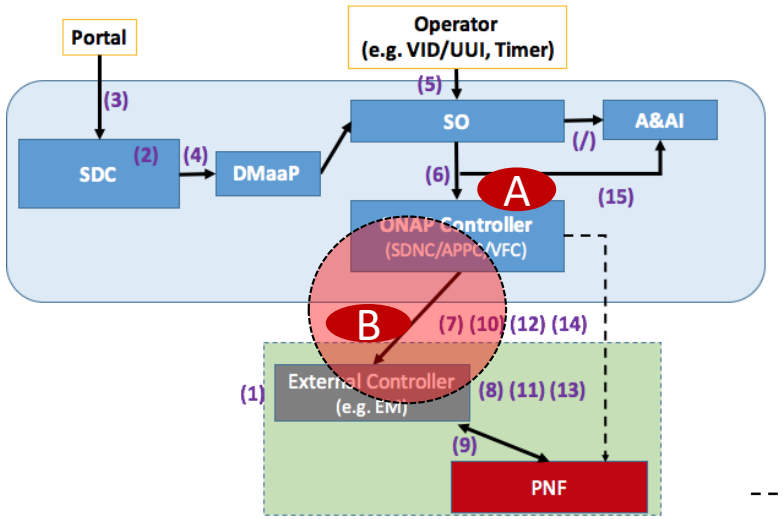
- Background & Roadmap
- Work done about PNF software upgrade using Ansible
  - What is delivered in Casablanca
  - What is delivered in Dublin
- Some proposals
  - Provide a SO Workflow for PNF S/W Upgrade
  - Query and Update A&AI
  - Support Ansible Management API
  - Enhance SDNC Northbound API

Background & Roadmap

- Software upgrade is an important part of network elements management and orchestration
- Some mechanisms of PNF software upgrade are the same as VNF software upgrade, such as DG and Ansible protocol
- Software upgrade procedures should align with 3GPP specifications
- Software upgrade procedures should be completeness, at least include **upgrade** and **rollback**
- A workflow for PNF software upgrade in SO is needed
- Software upgrade should support multi-vendors
- Software upgrade related APIs should be simple, readable and clear

