1. VNF SDK Project Home Page ................................................................. 2
1.1 Project Planning ..................................................................................... 4
   1.1.1 Mercury Release ............................................................................. 4
      1.1.1.1 VNF SDK Mercury Release Architecture Milestone Deliverables (M3) ........ 4
      1.1.1.2 VNF SDK Mercury Release Code Freeze Milestone Deliverables (M4) .......... 5
      1.1.1.3 VNF SDK Mercury Release Deliverables for Release Candidate Milestone (RC0) . 6
      1.1.1.4 VNF SDK Mercury Release Deliverables for Release Candidate Milestone (RC1) . 6
      1.1.1.5 VNF SDK Mercury Release Deliverables for Release Candidate Milestone (RC2) . 7
      1.1.1.6 VNF SDK Mercury Release Functionality Freeze Deliverables (M2) ................. 8
      1.1.1.7 VNF SDK Mercury Release Planning (M1) ........................................... 9
      1.1.1.8 VNF SDK Mercury Release Planning Deliverables (M1) ......................... 17
1.2 VNF SDK Architecture and Design ..................................................... 18
   1.2.1 VNF SDK Flow Diagram ............................................................... 18
   1.2.2 VNF SDK Use Case Definitions .................................................... 21
1.3 VNF SDK Blueprint ............................................................................ 27
   1.3.1 VNF SDK - Guide for Bundling VNFs .............................................. 32
1.4 VNF SDK Documentation .................................................................. 33
1.5 VNF SDK Meetings ........................................................................... 34
   1.5.1 February 9, 2017 VNF SDK Meeting Minutes ................................. 36
   1.5.2 January 20, 2017 VNF SDK Meeting Minutes .................................. 36
   1.5.3 Meeting Materials .......................................................................... 36
1.6 VNF SDK Old Home Page .................................................................. 37
1.7 VNF SDK Project Proposal ................................................................. 40
1.8 VNF SDK References .......................................................................... 42
1.9 VNF SDK Sub Projects ...................................................................... 44
   1.9.1 Design&Package Tool ................................................................. 45
   1.9.2 Function Test .............................................................................. 46
   1.9.3 Market Place ............................................................................... 49
   1.9.4 Validate & Lifecycle Test ............................................................. 51
1.10 VNF SDK Testing ............................................................................. 54
   1.10.1 VNF SDK Function Test CSIT use cases .................................... 54
   1.10.2 VNF SDK Marketplace CSIT use cases ..................................... 55
   1.10.3 VNF SDK package tool CSIT use cases ..................................... 55
   1.10.4 VNF SDK Validation / Lifecycle Test CSIT use cases ................. 56
Project Detail

<table>
<thead>
<tr>
<th>Project Description</th>
<th>The &quot;VNF Supplier APIs and SDK&quot; (short name: VNF SDK) project delivers functionality to enable a rich OPEN-O community ecosystem of VNF product suppliers and operators, with high degree of interoperability.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project PTL</td>
<td>Lizongbiao, Huawei</td>
</tr>
<tr>
<td>Participants</td>
<td>Cannonical, China Mobile, China Telecom, GigaSpaces, Huawei, Intel, ZTE</td>
</tr>
<tr>
<td>Personnel</td>
<td>Resources</td>
</tr>
</tbody>
</table>

Project Proposal and Planning

<table>
<thead>
<tr>
<th>Project Proposal</th>
<th>VNF SDK Project Proposal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Planning</td>
<td>VNF SDK Mercury Release Planning (M1)</td>
</tr>
<tr>
<td></td>
<td>VNF SDK Mercury Release Planning Deliverables (M1)</td>
</tr>
<tr>
<td></td>
<td>VNF SDK Mercury Release Functionality Freeze Deliverables (M2)</td>
</tr>
<tr>
<td>Testing</td>
<td>VNF SDK Testing</td>
</tr>
<tr>
<td>Documentation</td>
<td>VNF SDK Documentation</td>
</tr>
<tr>
<td>Legal</td>
<td></td>
</tr>
</tbody>
</table>

Sub-Projects

<table>
<thead>
<tr>
<th>Sub-Project</th>
<th>Description</th>
<th>Owner</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Design & Package Tool

- **Validate & Lifecycle Test**
  1. Develop Validate component to help validate VNF Package.
  2. Develop Lifecycle Test component to help test VNF lifecycle, e.g. instantiation, termination, etc.
  - Yuan Liu & Anbing Zhang
  - [Goto Validate & Lifecycle Test Page](#)

- **Function Test**
  1. Develop Function Test Framework to help test VNF functionality.
  2. Develop guidelines to help VNF vendors to plugin their VNF to the framework.
  - Murali Mohan Murthy
  - [Goto Function Test Page](#)

- **Market Place**
  - Alex & Sukesh

---

### Meeting Schedules

#### Logistics and Meeting Summary

#### Recently Updated

- **gao weitao**
  - Validate & Lifecycle Test updated Apr 28, 2017 • view change
  - VNF SDK Mercury Release Deliverables for Release Candidate Milestone (RC1) created Apr 13, 2017
  - VNF SDK Mercury Release Deliverables for Release Candidate Milestone (RC2) updated Apr 13, 2017 • view change
  - VNF SDK Mercury Release Deliverables for Release Candidate Milestone (RC0) updated Apr 13, 2017 • view change

- **Gildas Lanilis**
  - VNF SDK Project Home Page updated Mar 17, 2017 • view change

- **gao weitao**
  - VNF SDK Mercury Release Code Freeze Milestone Deliverables (M4) updated Mar 07, 2017 • view change

- **Brian Hedstrom**
  - Mercury Release updated Mar 03, 2017 • view change
  - Project Planning commented Mar 03, 2017
  - VNF SDK References updated Mar 03, 2017 • view change
  - VNF SDK Testing updated Mar 03, 2017 • view change

- **gao weitao**
  - VNF SDK Flow Diagram updated Mar 03, 2017 • view change
Project Planning

OPEN-O VNF SDK Design(11.15)

OPEN-O_VNF_SDK_Design(11.15) (1).pptx

Mercury Plan 2016-12-14

OPEN-O VNF-SDK Planning in Mercury Release(2106.12.15).pptx

Mercury Release

- VNF SDK Mercury Release Architecture Milestone Deliverables (M3)
- VNF SDK Mercury Release Code Freeze Milestone Deliverables (M4)
- VNF SDK Mercury Release Deliverables for Release Candidate Milestone (RC0)
- VNF SDK Mercury Release Deliverables for Release Candidate Milestone (RC1)
- VNF SDK Mercury Release Deliverables for Release Candidate Milestone (RC2)
- VNF SDK Mercury Release Functionality Freeze Deliverables (M2)
- VNF SDK Mercury Release Planning (M1)
- VNF SDK Mercury Release Planning Deliverables (M1)

VNF SDK Mercury Release Architecture Milestone Deliverables (M3)

<table>
<thead>
<tr>
<th>Practice Area</th>
<th>Checkpoint</th>
<th>Yes/No</th>
<th>Evidences</th>
<th>How to?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Has the Project team reviewed the APIs with the Architecture Committee (ARC)?</td>
<td>Yes</td>
<td></td>
<td>Architecture walkthrough to understand how each project contributes on Release Use Case. ARC to organize the walkthrough.</td>
</tr>
<tr>
<td></td>
<td>Is there a plan to address the findings the API review?</td>
<td>Yes</td>
<td>There have been no findings for API</td>
<td>The plan could be as simple as a Jira issue to track the implementation of findings or a documented plan within the wiki.</td>
</tr>
<tr>
<td></td>
<td>Does the team clearly understand that no changes in the API definition is allowed without formal TSC review and approval?</td>
<td>Yes</td>
<td>NA</td>
<td>In the case some changes are necessary, bring the request to the TSC for review and approval.</td>
</tr>
<tr>
<td></td>
<td>Is there any changes in the scope, functionalities, deliverable, resources, API, repositories since M1 milestone?</td>
<td>No</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Release Management

<table>
<thead>
<tr>
<th>Checkpoint</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Are committed Sprint Backlog Stories been marked as &quot;Done&quot; in Jira board?</td>
<td>Yes</td>
</tr>
<tr>
<td>Are all tasks associated with Sprint Backlog Stories been marked as &quot;Done&quot; in Jira?</td>
<td>Yes</td>
</tr>
<tr>
<td>Have all findings from previous milestones been addressed?</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Development

<table>
<thead>
<tr>
<th>Checkpoint</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Has the project team reach the Automated Unit Test Code Coverage expectation? (Refer to artifacts available in Sonar)</td>
<td>Yes</td>
</tr>
<tr>
<td>Is there any pending commit request older than 24 hours in Gerrit?</td>
<td>No</td>
</tr>
<tr>
<td>Are all the Jenkins jobs successfully passed (Merge-Jobs)?</td>
<td>Yes</td>
</tr>
<tr>
<td>Are all binaries available in Nexus?</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Integration and Testing

<table>
<thead>
<tr>
<th>Checkpoint</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Have 50% of Continuous System Integration Testing (CSIT) Use Cases been implemented successfully in Jenkins?</td>
<td>Yes</td>
</tr>
<tr>
<td>Have you reviewed all the CSIT Test Case with Integration Team?</td>
<td>Yes</td>
</tr>
<tr>
<td>Has the project code successfully passed the Daily Build process?</td>
<td>Yes</td>
</tr>
</tbody>
</table>

## VNF SDK Mercury Release Code Freeze Milestone Deliverables (M4)

The following items are expected to be completed for the project to Pass the M4 Code Freeze Milestone.

