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

---

1. Define PNF software upgrade procedures
2. Implement the basic APIs: precheck, upgrade and postcheck

1. Align with 3GPP specifications
2. Enhance and implement rollback API

1. Add SO workflow for PNF software upgrade and update A&AI
2. Define playbook naming rules and support Ansible management API
3. Enhance SDNC northbound API
What is delivered in Casablanca

Playbook

<table>
<thead>
<tr>
<th>SDNC northbond API</th>
<th>Payload field of Input Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST /operations/LCM:upgrade-pre-check</td>
<td>&quot;payload&quot;: {&quot;pnf-flag&quot;:true, &quot;pnf-name&quot;: &quot;5gDU0001&quot;, &quot;pnfid&quot;: &quot;5gDU0001&quot;, &quot;ipaddress-v4-oam&quot;: &quot;EC_ip&quot;, &quot;oldSwVersion&quot;: &quot;v1&quot;, &quot;targetSwVersion&quot;: &quot;v2&quot;, &quot;ruleName&quot;: &quot;r001&quot;, &quot;Id&quot;: &quot;10&quot;, &quot;additionalData&quot;: &quot;{}&quot;}}</td>
</tr>
<tr>
<td>POST /operations/LCM:upgrade-software</td>
<td>&quot;payload&quot;: {&quot;pnf-flag&quot;:true, &quot;pnf-name&quot;: &quot;5gDU0001&quot;, &quot;pnfid&quot;: &quot;5gDU0001&quot;, &quot;ipaddress-v4-oam&quot;: &quot;EC_ip&quot;, &quot;oldSwVersion&quot;: &quot;v1&quot;, &quot;targetSwVersion&quot;: &quot;v2&quot;, &quot;Id&quot;: &quot;10&quot;, &quot;additionalData&quot;: &quot;{}&quot;}}</td>
</tr>
<tr>
<td>POST /operations/LCM:upgrade-post-check</td>
<td>&quot;payload&quot;: {&quot;pnf-flag&quot;:true, &quot;pnf-name&quot;: &quot;5gDU0001&quot;, &quot;pnfid&quot;: &quot;5gDU0001&quot;, &quot;ipaddress-v4-oam&quot;: &quot;EC_ip&quot;, &quot;oldSwVersion&quot;: &quot;v1&quot;, &quot;targetSwVersion&quot;: &quot;v2&quot;, &quot;ruleName&quot;: &quot;r102&quot;, &quot;Id&quot;: &quot;10&quot;, &quot;additionalData&quot;: &quot;{}&quot;}}</td>
</tr>
</tbody>
</table>

Playbook

|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 \textit{swFallback} operation support for EC/xNF

2. API and playbook enhancement for software-upgrade API based on 3GPP TS 32.532 SwM operations: \textit{downloadNESw}, \textit{installNESw} and \textit{activateNESw}

Field in payload of Input of northbound API:

- Sample of payload field of input:

  ```json
  {"pnf-flag": "true", "pnfId": "5gDU0001", "ipaddress-v4-oam": "192.168.35.83", "targetSwVersion": "\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\n```
What is delivered in Dublin (2): rollback LCM API support

**Target:** POST /operations/LCM:rollback

- LCM API Handler
- DirectedGraph
- AnsibleAdaptor
- REST
- Ansible Server

**Status:**

- **Yang model of rollback defined in CCSDK**

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

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

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

**Solution:** re-using existing rollback action

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

- Payload field of output:
  ```json
  "{"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

1. Pre-Check
   - Input: \{\text{swToBeDownloaded}\} \{\text{neIdentifier}\}
   - Status: download status: Failure

2. Upgrade
   - Input: \{\text{swToBeInstalled}\} \{\text{neIdentifier}\}
   - Status: install status: Failure

3. Post-Check
   - Status: Fail if Roll-back

4. Roll-back
   - Status: Fail if upgrade

5. Manual processing

End of Workflow

External Controller/PNF

Statuses:
- download status: Success
- install status: Success
- activate status: Success
- download status: Failure
- install status: Failure
- activate status: Failure

update A&AI
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

---

**Related fields in PNF Schema:**

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

**Related fields in SoftwareVersion Schema:**

<table>
<thead>
<tr>
<th>java-attribute</th>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>softwareVersionId</td>
<td>software-version-id</td>
<td>Identifier of the software version</td>
</tr>
<tr>
<td>isActiveSwVer</td>
<td>is-active-sw-ver</td>
<td>used to indicate whether or not this software-version is the active one (activeSw = true)</td>
</tr>
</tbody>
</table>
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\\\\": \\\\"filter\\\\": \\\\"$.([?.@.nElIdentification == "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!