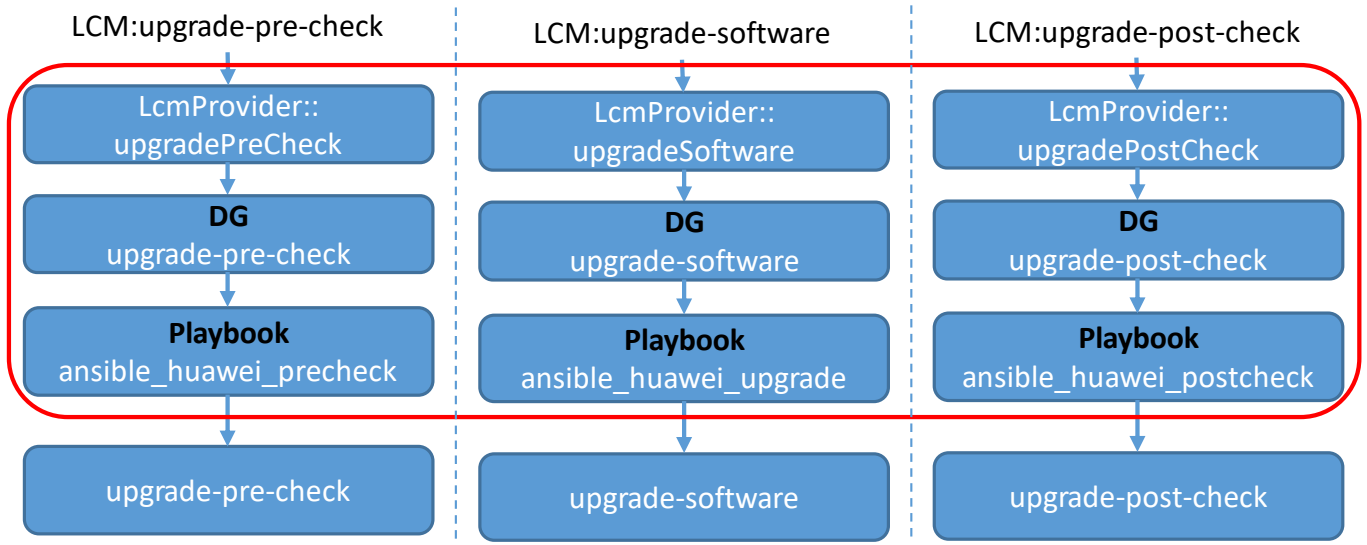


What is delivered in Casablanca



(1) Provisioning. Preparing necessary artifacts, and service register to ONAP (MSB).
(2) Upgrade workflow design.
(3) Service Modeling, pnfSwVersionList, pnfSwUpgradeRulesList.
(4) Distribute
(5) Execute workflow with necessary parameters.
(6) Call ONAP controller to execute actions.
(7) Precheck (skip update AAI entry)
(8) Apply checking rules, collect necessary data for prechecking
(9) Retrieval req/res
(10) S/W upgrade
(11) swDownload and activate
(12) postcheck (skip update AAI entry)
(13) Apply checking rules, collect and do comparison
(14) Upgrade Result notification (Succ/Fail)
(15) Update S/W version in AAI, e.g. 'active' version.

Direct PNF interface in the absence of EC in the workflow for configuration or software management



A

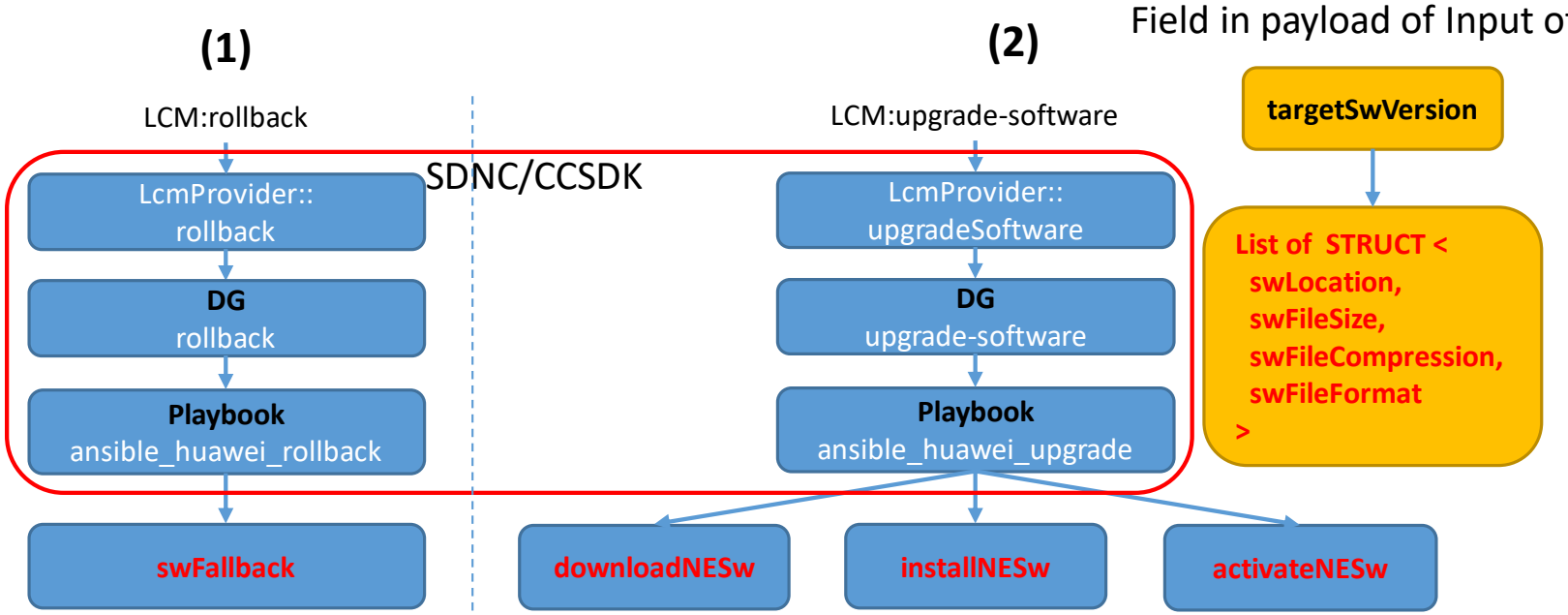
SDNC northbond API	Payload field of Input Parameters
POST /operations/LCM:upgrade-pre-check	"payload": "{\"**pnf-flag**\":\"true\", \"pnf-name\": \"5gDU0001\",\"**pnfId**\": \"5gDU0001\", \"**ipaddress-v4-oam**\": \"EC_ip\",\"**oldSwVersion**\": \"v1\", \"**targetSwVersion**\": \"v2\", \"**ruleName**\": \"r001\", \"Id\": \"10\", \"additionalData\":\"{}\"}"}}
POST /operations/LCM:upgrade-software	"payload": "{\"**pnf-flag**\":\"true\", \"pnf-name\": \"5gDU0001\",\"**pnfId**\": \"5gDU0001\", \"**ipaddress-v4-oam**\": \"EC_ip\",\"**oldSwVersion**\": \"v1\", \"**targetSwVersion**\": \"v2\", \"Id\": \"10\", \"additionalData\":\"{}\"}"}}
POST /operations/LCM:upgrade-post-check	"payload": "{\"**pnf-flag**\":\"true\", \"pnf-name\": \"5gDU0001\",\"**pnfId**\": \"5gDU0001\", \"**ipaddress-v4-oam**\": \"EC_ip\",\"**oldSwVersion**\": \"v1\", \"**targetSwVersion**\": \"v2\", \"**ruleName**\": \"r102\", \"Id\": \"10\", \"additionalData\":\"{}\"}"}}