### M4 Release Code Freeze Milestone overview is available in wiki.

**Usage**

1. Copy and Paste this template in the wiki Space along with your Release document
2. Fill out the Yes/No column
3. Provide link to evidence (when necessary)

## Practice Area

<table>
<thead>
<tr>
<th>Checkpoint</th>
<th>Yes/No</th>
<th>Evidences</th>
<th>How to?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Management</td>
<td>All all JIRA Stories supporting the release use case been implemented?</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>List the Stories that will not be implemented in this current Release.</td>
<td>Yes</td>
<td>Venus Story List</td>
</tr>
<tr>
<td></td>
<td>Are committed Sprint Backlog Stories been coded and marked as &quot;Done&quot; in Jira?</td>
<td>In Progress</td>
<td>will done by 3/8</td>
</tr>
<tr>
<td></td>
<td>Are all tasks associated with committed Sprint Backlog Stories been marked as &quot;Done&quot; in Jira?</td>
<td>Yes</td>
<td>Task List</td>
</tr>
<tr>
<td>Release Management</td>
<td>Have all issues pertaining to FOSS been addressed?</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Have all findings from previous milestones been addressed?</td>
<td></td>
<td>Provide link to JIRA findings</td>
</tr>
<tr>
<td>Development</td>
<td>Are all Defects of priority Highest and High in status &quot;Done&quot; in Jira?</td>
<td>Yes</td>
<td>VNF-SDK Highest and high Bug</td>
</tr>
<tr>
<td></td>
<td>Have all Google Coding Style issues been addressed?</td>
<td>Yes</td>
<td>OPEN-O Java code style</td>
</tr>
<tr>
<td></td>
<td>Is there any pending commit request older than 24 hours in Gerrit?</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
### VNF SDK Mercury Release Deliverables for Release Candidate Milestone (RC0)

<table>
<thead>
<tr>
<th>Practice Area</th>
<th>Checkpoint</th>
<th>Yes/No</th>
<th>Evidences</th>
<th>How to?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Management</td>
<td>Are all tasks associated with the release been marked as &quot;Done&quot; in Jira?</td>
<td>Yes</td>
<td>VNF-SDK JIRA Task</td>
<td></td>
</tr>
<tr>
<td>Release Management</td>
<td>As the project team identified the reduced set of Committers who will proceed with Code Merge after RC0?</td>
<td>Yes</td>
<td>Victor Gao</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Have all findings from previous milestones been addressed?</td>
<td>Yes</td>
<td>VNF-SDK JIRA Issues</td>
<td></td>
</tr>
<tr>
<td>Development</td>
<td>Have all Defects of priority Highest and High been in status &quot;Done&quot; in Jira?</td>
<td>Yes</td>
<td>VNF-SDK JIRA Issues</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is there any pending commit request older than 24 hours in Gerrit?</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Are all the Jenkins jobs successfully passed (verify + merge jobs)?</td>
<td>Yes</td>
<td>Verify Jobs Merge Jobs</td>
<td></td>
</tr>
<tr>
<td>Integration and Testing</td>
<td>Have all CSIT Use Cases (created by each project team) passed?</td>
<td>Yes</td>
<td>CSIT Jobs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Have all end-to-end CSIT (created by Integration Team) been identified?</td>
<td>Yes</td>
<td>Link to evidence</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is there a Docker images available for each repository?</td>
<td>Yes</td>
<td>VNF-SDK Docker</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Has the project code successfully passed the Daily Build process?</td>
<td>Yes</td>
<td>VNF-SDK Daily Build</td>
<td>Goal is to ensure the latest project commit has not broken the Integration Daily Build</td>
</tr>
<tr>
<td>Documentation</td>
<td>Is the team ready to contribute in the following 3 documentations:</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1. Mercury Release Notes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. Mercury Tarball Installation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Mercury Docker Installation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### VNF SDK Mercury Release Deliverables for Release Candidate Milestone (RC1)

<table>
<thead>
<tr>
<th>Practice Area</th>
<th>Checkpoint</th>
<th>Yes/No</th>
<th>Evidences</th>
<th>How to?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Management</td>
<td>Are all tasks associated with the release been marked as &quot;Done&quot; in Jira?</td>
<td>Yes</td>
<td>VNF-SDK JIRA Task</td>
<td></td>
</tr>
<tr>
<td>Release Management</td>
<td>As the project team identified the reduced set of Committers who will proceed with Code Merge after RC0?</td>
<td>Yes</td>
<td>Victor Gao</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Have all findings from previous milestones been addressed?</td>
<td>Yes</td>
<td>VNF-SDK JIRA Issues</td>
<td></td>
</tr>
<tr>
<td>Development</td>
<td>Have all Defects of priority Highest and High been in status “Done” in Jira?</td>
<td>Yes</td>
<td>VNF-SDK JIRA Issues</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------------------------------------------</td>
<td>-----</td>
<td>---------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is there any pending commit request older than 24 hours in Gerrit?</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Are all the Jenkins jobs successfully passed (verify + merge jobs)?</td>
<td>Yes</td>
<td>Verify Jobs Merge Jobs</td>
<td></td>
</tr>
<tr>
<td>Integration and Testing</td>
<td>Have all CSIT Use Cases (created by each project team) passed?</td>
<td>Yes</td>
<td>CSIT Jobs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Have all end-to-end CSIT (created by Integration Team) been identified?</td>
<td>Yes</td>
<td>Link to evidence</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is there a Docker images available for each repository?</td>
<td>Yes</td>
<td>VNF-SDK Docker</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Has the project code successfully passed the Daily Build process?</td>
<td>Yes</td>
<td>VNF-SDK Daily Build</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Goal is to ensure the latest project commit has not broken the Integration Daily Build</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td>Is the team ready to contribute in the following 3 documentations:</td>
<td>Yes</td>
<td>Each service in VNF-SDK already have docker, so we don't need Tarball installation</td>
<td></td>
</tr>
</tbody>
</table>
|             | 1. Mercury Release Notes  
|             | 2. Mercury Tarball Installation  
|             | 3. Mercury Docker Installation |

**VNF SDK Mercury Release Deliverables for Release Candidate Milestone (RC2)**

<table>
<thead>
<tr>
<th>Practice Area</th>
<th>Checkpoint</th>
<th>Yes/No</th>
<th>Evidences</th>
<th>How to?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Management</td>
<td>Are all tasks associated with the release been marked as “Done” in Jira?</td>
<td>Yes</td>
<td>VNF-SDK JIRA Task</td>
<td></td>
</tr>
<tr>
<td>Release Management</td>
<td>(For RC0 only) As the project team identified the reduced set of Committers who will proceed with Code Merge after RC0?</td>
<td>NA</td>
<td>Provide the name of committers. Recommendation is 3 committers max per project (1 per geographic area)?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Have all findings from previous milestones been addressed?</td>
<td>Yes</td>
<td>VNF-SDK JIRA Issues</td>
<td></td>
</tr>
<tr>
<td>Development</td>
<td>Have all Defects of priority Highest and High been in status “Done” in Jira?</td>
<td>Yes</td>
<td>VNF-SDK JIRA Issues</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Are all JIRA issues used as a reference in a commit in status “Done” in Jira?</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is there any pending commit request older than 24 hours in Gerrit?</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Are all the Jenkins jobs successfully passed (verify + merge jobs)?</td>
<td>Yes</td>
<td>Verify Jobs Merge Jobs</td>
<td></td>
</tr>
<tr>
<td>Integration and Testing</td>
<td>Have all CSIT Use Cases (created by each project team) passed?</td>
<td>Yes</td>
<td>CSIT Jobs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Have all end-to-end CSIT (created by Integration Team) been identified (for RC0) and passed (for RC1)?</td>
<td>Yes</td>
<td>Link to evidence</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is there a Docker images available for each repository?</td>
<td>Yes</td>
<td>VNF-SDK Docker</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Has the project code successfully passed the Daily Build process?</td>
<td>Yes</td>
<td>VNF-SDK Daily Build</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Goal is to ensure the latest project commit has not broken the Integration Daily Build</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Practice Area</td>
<td>Checkpoint</td>
<td>Yes/No</td>
<td>Evidence - Comment</td>
<td>How to?</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------</td>
<td>-------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td><strong>Documentation</strong></td>
<td>Is the team ready to contribute in the following 3 documentations:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1. Mercury Release Notes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. Mercury Tarball Installation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Mercury Docker Installation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>VNF SDK Mercury Release Functionality Freeze Deliverables (M2)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Product Management</strong></td>
<td>Are all provisional APIs interface (stub) defined (at beta-quality level)?</td>
<td>Yes</td>
<td>Function Test Interface</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is there a final list of externally consumable APIs available?</td>
<td>Yes</td>
<td>VNF-SDK Outgoing APIs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For all completed Sprints, have Sprint Backlog Stories been marked as &quot;Done&quot;</td>
<td>Yes</td>
<td>VNF-SDK Sprint 1</td>
<td>Difference between a Product and Sprint</td>
</tr>
<tr>
<td></td>
<td>in Jira?</td>
<td></td>
<td></td>
<td>Backlog</td>
</tr>
<tr>
<td></td>
<td>Are all tasks associated with the completed Sprint Backlog Stories been</td>
<td>Yes</td>
<td>VNF-SDK Sprint 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>marked as &quot;Done&quot; in Jira?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Has all Product Backlog not implemented in current release been scoped for</td>
<td>Yes</td>
<td>JIRA Venus Release Backlog</td>
<td>to document</td>
</tr>
<tr>
<td></td>
<td>next release?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Release Management</strong></td>
<td>Have all source code files been updated with License Apache 2 header?</td>
<td>Yes</td>
<td>Definition of Source code file</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Has the year format in copyright header of all source code files been</td>
<td>Yes</td>
<td>Guidance on year format</td>
<td></td>
</tr>
<tr>
<td></td>
<td>updated? (Rules for new files created in 2017 and existing files modified in</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2017 are different)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>In case source code can't be edited, has a &quot;License.txt&quot; file been placed</td>
<td>Yes</td>
<td>Guidance for source code file that</td>
<td></td>
</tr>
<tr>
<td></td>
<td>at the root directory for which the license is applicable?</td>
<td></td>
<td>can't be edited</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Has the project FOSS Table been updated with latest third party code?</td>
<td>Yes</td>
<td>VNF-SDK FOSS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Do you have a plan to address any issue raised by Fossology?</td>
<td>Yes</td>
<td>TBD</td>
<td></td>
</tr>
<tr>
<td><strong>Development</strong></td>
<td>For new projects approved for this release, has all source code been</td>
<td>Yes</td>
<td>VNF-SDK Gerrit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>placed into Gerrit?</td>
<td></td>
<td>MarketPlace/Design Pkg Tool/Function</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Test already committed Source code,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>lc test still coding now</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Has the project team reach the Automated Unit Test Code Coverage</td>
<td>No</td>
<td>Goal: 50% for Incubation project</td>
<td>Guidance on Code Coverage and Static Code</td>
</tr>
<tr>
<td></td>
<td>expectation? (Refer to artifacts available in Sonar)</td>
<td></td>
<td>Now Only UI Code, UT Coverage</td>
<td>Analysis Tools: Sonar</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>not involved.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>71% Code Coverage reached for repo</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>vnf-sdk-design-pkg</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(code in python, report made on</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>local env)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is there any binaries (jar, war, tar, gz, gzip, zip files) in Gerrit project</td>
<td>No</td>
<td>• NEVER embed jar, war,</td>
<td>Practices for Mercury Release</td>
</tr>
<tr>
<td></td>
<td>repository?</td>
<td></td>
<td>tar, gz, gzip, zip in Gerrit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Could you ensure that all proprietary trademarks, logos, product names,</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>company name, etc. have been removed? All OPEN-O deliverables must comply</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>with this rule and be agnostic of any proprietary symbols.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is there any pending commit request older than 24 hours in Gerrit?</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Are all the Jenkins jobs successfully passed (merge jobs)?</td>
<td>Yes</td>
<td>VNF-SDK Jenkins Job</td>
<td></td>
</tr>
</tbody>
</table>
Are all snapshot binaries available in Nexus? | In Progress | VNF-SDK Nexus
---|---|---

Integration and Testing

| Have Continuous System Integration Testing (CSIT) Use Cases been documented in wiki? | Yes | VNF SDK Testing
---|---|---

| Have you implemented in Jenkins at least 1 CSIT test case for each of the project repository? | In Progress | As an evidence, provide a link to Jenkins (CSIT Jobs) that shows a sample test case implemented (1 job for each repo).
---|---|---

| Has the project code successfully passed the Daily Build process? | Yes | VNF-SDK Merge Job
---|---|---

**VNF SDK Mercury Release Planning (M1)**

- 1 Overview
- 2 Scope
  - 2.1 What is this release trying to address?
  - 2.2 Use Cases
  - 2.3 Minimum Viable Product
  - 2.4 Functionalities
  - 2.5 Longer term roadmap
- 3 Release Deliverables
- 4 Sub-Components
- 5 Architecture
  - 6.1 API Dependencies
  - 6.2 API Incoming Dependencies
  - 6.3 API Outgoing Dependencies
  - 6.4 Third Party Products Dependencies
- 7 Testing and Integration Plans
  - 7.1 Function Test Plan and Resources
- 8 Gaps
- 9 Known Defects and Issues
- 10 Risks
- 11 Resources
- 12 Release Milestone
- 13 Team Internal Milestone
- 14 Project Plan
  - 14.1 Vendor Neutral
  - 14.2 Free and Open Source Software
  - 14.3 Documentation, Training, Tutorial
  - 14.4 Board policy (including IPR)
- 15 Release key facts

**Overview**

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Enter the name of the project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Release Name</td>
<td>VNF SDK</td>
</tr>
<tr>
<td>Target Release Name</td>
<td>Mercury Release</td>
</tr>
</tbody>
</table>
**Project Lifecycle State**  | Incubation  
---|---  
**Participating Company**  | CMCC, Gigaspaces, Huawei, Intel, ZTE  

**Scope**

**What is this release trying to address?**

- Currently VNF on-boarding, instantiation and maintenance procedures, used by NFVI operators are highly manual, time consuming and specifically tailored to individual operators, VIMs, NFV infrastructures and VNF types.
- VNF suppliers want to participate in a large ecosystem for more market opportunities, and to develop their VNF products in a way that is attractive to the largest range of buyers.
- In order to support multiple service and business scenarios, VNF suppliers want to package and deliver their VNF products in a way that is highly standardized, automated and configurable.
- VNF suppliers, who rely on specific NFVI and VIM capabilities, as well as telemetry feedback, want to ensure that their VNF infrastructure and VIM dependencies are met, so that their VNF product can function as designed.
- Current versions of ETSI/MANO VNF package (VNFD) and VNF record (VNFR) definitions prevent VNF suppliers from expressing NFVI and VIM dependencies and requirements.

**Use Cases**

This release will also add support of the Mercury release Customer Use Case: Mercury Use Case.

**Minimum Viable Product**

MVP has the following features:

- Design & Package Tools
- Validate & Lifecycle Test
- Function Test
- Market Place

**Functionality**

List the functionalities that this release is committing to deliver.