B

Playbook	Parameters
ansible_huawei_precheck	{{**pnfId**}} {{**oldSwVersion**}} {{**targetSwVersion**}} {{**ruleName**}}
ansible_huawei_upgrade	{{**pnfId**}} {{**oldSwVersion**}} {{**targetSwVersion**}}
ansible_huawei_postcheck	{{**pnfId**}} {{**oldSwVersion**}} {{**targetSwVersion**}} {{**ruleName**}}

What is delivered in Dublin (1): Scope

1. Rollback (or Fallback) API based on 3GPP TS 32.532 SwM *swFallback* operation support for EC/xNF
2. API and playbook enhancement for software-upgrade API based on 3GPP TS 32.532 SwM operations: *downloadNESw*, *installNESw* and *activateNESw*



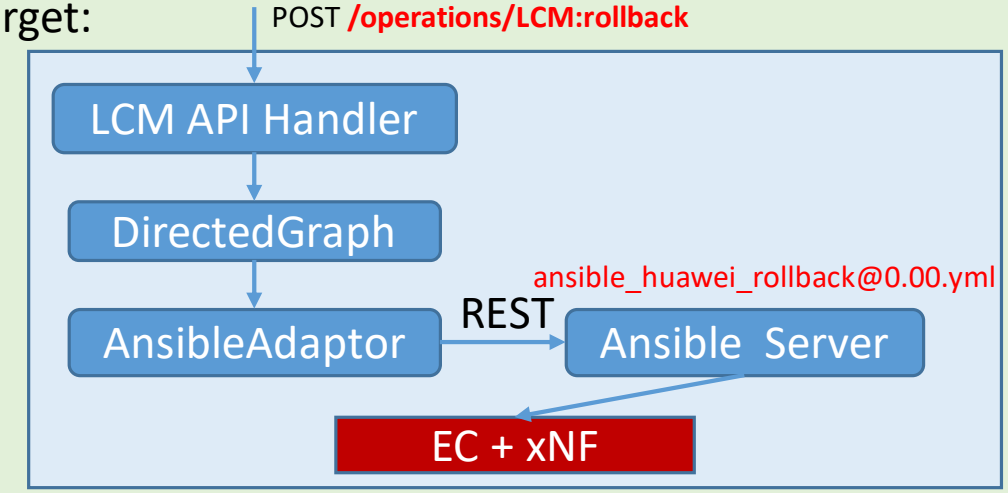
Field in payload of Input of northbound API:

- Sample of payload field of input:

"{\"pnf-flag\":\"true\", \"pnfId\": \"5gDU0001\", \"ipaddress-v4-oam\": \"192.168.35.83\", \"targetSwVersion\": \"\\\\'[{\\\\\\\"swLocation\\\\\\\": \\\\\\\"http://192.168.35.96:10080/ran_du_pkg1-v2.zip\\\\\\\", \\\\\\\"swFileSize\\\\\\\": 353, \\\\\\\"swFileCompression\\\\\\\": \\\\\\\"ZIP\\\\\\\", \\\\\\\"swFileFormat\\\\\\\": \\\\\\\"zip\\\\\\\"}, {\\\\\\\"swLocation\\\\\\\": \\\\\\\"http://192.168.35.96:10080/ran_du_pkg2-v2.zip\\\\\\\", \\\\\\\"swFileSize\\\\\\\": 353, \\\\\\\"swFileCompression\\\\\\\": \\\\\\\"ZIP\\\\\\\", \\\\\\\"swFileFormat\\\\\\\": \\\\\\\"zip\\\\\\\"}]\\\\\"}"

What is delivered in Dublin (2): rollback LCM API support

Target:



Status:

Yang model of rollback defined in CCSDK

```
rpc rollback {
  description "An operation to rollback to particular snapshot of a virtual network function (or VM)";
  input {
    uses common-header;
    leaf action {
      type action;
      mandatory true;
    }
    uses action-identifiers;
    leaf payload {
      type payload;
      mandatory false;
    }
    leaf identity-url {
      type string;
      mandatory true;
    }
    leaf snapshot-id {
      type string;
      mandatory true;
    }
  }
  output {
    uses common-header;
    uses status;
  }
}
```

Processing method in LcmProvider.java can be re-used in *ccsdk/sli/northbound* repo:

```
public ListenableFuture<RpcResult<RollbackOutput>> rollback(RollbackInput input) {
  ……
}
```

- Solution: re-using existing rollback action

```
rpc rollback {
  description "An operation to rollback to particular snapshot of a virtual network function (or VM)";
  input {
    uses common-header;
    leaf action {
      type action;
      mandatory true;
    }
    uses action-identifiers;
    leaf payload {
      type payload;
      mandatory false;
    }
    leaf identity-url {
      type string;
      mandatory false;
    }
    leaf snapshot-id {
      type string;
      mandatory false;
    }
  }
  output {
    uses common-header;
    uses status;
    leaf payload {
      type payload;
      mandatory false;
    }
  }
}
```

Modify the Yang Model of rollback in *ccsdk/sli/northbound* repo:

1. Modify mandatory of identity-url and snapshot-id in input parameter to ***false***;
2. Add an optional payload field in output parameter.

- Samples:
- Payload filed of input:

"{\"pnf-flag\":\"true\", \"ipaddress-v4-oam\": \"192.168.35.83\", \"filter\": \"\\\\'$..[?(@.nEIdentification == \\\\\\\"5gDU0001\\\\\\\")]\\\\'\"}"

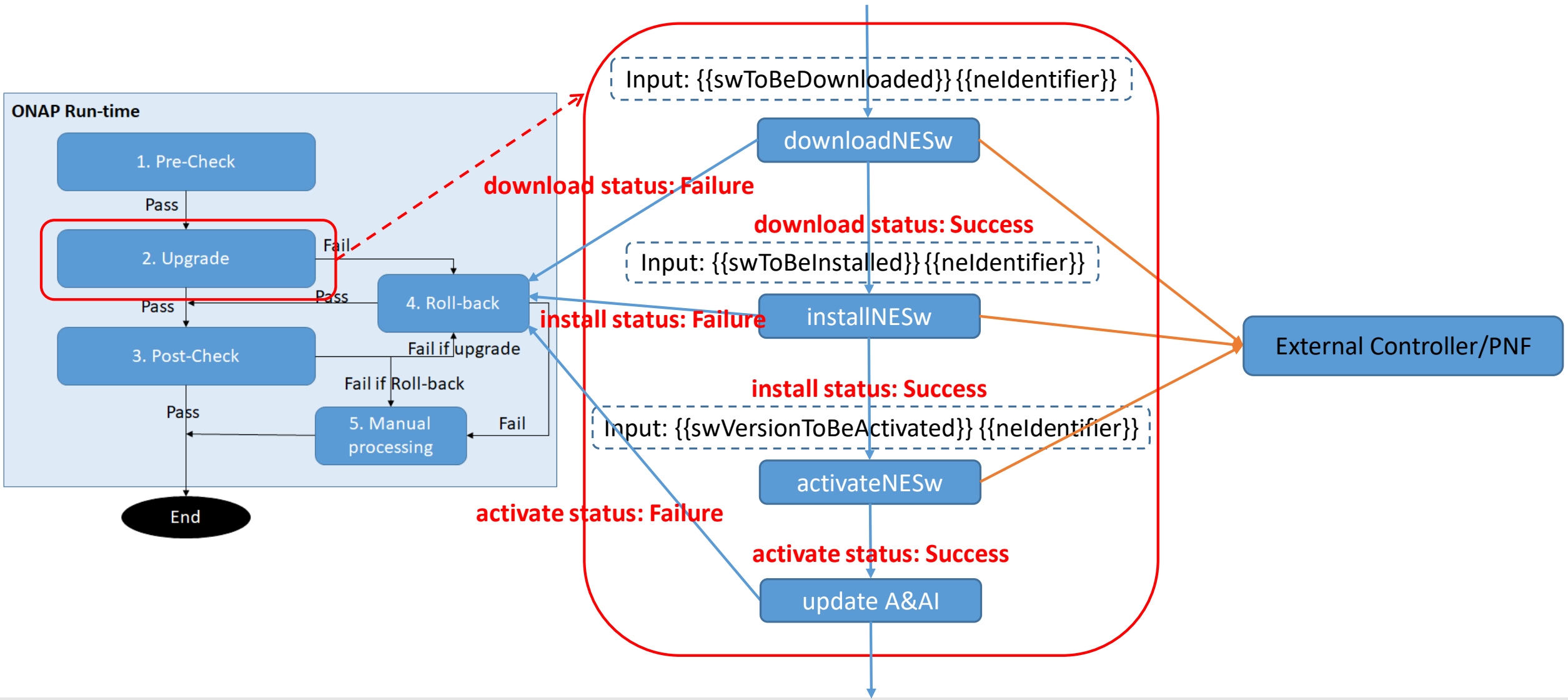
- Payload field of output:

"{\"nEList\": [{\"nEIdentification\": \"5gDU0001\", \"swFallbackStatus\": \"fallbackSuccessful\"}], \"result\": \"Success\"}"

- All Test Cases:

https://wiki.onap.org/display/DW/5G+-+PNF+SW+Upgrade+-+Integration+Test+Cases

Requirement 1: Provide a SO Workflow for PNF S/W Upgrade



Requirement 2: Query and Update A&AI

- Propose to define the naming rules of playbooks:
  - PlaybookName:= <equip-vendor>_<equip-model>_<lcm-action>@<version>
- Query A&AI
  - Generate *PlaybookName* according to A&AI fields and lcm-action name
  - Retrieve the *ipaddress-v4-oam*
- Update A&AI
  - Update software-version(s) when upgrade-software or rollback

aai/schema-service/aai-schema/src/main/resources/onap/oxm/v16/aai_oxm_v16.xml
https://gerrit.onap.org/r/gitweb?p=aai/schema-service.git;a=blob;f=aai-schema/src/main/resources/onap/oxm/v16/aai_oxm_v16.xml