<table>
<thead>
<tr>
<th>Functionality Name</th>
<th>In or Out</th>
<th>Priority</th>
<th>Stretch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package &amp; Designer Tools</td>
<td>IN</td>
<td>H</td>
<td>VNF Package Designer, part of VNF Supplier SDK tools.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The below are the tasks we need to do:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The command line interface (CLI) for designer (optional?).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Provide a graphical tool to define the VNF product model and package for VNF vendor engine.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Standardized VNF product packaging based on TOSCA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• VNF Package builder</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• VNF Package validator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• VNF Package extractor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• VNF Package Parser</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• VNF Package Dry Run</td>
</tr>
<tr>
<td>Validate &amp; Lifecycle Test</td>
<td>IN</td>
<td>H</td>
<td>Lifecycle test should do the task as below:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Reuse the onboarding operation for VNF package in NFVO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Reuse the installation operation for VNF Package in NFVO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Reuse the resource allocation operation for VNF package in NFVO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Reuse the workflow in NFVO</td>
</tr>
<tr>
<td>Functional Test</td>
<td>IN</td>
<td>H</td>
<td>The function test for VNF-SDK should do as below:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Supply a FT framework for VNF product developer/ NFV operator/VNF product Dev-Ops teams</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Support some measurements (like VNF KPI) about the FT</td>
</tr>
</tbody>
</table>
Market place for VNF-SDK should do the task as below:

- Upload and download of VNF products and offerings
- Supply a user-friendly GUI for operator/User
- Show the interface about other component in VNF-SDK (just like Lifecycle test, Functional Test, Upload, Download, etc)

Longer term roadmap

- SDN-O consideration
  - PNF
  - VNF FT that SDN-O related (connectivity)
  - Controller/Driver certification
- Network Service On-boarding consideration

Release Deliverables

Indicate the outcome (Executable, Source Code, Library, API description, Tool, Documentation, Release Note...) of this release.

<table>
<thead>
<tr>
<th>Deliverable Name</th>
<th>Deliverable Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design &amp; Package Tool</td>
<td>Provider a graphical tool to define the VNF product model and package for vnf vendor engineer</td>
</tr>
<tr>
<td>Validate &amp; Lifecycle Test</td>
<td>Validate the package from vendor, and do the lifecycle test for it.</td>
</tr>
<tr>
<td>Function Test</td>
<td>Supply a test framework for vendor, so that before upload to Market place, provider can know the basic function is OK</td>
</tr>
<tr>
<td>Market Place</td>
<td>For publication, upload and download of VNF products and offerings</td>
</tr>
</tbody>
</table>

Sub-Components

See the repository table.

Architecture

Include High level architecture diagram.
API Dependencies

List the API this release is expecting from other releases. Prior to Release Planning review, Team Leads must agreed on the date by which the API will be fully defined. The API Delivery date must not be later than the release Functionality Freeze date.
Prior to the delivery date, it is a good practice to organize an API review.

**API Incoming Dependencies**

<table>
<thead>
<tr>
<th>API Name</th>
<th>API Description</th>
<th>API Definition Date</th>
<th>API Delivery date</th>
<th>API Definition link (i.e. swagger)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micro Service Bus API</td>
<td>API for registration of the microservices</td>
<td>Feb 16, 2017</td>
<td>Microservice_Bus_API_Documentation</td>
<td></td>
</tr>
<tr>
<td>NFVO Lifecycle API</td>
<td>API for test NFVO lifecycle and Validation</td>
<td>Feb 16, 2017</td>
<td>NFVO LCM API</td>
<td></td>
</tr>
<tr>
<td>Model Designer API</td>
<td>API for common tosca model designer</td>
<td>Feb 16, 2017</td>
<td>TBD</td>
<td></td>
</tr>
<tr>
<td>Function Test API</td>
<td>API for Function framework API</td>
<td>Feb 16, 2017</td>
<td>Function Test</td>
<td></td>
</tr>
</tbody>
</table>

**API Outgoing Dependencies**

API this release is delivering to other releases.

<table>
<thead>
<tr>
<th>API Name</th>
<th>API Description</th>
<th>API Definition Date</th>
<th>API Delivery date</th>
<th>API Definition link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download API</td>
<td>API for Operator download VNF Marketplace's Package</td>
<td>Jan 24, 2017</td>
<td>Feb 16, 2017</td>
<td>Market Place</td>
</tr>
<tr>
<td>Upload API</td>
<td>API for VNF vendor upload VNF Package to VNF Marketplace</td>
<td>Jan 24, 2017</td>
<td>Feb 16, 2017</td>
<td>Market Place</td>
</tr>
</tbody>
</table>

**Third Party Products Dependencies**

Currently, VNF SDK didn't have third party dependencies.

**Testing and Integration Plans**

*Function Test Plan and Resources*

Describe the plan to integrate and test the release deliverables with the overall OPEN-O system. Confirm that resources have been allocated to perform such activities. The Release lifecycle accounts for an [Testing and Integration Review](#) where detailed plan are expected.

This activity must be in sync with [Release Integration Review milestone (RC0)](#).

Projects may apply for a system test waiver if they think they have top-level features not requiring system test or covered by other top-level features test.

Projects must specify whether they plan to use Open-O CI infrastructure for system test. It is recommended to use the Open-O CI infrastructure unless there is some HW or SW resource that cannot be installed there. Projects running system test in external Labs are required to report system test results in a timely fashion after release creations, e.g., weekly, RC, and formal releases.

**Gaps**

This section is used to document a limitation on a functionality or platform support. We are currently aware of this limitation and it will be delivered in the future. List identified release gaps (if any), and its impact.

<table>
<thead>
<tr>
<th>Gaps identified</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>To fill out</td>
<td>To fill out</td>
</tr>
</tbody>
</table>

**Known Defects and Issues**

Provide a link toward the list of all known project defects.
Risks

List the risks identified for this release along with the plan to prevent the risk to occur (mitigation) and the plan of action in the case the risk would materialized (contingency).

<table>
<thead>
<tr>
<th>Risk identified</th>
<th>Mitigation Plan</th>
<th>Contingency Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

Resources

Resources assigned to project are consolidated in a single place. Edit all the table with the ALL information, that helps to expedite Linux Foundation operations.

Release Milestone

The milestones are defined at the Release Level and all the supporting project agreed to comply with these dates.

Team Internal Milestone

Project Plan

<table>
<thead>
<tr>
<th>Sub-Project</th>
<th>Owner</th>
<th>task</th>
<th>Participate</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUI Design&amp;Package Tool</td>
<td>zhaohuabing</td>
<td>Designer</td>
<td>committer <a href="mailto:victor.zhanganzhang@zte.com.cn">victor.zhanganzhang@zte.com.cn</a>, <a href="mailto:tian.jia@zte.com.cn">tian.jia@zte.com.cn</a>, <a href="mailto:zhao.huabing@zte.com.cn">zhao.huabing@zte.com.cn</a>, <a href="mailto:qin.lihan@zte.com.cn">qin.lihan@zte.com.cn</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VNF Model</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Krzysztof</td>
<td>Package Builder</td>
<td>Committer <a href="mailto:krzysztof.frukacz@gigaspaces.com">krzysztof.frukacz@gigaspaces.com</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Package Exactor</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Package Validator</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Package Parser</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Package Dry Run</td>
<td></td>
</tr>
<tr>
<td>Validate &amp; Lifecycle Test</td>
<td>zhanganbing</td>
<td>API between Validation and NFVO</td>
<td>committer <a href="mailto:gong.mingsheng@zte.com.cn">gong.mingsheng@zte.com.cn</a>, <a href="mailto:zhanganbing@chinamobile.com">zhanganbing@chinamobile.com</a></td>
</tr>
<tr>
<td>Function Test</td>
<td>Murali</td>
<td>Hook for reusing NFVO resource allocation operation API between LC Test and NFVO/VNFM Hook for reusing NFVO workflow</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>--------</td>
<td>----------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Market Place</td>
<td>Sukesh/Alex</td>
<td>Test Framework Selection &amp; guidance of framework Measurement for Function Test API between FT and VNFM/NFVO Function Test hook</td>
<td></td>
</tr>
</tbody>
</table>

Vendor Neutral

If this project is coming from an existing proprietary codebase, ensure that all proprietary trademarks, logos, product names, etc. have been removed. All OPEN-O deliverables must comply with this rule and be agnostic of any proprietary symbols.

Free and Open Source Software

FOSS activities are critical to the delivery of the whole OPEN-O initiative. The information may not be fully available at Release Planning, however to avoid late refactoring, it is critical to accomplish this task as early as possible.

List all third party Free and Open Source Software used within the release and provide License type (BSD, MIT, Apache, GNU GPL,... ). In the case non Apache License are found inform immediately the TSC and the Release Manager and document your reasoning on why you believe we can use a non Apache version 2 license.

Each project must edit its table within the FOSS VNF SDK

Documentation, Training, Tutorial

- High level list of documentation, training and tutorials necessary to understand the release capabilities, configuration and operation.
- Documentation includes items such as:
  - Installation instructions
  - Configuration instructions
  - Developer guide
  - End User guide
  - Admin guide

Note

It is expected all materials to be published directly either in the wiki or in OPEN-O website. Do not provide deliverable in PDF, powerpoint or word documents (these format are not easy to edit and lead to versioning issues).

Board policy (including IPR)

Indicate if the release meets the Board policy.
**VNF SDK Mercury Release Planning Deliverables (M1)**

The following items are expected to be completed for the project to **pass the M1 Release Planning Milestone**.

**M1 Release Planning Milestone definition.**

<table>
<thead>
<tr>
<th>Practice Area</th>
<th>Checkpoint</th>
<th>Yes/No</th>
<th>Evidence - Comment</th>
<th>How to?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Management</td>
<td>Are Product Backlog Epics entered in Jira?</td>
<td>Yes</td>
<td>12 issues</td>
<td>Create a Backlog item Difference between a Product and Sprint Backlog</td>
</tr>
<tr>
<td></td>
<td>Are Product Backlog Stories entered in Jira?</td>
<td>Yes</td>
<td>28 issues</td>
<td>Create a Backlog item</td>
</tr>
<tr>
<td></td>
<td>Are Product Backlog Stories linked to Product Backlog Epics?</td>
<td>Yes</td>
<td>VNF SDK Backlog</td>
<td>Work in a Sprint</td>
</tr>
<tr>
<td></td>
<td>Are Product Backlog Stories prioritized?</td>
<td>Yes</td>
<td></td>
<td>Prioritize a Backlog item</td>
</tr>
<tr>
<td></td>
<td>Is the project team ready to perform estimate for the top Story (for coming Sprint) in Product backlog?</td>
<td>Yes</td>
<td></td>
<td>Estimate a Backlog item</td>
</tr>
<tr>
<td></td>
<td>Is the project team ready to create a 2 weeks Sprint in Jira?</td>
<td>Yes</td>
<td></td>
<td>Create a Sprint</td>
</tr>
<tr>
<td></td>
<td>Is Team Member willing to create Tasks and associate them with Stories in Jira?</td>
<td>Yes</td>
<td></td>
<td>Create a Backlog item</td>
</tr>
<tr>
<td>Release Management</td>
<td>Is there a Release Planning Template available and completed in wiki?</td>
<td>Yes</td>
<td>VNF SDK Mercury Release Plan</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Have all the &quot;Release Components Name&quot; been defined in Resources and Repositories for your project? (this includes all Sub-Components Names, Sub-Components Repositories Names, Maven Group ID, Sub-Components Description)</td>
<td>Yes</td>
<td>Resources and Repositories#ReleaseComponentsName.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Have all the &quot;Resources committed to the Release&quot; been defined in Resources and Repositories for your project? This includes First and Last names, LFID, Email Address and Location for PTL, Project Manager, Committers and Contributors.</td>
<td>Yes</td>
<td>Resources and Repositories#ResourcescommittedtotheRelease.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Have new developers made themself familiar on the Onboarding Process?</td>
<td>Yes</td>
<td></td>
<td>Onboarding</td>
</tr>
<tr>
<td></td>
<td>Is the project team aware of the Release milestone? Any misses will required TSC exception.</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integration and Testing</td>
<td>Is the Project Team committed to automate Unit Test?</td>
<td>Yes</td>
<td>UT Coverage &gt;= 50%</td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------</td>
<td>-----</td>
<td>-------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is the Automated Unit Test infrastructure in place?</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is the project team committed to create Continuous System Integration Testing (CSIT) test case?</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development</td>
<td>Is the project team committed to perform Scrum ceremonies?</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Are the project team member aware of Continuous Integration Principles (don’t break the build, Fix the build,...)?</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Has the project team clear understanding on the Code Coverage expectations?</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Does the project team understand the Free and Open Source Software (FOSS) process?</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is the project team willing to fill out accordingly the FOSS table?</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is the project team aware of the Commit Process?</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Does the project team understand the purpose of Code Review?</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is the project team aware of the Coding Guidelines?</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VNF SDK Architecture and Design

This page and the child pages capture the overall VNF SDK Architecture and Design, including various artifacts such as Information Models, including static diagrams such as component, class, deployment, object and package diagrams and behavioral diagrams including use case, sequence, activity and state diagrams.

- VNF SDK Flow Diagram
- VNF SDK Use Case Definitions

VNF SDK Flow Diagram

High-Level Sequence Diagram Create Package
Relationship Between VNF-SDK and Other Project

Notes:
VNF-SDK Related tools can be delivered in two ways: Standalone, part of OPEN-O
User(Vendor/Operator) Perspective Flow

VNF Vendor Could use VNF-SDK Model Designer to generate VNF Package, also can done this manually.

Invalid/Fail

Notice:
- 1. VNF Package Model or other Model in VNF-SDK are discussed with Modeling project,
- 2. VNF-Package follow TOSCA CSAR format
- 3. VNF SDK Use Hook to invoke other project’s function, such as NFVO CLI

Design & Package Tool
- Designer
- VNF Model
- Package Builder
- Package Validator
- Package Extractor
- Package Report

OPEN-O R1 Model Designer
VNF SDK Use Case Definitions

The following use cases are work in progress.

- Mercury Use Cases
  - Detailed Use Case Definitions
    - Design VNF Package
    - Upload New VNF Package
    - Update Existing VNF Package
    - Manage Marketplace Users (Deferred)
  - Manage Marketplace VNF Package Content
    - Onboard VNF Package
    - Validate VNF Package
    - Test VNF Package Lifecycle
    - Test VNF Package Function
Mercury Use Cases

Use cases define how different users interact with a system under design. Each use case represents an action that may be performed by a user (defined in UML as an Actor with a user persona).

Detailed Use Case Definitions

This section defines each use case in greater details.

Design VNF Package

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>VNF-SDK-UC-01</td>
</tr>
<tr>
<td>Title</td>
<td>Design VNF Package</td>
</tr>
<tr>
<td>Description</td>
<td>In this use case, the VNF supplier uses the TOSCA Common Toolkit or the VNF SDK to design and create a new VNF package which performs a specified network function.</td>
</tr>
<tr>
<td>Primary Actor</td>
<td>VNF Supplier</td>
</tr>
<tr>
<td>Preconditions</td>
<td></td>
</tr>
<tr>
<td>Main Success Scenario</td>
<td></td>
</tr>
</tbody>
</table>
Upload New VNF Package

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>VNF-SDK-UC-02</td>
</tr>
<tr>
<td>Title</td>
<td>Upload New VNF Package</td>
</tr>
<tr>
<td>Description</td>
<td>In this use case, the VNF Supplier uploads the market-ready VNF package to the marketplace staging area.</td>
</tr>
<tr>
<td>Primary Actor</td>
<td>VNF Supplier</td>
</tr>
<tr>
<td>Preconditions</td>
<td></td>
</tr>
<tr>
<td>Main Success Scenario</td>
<td></td>
</tr>
<tr>
<td>Alternate Scenarios</td>
<td></td>
</tr>
<tr>
<td>Exception Scenarios</td>
<td></td>
</tr>
<tr>
<td>Post Conditions</td>
<td>VNF package resides in the staging area of the marketplace and Marketplace Operator has been notified</td>
</tr>
</tbody>
</table>

Update Existing VNF Package

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>VNF-SDK-UC-03</td>
</tr>
<tr>
<td>Title</td>
<td>Update Existing VNF Package</td>
</tr>
<tr>
<td>Description</td>
<td>In this use case, the VNF Supplier needs to update a VNF package that has already been published, perhaps fixing a defect in the existing package, or adding a new feature to the package.</td>
</tr>
<tr>
<td>Primary Actor</td>
<td>VNF Supplier</td>
</tr>
<tr>
<td>Preconditions</td>
<td></td>
</tr>
<tr>
<td>Main Success Scenario</td>
<td></td>
</tr>
<tr>
<td>Alternate Scenarios</td>
<td></td>
</tr>
<tr>
<td>Exception Scenarios</td>
<td></td>
</tr>
<tr>
<td>Post Conditions</td>
<td>The updated VNF package resides in the staging area of the marketplace and Marketplace Operator has been notified</td>
</tr>
</tbody>
</table>

Manage Marketplace Users (Deferred)

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>VNF-SDK-UC-04</td>
</tr>
<tr>
<td>Title</td>
<td>Manage Marketplace Users</td>
</tr>
</tbody>
</table>
Manage Marketplace VNF Package Content

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>VNF-SDK-UC-05</td>
</tr>
<tr>
<td>Title</td>
<td>Manage Marketplace VNF Package Content</td>
</tr>
<tr>
<td>Description</td>
<td>In this Use Case, the Marketplace Operator has the responsibility of managing all VNF Package Content in the marketplace.</td>
</tr>
<tr>
<td>Primary Actor</td>
<td>Marketplace Operator</td>
</tr>
</tbody>
</table>

Onboard VNF Package

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>VNF-SDK-UC-05-01</td>
</tr>
<tr>
<td>Title</td>
<td>Onboard VNF Package</td>
</tr>
<tr>
<td>Description</td>
<td>In this Use Case, the Marketplace Operator onboards a VNF package placed in the staging area (e.g., newly uploaded package).</td>
</tr>
<tr>
<td>Primary Actor</td>
<td>Marketplace Operator</td>
</tr>
</tbody>
</table>

Validate VNF Package

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>VNF-SDK-UC-05-01-01</td>
</tr>
<tr>
<td>Title</td>
<td>Validate VNF Package</td>
</tr>
</tbody>
</table>
In this Use Case, the Marketplace Operator validates the VNF package to ensure it complies with package rules.