Related fields in **PNF** Schema:

java-attribute	name	description
pnfName	pnf-name	unique name of Physical Network Function.
pnfId	pnf-id	id of pnf
equipType	equip-type	Equipment type. Source of truth should define valid values.
equipVendor	equip-vendor	Equipment vendor. Source of truth should define valid values.
equipModel	equip-model	Equipment model. Source of truth should define valid values.
ipaddressV4Oam	ipaddress-v4-oam	ipv4-oam-address with new naming convention for IP addresses
swVersion	sw-version	sw-version is the version of SW for the hosted application on the PNF.
pnfIpv4Address	pnf-ipv4-address	This is the IP address (IPv4) for the PNF itself. This is the IPv4 address that the PNF itself can be accessed at.
softwareVersions	software-versions	Collection of software versions. Array of SoftwareVersion.

Related fields in SoftwareVersion Schema:

java-attribute	name	description
softwareVersionId	software-version-id	Identifier of the software version
isActiveSwVer	is-active-sw-ver	used to indicate whether or not this software-version is the active one (activeSw = true)

Requirement 3: Ansible Management API support

- Ansible Server
  - Inventory Management API
    - Create EMS/xNF host entry: POST /inventory, parameters: hostname_or_ip, connection_type, ssh_user, ssh_private_key_file or ssh_password
    - Get EMS/xNF host entry: GET /inventory[/{hostname_or_ip}]
    - Update EMS/xNF host entry: PUT /inventory/{hostname_or_ip}
    - Delete EMS/xNF host entry: DELETE /inventory/{hostname_or_ip}
  - Playbook Management API
    - Create Playbook: POST /playbooks, parameters: equip-vendor, equip-model, lcm-action, version; body: <playbook>
    - Get Playbook(s): GET /playbooks[/{playbookname}]
    - Update Playbook : PUT /playbooks/{playbookname}
    - Delete Playbook: DELETE /playbooks/{playbookname}
  - Impact
    - **Support management of multi-vendors' inventories of External Controller/PNF**
    - **Support management of multi-vendors' playbooks**

Requirement 4: Enhance SDNC Northbound LCM APIs

- LCM APIs: upgrade-pre-check, upgrade-software, upgrade-post-check and rollback
  - Format of input payload is not readable and **dependent on** internal implementation
    - Too many escapes for double quotes:
      - upgrade-software: \"targetSwVersion\": \"\\\\\'[{\\\\\\\"swLocation\\\\\\\": \\\\\\\"http://192.168.35.96:10080/ran_du_pkg1-v2.zip\\\\\\\", ...
      - rollback: \"filter\": \"\\\\\'$..[?(@.nEIdentification == \\\\\\\"5gDU0001\\\\\\\")]\\\\\'\"
    - Dependent on the times of parsing JSON string of input payload
  - Add new fields of input for specified LCM API
    - **Add new fields to YANG models for upgrade-software and rollback**
  - Output payload is not unified currently:
    - upgrade-pre-check: payload is mandatory (https://gerrit.onap.org/r/c/ccsdk/sli/northbound/+/75605)
    - upgrade-software: **no payload**
    - upgrade-post-check: payload is mandatory (https://gerrit.onap.org/r/c/ccsdk/sli/northbound/+/75605)
    - rollback: payload is optional (https://gerrit.onap.org/r/c/ccsdk/sli/northbound/+/82646)
    - Proposal: unify to ***optional payload***

Conclusions

- Provide a SO Workflow for PNF S/W Upgrade and update A&AI
- Define playbook naming rules and support Ansible management API to support multi-vendors' playbooks coexistence
- Enhance SDNC northbound API to simplify input parameters of LCM APIs

Thank You!