**Primary Actor**
Marketplace Operator

**Preconditions**

**Main Success Scenario**

**Alternate Scenarios**

**Exception Scenarios**

**Post Conditions**
The VNF package is validated.

---

### Test VNF Package Lifecycle

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>VNF-SDK-UC-05-01-02</td>
</tr>
<tr>
<td>Title</td>
<td>Test VNF Package Lifecycle</td>
</tr>
<tr>
<td>Description</td>
<td>In this Use Case, the Marketplace Operator tests the end-to-end lifecycle of the VNF package.</td>
</tr>
<tr>
<td>Primary Actor</td>
<td>Marketplace Operator</td>
</tr>
<tr>
<td>Preconditions</td>
<td></td>
</tr>
</tbody>
</table>

**Main Success Scenario**

**Alternate Scenarios**

**Exception Scenarios**

**Post Conditions**
The VNF Package passed the lifecycle testing.

### Test VNF Package Function

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>VNF-SDK-UC-05-01-03</td>
</tr>
<tr>
<td>Title</td>
<td>Test VNF Package Function</td>
</tr>
<tr>
<td>Description</td>
<td>In this Use Case, the Marketplace Operator tests the network function of the VNF package.</td>
</tr>
<tr>
<td>Primary Actor</td>
<td>Marketplace Operator</td>
</tr>
<tr>
<td>Preconditions</td>
<td></td>
</tr>
</tbody>
</table>

**Main Success Scenario**

**Alternate Scenarios**

**Exception Scenarios**

**Post Conditions**
The VNF Package passed the function testing.

### Download New VNF Package

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>VNF-SDK-UC-06</td>
</tr>
<tr>
<td>Title</td>
<td>Download New VNF Package</td>
</tr>
<tr>
<td>Description</td>
<td>In this Use Case, the Service Provider searches, finds and downloads a VNF package that meets their network service needs.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Primary Actor</th>
<th>Service Provider</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preconditions</td>
<td></td>
</tr>
<tr>
<td>Main Success Scenario</td>
<td></td>
</tr>
<tr>
<td>Alternate Scenarios</td>
<td></td>
</tr>
<tr>
<td>Exception Scenarios</td>
<td></td>
</tr>
<tr>
<td>Post Conditions</td>
<td>VNF SDK package successfully downloaded by Service Provider</td>
</tr>
</tbody>
</table>

**Receive Update Alerts (Deferred)**

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>VNF-SDK-UC-07</td>
</tr>
<tr>
<td>Title</td>
<td>Receive Update Alerts</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Primary Actor</td>
<td>Service Provider</td>
</tr>
<tr>
<td>Preconditions</td>
<td></td>
</tr>
<tr>
<td>Main Success Scenario</td>
<td></td>
</tr>
<tr>
<td>Alternate Scenarios</td>
<td></td>
</tr>
<tr>
<td>Exception Scenarios</td>
<td></td>
</tr>
<tr>
<td>Post Conditions</td>
<td></td>
</tr>
</tbody>
</table>

**Download Updated VNF Package (Deferred)**

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>VNF-SDK-UC-08</td>
</tr>
<tr>
<td>Title</td>
<td>Download Updated VNF Package</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Primary Actor</td>
<td>Service Provider</td>
</tr>
<tr>
<td>Preconditions</td>
<td></td>
</tr>
<tr>
<td>Main Success Scenario</td>
<td></td>
</tr>
<tr>
<td>Alternate Scenarios</td>
<td></td>
</tr>
<tr>
<td>Exception Scenarios</td>
<td></td>
</tr>
<tr>
<td>Post Conditions</td>
<td></td>
</tr>
</tbody>
</table>

**Use Case Personas**

This section describes the Use Case personas or UML Actors.

**VNF Supplier**

The VNF Supplier has responsibility for designing, defining and developing a VNF package which Service Provider has a need for.
Marketplace Operator

The Marketplace Operator has responsibility for managing and controlling the marketplace environment, including managing users and VNF content placed in the marketplace.

Service Provider

The Service Provider is the consumer of the VNF packages in the marketplace. A particular VNF is downloaded to meet a business need the Service Provider has.

VNF SDK Blueprint

This page presents our work in progress and discussion towards defining a TOSCA "template" for VNF vendors to use when bundling their VNFs.

- 1 History
- 2 Files
- 3 Reading the Blueprint
- 4 Working with the Blueprint
- 5 Current Limitations
  - 5.1 Extensions
- 6 Combining the Worlds of VNF and Compute
- 7 Challenge of Infrastructure Optimization
- 8 Compute Dependencies
  - 8.1 Compute Dependencies
  - 8.2 Compute Profile Definitions
- 9 Telemetry
- 10 Integration with Service Chains
- 11 See Also

History

Draft 1: Nov 24 2016. Opening the discussion. Based on input from Krzysztof Frukacz, Tal Liron, Alex Vul, Michael Brenner, Nati Shalom.


Draft 3: Dec 8 2016. Added KPI and simple Ping Pong example.

Draft 4: 4 Jan 2016. Added analysis document for EPA in Openstack vs VNF profile

Draft 5: 24 Jan 2017 Updated the definition

Draft 6 10 Feb 2017 Removed references to old version. Updated description

Files

- pingpong.yaml - Simple example VNF blueprint.
- vrouter.yaml - Fuller example blueprint for a complete virtual router VNF.
- vnfsdk-profile-1.0.yaml - Imported types for the above profiles.
- compute-profiles.yaml - Groups dependencies together as “compute profiles”.

Guiding Principles

1. **Adhere** to TOSCA standard and its simple profile for NFV
   a. By continuing to iterate, we will be able to identify the pain points and advise the OASIS community on fixing and enhancing the profile
2. **Ensure** interoperability with greater community
   a. TOSCA is open-ended, so our challenge is to make sure that Open-O’s VNF blueprints could be consumed by other MANO implementations
3. **Expose** infrastructure optimizations
   a. Hardware platform (e.g. Intel EPA)
   b. Consideration for containers (in addition to virtual machines and bare metal)
   c. Entry points for quality-assurance telemetry
Reading the Blueprint

We try to make the blueprint as humanly readable as possible, but until there is a good TOSCA IDE that can show you inheritance and do live validation (ARIA is designed to support one in the future), you will have to look up the node/capability/relationship types yourself.

Please use the standard profiles as reference while you read.

Working with the Blueprint

We recommend that you make sure that the blueprint is valid as you edit it. You can use ARIA to test your blueprint and see the final deployment plan.

Install requirements on Ubuntu:

```
sudo apt install python-setuptools
sudo -H easy_install pip
sudo -H pip install virtualenv
```

Install ARIA:

```
virtualenv env
. env/bin/activate
pip install git+http://git-wip-us.apache.org/repos/asf/incubator-ariatosca.git
```

Run ARIA parser:

```
aria parse vrouter.yaml
```

Current Limitations

TOSCA currently has two profiles: `tosca_simple_yam_1.0` and `tosca_simple_profile_for_nfv_1.0`, which extends the former.

But they are too open ended. For example, there are basic policy types (Scaling, Placement, Performance) but they are empty, waiting to be extended by actual implementations.

Unfortunately, for VNF vendors this means that there is no reference implementation for them to target, and that work would have to be duplicated for each implementation.

An analysis provided here contains a few example EPA features that are supported by Openstack. A lot of them are not covered by TOSCA simple profile for NFV v1.0.

Extensions

We are suggesting that extensions to the profile are necessary to create a shared common ground between VNF vendor and operator. This `tosca_vnfsdk_profile_1.0` extends on the two other profiles and introduces more specific types (policies and relationships), while still leaving room for implementations to provide their own algorithms and systems and compete.

Combining the Worlds of VNF and Compute

As it stands, the TOSCA NFV profile CSD03 types are mostly isolated from the basic profile types. Specifically, `tosca.nodes.nfv.VDU` duplicates the logic of `tosca.nodes.Compute`. It is impossible to relate them as is, because `Compute`'s host capability only allows for `tosca.nodes.SoftwareImage`
eComponent sources, and VDU does not derive from SoftwareComponent.

Without changing the NFV profile for now, we are suggesting a workaround: use the dependency requirement, which all nodes support, but extend tosca.relationships.DepsOn to allow us to configure the relationship between the VDU and Compute nodes. This allows us to reuse types and mechanisms defined in TOSCA simple YAML profile with our NFV profile nodes, like VDU.

We will expand on this example relationship later on:

```yaml
node_templates:
    router_vdu:
        type: tosca.nodes.nfv.VDU
        requirements:
            - dependency:
                node: router_host
                relationship:
                    type: vnfsdk.DependsOn
    router_host:
        type: tosca.nodes.Compute
        capabilities:
            scalable:
              properties:
                max_instances: 4
```

**Challenge of Infrastructure Optimization**

We believe that tosca.capabilities.Compute.Container.Architecture as defined in the TOSCA NFV profile is entirely insufficient for VNF vendors.

However, we also believe that extending this basic type with a large “shopping list” of properties would be worse, as:

1. we cannot anticipate future (and even near-future) technology
2. the current state of resource orchestration (there is none) means that shopping lists could not be fulfilled to satisfaction.

**Compute Dependencies**

Instead, we suggest a three-way approach:

1. Support the TOSCA NFV properties as is, as a lowest-common denominator of absolute requirements for the VDU, and recommend that this feature be deprecated.
2. Allow VNF vendors to specify arbitrary dependencies for the algorithm via an extensible spec that allows for future technological innovation.
3. Group features together as “compute profiles” (e.g. “high_availability”, “dpdk_enabled”, “ovs_ready”) and use TOSCA policies to map the VNF vendor’s annotations to profiles using an algorithm. These performance profiles will be configured by the operator rather than the VNF vendor.
ping_vdu:
  description: >-
   The "ping" VDU generates traffic.
  type: tosca.nodes.nfv.VDU
  interfaces:
   Standard:
    configure: scripts/vdu/ping_configure.sh # included in the CSAR
  requirements:
   - dependency:
     node: ping_host # our Compute node
   relationship:
     type: vnfsdk.DependsOn
   properties:
     # For the "ping" VDU we are choosing various compute dependencies:
   dependencies:
    compute_dependencies:
     cpu_allocation:
      string_map:
       cpu_affinity: { get_input: pong.cpu_affinity }
       thread_allocation: isolate
     container:
      boolean: false
     dpdk:
      string: '16.07'

Compute Profile Definitions

We suggest that Compute Profiles can be defined by the operator using TOSCA to create a "bucket" of compute dependencies. The profiles are suppose to simplify the definitions of Compute Dependencies, as in a lot of deployments there are many VDUs with the same required EPA features running on the same infrastructure hardware. If a Compute Profile is not used, then all Compute Dependencies have to specified for a given VDU.

dpdk:
  description: >-
   DPDK compute profile.
  compute_dependencies:
   dpdk:
    string: { get_input: dpdk.version }
   mem_page_size:
    string: large
   optional: true
   sr-iov:
    boolean: true
   cpu_allocation:
    string_map:
     cpu_affinity: dedicated

Below is an example of ping_vdu that specifies a pre-defined profile instead of specifying every Compute Dependency explicitly.
ping_vdu:
  description: >-
      The "ping" VDU generates traffic.
  type: tosca.nodes.nfv.VDU
  interfaces:
    Standard:
      configure: scripts/vdu/ping_configure.sh # included in the CSAR
  requirements:
    - dependency:
        node: ping_host # our Compute node
        relationship:
          type: vnfsdk.DependsOn
          properties:
            # For the "ping" VDU we are choosing a profile that already
            includes
            # various compute dependencies:
            compute_profile: dpdk

Telemetry

Our goal here is neither comprehensive FCAPS nor to step into the domain of NETCONF and YANG, but rather to allow VNF vendors to define arbitrary indicators for measuring acceptable and good performance.

ping_vdu:
  description: >-
      The "ping" VDU generates traffic.
  type: tosca.nodes.nfv.VDU
  interfaces:
    Standard:
      configure: scripts/vdu/ping_configure.sh # included in the CSAR
  requirements:
    - dependency:
        node: ping_host # our Compute node
        relationship:
          type: vnfsdk.DependsOn
          properties:
            # We'll expose two indicators from "ping" that can be used
            by monitoring tools:
            indicators:
              hits:
                type: int
                implementation: scripts/vdu/ping_get_hits.sh # included
                in the CSAR
              connectivity:
                type: string
                implementation: { get_input: ping.connectivity.indicator }
Integration with Service Chains

The VNF’s CSAR will need to be integrated into service chains. We will use TOSCA substitution mapping to treat the entire blueprint as one logical VNF, specifically mapping our internal connection to the consuming blueprint:

```yaml
substitution_mappings:
  node_type: tosca.nodes.nfv.VNF
  requirements:
    virtual_link: [ ingress_route, virtual_link ]
```

See Also

- Tacker vnfd
- T-Nova VNF

VNF SDK - Guide for Bundling VNFs

**DRAFT**

Step 1: Requirements

1. Virtual machine images for each of your VDUs (likely in qcow2 format).
2. Scripts for installing and configuring each VDU per the standard TOSCA lifecycle workflows.
3. Optional: scripts for retrieving indicators from each VDU.
4. The VNF SDK.

Step 2: Start with the provided VNF blueprint as a template

The VNF SDK comes with a "pingpong.yaml" template.

Copy the file and rename it as is appropriate for your VNF.

Step 3: Edit the VNF blueprint

You will need a node template for each VDU, which in turn has a "dependency" requirement on a host (a Compute node template).

The "dependency" requirement should have a relationship of type "vnfsdk.DependsOn".

There are two optional properties you can assign to this relationship:

1. "indicators" is a map of indicators. The keys are arbitrary, for you to name, and depend on the specific capabilities of your VNF. These provide the link between TOSCA and your indicator retrieval scripts.
2. "compute_dependencies" is a map of features you require the infrastructure to provide for the compute node hosting your VDU. The keys are arbitrary, however we will maintain a master list of names supported by various orchestrators (TBD). Note that new names will likely be supported in the future as technological innovation progresses, while deprecated ones will be ignored by orchestrators. Likewise, names unsupported by a particular orchestrator will be ignored.

Notes:

1. For "compute_dependencies", you have the option of specifying a single special key, "profile", that refers to a bundle of compute dependencies specified in "compute-profiles.yaml". Optionally, you can override any of the dependencies in the bundle by specifying them explicitly under "compute_dependencies".
2. By default, compute dependencies are considered to be absolute requirements for your VNF to function. However, if your VNF can also function without them, you can set "optional: true". This way the orchestrator will try to satisfy the optional requirements, but will not report an error if it cannot.
3. You may optionally consider allowing operators to override your dependencies. This is useful for allowing the VNF to run on various compute infrastructures other than the defaults you specify, as long as they are indeed supported by your VNF. In TOSCA, this is easily
handled by declaring "inputs" for your blueprint, with "default" values, and then using the "get_input" intrinsic function to apply the provided value to any specific "compute_dependencies". The "pingpong.yaml" has an example of such usage. The same technique can be used to allow the operator to override your indicator scripts, and there is an example of that as well.

Otherwise, this is a standard TOSCA blueprint using the Simple Profile for NFV, and you may extend the blueprint as is required.

**Step 4: Bundle the VNF**

Arrange all your files (the TOSCA template, your scripts, and your virtual machine images) in the correct directory structure. File references provided in the blueprint must match the directory structure. If a script is referenced using a path with sub-directory "example" the initial directory must contain a directory called "example" with that script file inside. The directory structure will be kept inside the CSAR archive.

Run the bundling tool that is included in the VNF SDK.

The result will be a ".csar" file that can now be deployed by operators.

---

**VNF SDK Documentation**

Filter By File Extension:  Filter By Label:

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Creator</th>
<th>Creation Date</th>
<th>Last Modification Date</th>
<th>Labels</th>
<th>Attached To</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="#">VNF SDK Use Case Diagram.png</a></td>
<td>98 kB</td>
<td>Brian Hedstrom</td>
<td>Mar 02, 2017</td>
<td>Mar 02, 2017</td>
<td>No labels</td>
<td>VNF SDK Use Case Definitions</td>
</tr>
<tr>
<td><a href="#">Relationship between VNF-SDK and other project.png</a></td>
<td>34 kB</td>
<td>gao weitao</td>
<td>Mar 02, 2017</td>
<td>Mar 02, 2017</td>
<td>No labels</td>
<td>VNF SDK Flow Diagram</td>
</tr>
<tr>
<td><a href="#">marketplace ut.png</a></td>
<td>17 kB</td>
<td>gao weitao</td>
<td>Feb 22, 2017</td>
<td>Feb 22, 2017</td>
<td>No labels</td>
<td>VNF SDK Mercury Release Architecture Milestone Deliverables (M3)</td>
</tr>
<tr>
<td><a href="#">vnf_sdk_highestlevel_sequence_diagram_new.png</a></td>
<td>40 kB</td>
<td>gao weitao</td>
<td>Feb 21, 2017</td>
<td>Feb 21, 2017</td>
<td>No labels</td>
<td>VNF SDK Flow Diagram</td>
</tr>
<tr>
<td><a href="#">vnf_sdk_highestlevel_sequence_diagram_new.png</a></td>
<td>40 kB</td>
<td>gao weitao</td>
<td>Feb 21, 2017</td>
<td>Feb 21, 2017</td>
<td>No labels</td>
<td>VNF SDK Flow Diagram</td>
</tr>
<tr>
<td><a href="#">Open-O VNF SDK</a></td>
<td>75 kB</td>
<td>Brian Hedstrom</td>
<td>Feb 16, 2017</td>
<td>Feb 16, 2017</td>
<td>No labels</td>
<td>VNF SDK Documentation</td>
</tr>
</tbody>
</table>
VNF SDK Meetings

- Logistics
- VNF SDK Meetings Minutes
  - December 1st, 2016: Meeting Notes
  - December 14th, 2016: Meeting Notes (TBD)
  - January 20, 2017 Meeting Notes
  - February 9, 2017 Meeting Notes

Logistics
VNF SDK Meetings Minutes

December 1st, 2016: Meeting Notes

Participants: Gigaspace, Intel, Huawei, ZTE

Agenda

- Gigaspaces introduce the details info about VNF-Onboarding (like HW awareness, container, etc)
- Alex/Huabing suggest we should decompose each model into more detail (like microservice level)
- Tal want more guys add comment on gigaspaces blueprint
- We talk about detail of VNF-SDK repo
- Marketplace:
  1. Alex suggest Canonical contribute this?
  2. Alex said marketplace should automatically download package and upload package, meanwhile do the VNF on-boarding

December 14th, 2016: Meeting Notes (TBD)

Agenda:

- We discuss the detail of VNF-SDK
- See detail meeting notes and decision at here

Join from PC, Mac, Linux, iOS or Android: https://huawei.zoom.us/j/817298236

Or iPhone one-tap (US Toll): +14086380968,817298236# or +16465588656,817298236#

Or Telephone:
  Dial: +1 408 638 0968 (US Toll) or +1 646 558 8656 (US Toll)
  Meeting ID: 817 298 236
International numbers available: https://huawei.zoom.us/zoomconference?m=fPXGPQ61eAZ0QYI4n9jzhbNlh57Ti7kq

**January 20, 2017 Meeting Notes**

**Agenda:**
1. Discussion on VNF-SDK Design & Package Tool  Huabing/Krzysztof (40min)
2. Sub Project status update  Sub-project owner (20min)

**February 9, 2017 Meeting Notes**

**Agenda:**
1. VNF Package (CSAR) Spec Discussion  Victor Gao(20min)
2. VNF SDK sub-project status update  sub PTL

**February 9, 2017 VNF SDK Meeting Minutes**

**January 20, 2017 VNF SDK Meeting Minutes**

**Participate: CMCC, Intel, Huawei, ZTE**

**Agenda**
1. Huabing give his understanding of Design and package tool
2. Discussion on design & package tool's GUI & CLI
3. Discussion on dry run functionality
4. Discussion on VNF-SDK weekly meeting time
   a. Since VNF-SDK time is too late for China team, we decide have the weekly meeting on China Time morning and PST time morning in turns.

**Action item:**
1. Alex & Krzysztof update the detail info about package tool (dry run, package validator, package parser)
2. Victor arrange a time to discuss Market place with Alex and Sukesh
3. Victor arrange a time for VNF-SDK weekly meeting on Morning (China Time)

**Meeting Materials**

<table>
<thead>
<tr>
<th>File</th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suggestion for some argues in VNF SDK Project(2017.01.24).pptx</td>
<td>Jan 24, 2017 by gao weitao</td>
</tr>
<tr>
<td>Jan 24 TSC topics for VNF-SDK</td>
<td></td>
</tr>
</tbody>
</table>
VNF SDK Old Home Page

- Project Description
- Project Status
- Participants
  - VNF SDK Blueprint
- Sub Project List
  - Format Transform
  - Validate
  - Lifecycle Test
  - Function test
  - Marketplace
- Team Meeting Calendar
- VNF SDK Functional Components & Interfaces
  - Functional Components
  - Interfaces
- VNF SDK Contributions (PROPOSED)
- Project Plan
  - OPEN-O VNF SDK Design(11.15)
  - Mercury Plan 2016-12-14
- Recent space activity

Project Description

The "VNF Supplier APIs and SDK" (short name: VNF SDK) project delivers code, documentation and best practices for the following functionality:

- Functionality for automated delivery and publication of VNF products with explicit VNFI and VIM dependencies in a supplier/provider neutral and standardized way
- Functionality to enable automated download and CI/CD of VNF products in a supplier/provider neutral and standardized way
- Hooks for automated KPI specification, telemetry generation and VNF performance testing
- Functionality to enable a rich OPEN-O community ecosystem of VNF product suppliers and operators, with high degree of interoperability

This functionality is intended for use by VNF product developers, VNF product DevOps teams, as well as operators of OPEN-O managed infrastructure. As a stretch goal, this project will deliver authoring tools to ease the creation of TOSCA blueprints by VNF developers.

Project Status

Approved on August 31, 2016

Participants

- CMCC
- Intel
- GigaSpaces
- Huawei
- ZTE
- Canonical

Team Meeting Calendar

VNF SDK Functional Components & Interfaces

The following diagram illustrates the VNF SDK functional components and their public interfaces. Additional information about each project can be found by following the links below.

Functional Components

- VNF Product Model/Blueprint

  VNF product model/blueprint provides a declarative way to define deployment, operational and functional attributes of a VNF product. The VNF product is defined in terms of deployment time requirements and dependencies and exposed telemetry indicator definitions.

  The deployment time requirements and dependencies define any and all compute infrastructure needs of the VNF product, such as specific hardware architecture, on-chip features, instruction set availability and hypervisor capabilities.

  The telemetry indicator definitions define a set of default indicators exposed by a given VNF product for use by monitoring and assurance tools. This list can be extended and customized once a given VNF product is on-boarded and instantiated at run-time.

  The VNF product model is specified using the TOSCA NFV simple profile. It is persisted, along with the product executables and data, using TOSCA CSAR files.
VNF SDK Blueprint

Sub Project List

Format Transform

- **Suggestion**
  Murali: Whether we need to have a GUI to automatically generate VNF/NSD packages similar to OSM? Is this being considered at this stage of project? (https://riftio.com/vnf-package-generator/)

- **Contributor/Committer:**
  - Dan Kilman - LFID dankilman - GigaSpaces
  - Nir Biran - LFID nirb - GigaSpaces
  - Alex Vul - LFID avul - Intel
  - Marcus Williams - LFID mgkwill - Intel

Validate

- **Suggestion**

- **Contributor/Committer:**
  - Tal Liron - LFID tal.liron - GigaSpaces
  - Avia Efrat - LFID avia - GigaSpaces
  - Ran Ziv - LFID ranZiv - GigaSpaces
  - Krzysztof Fruckacz - LFID - kfruckacz-gs
  - Alex Vul - LFID avul - Intel
  - Marcus Williams - LFID mgkwill - Intel

Lifecycle Test

- **Suggestion**

- **Contributor/Committer:**
  - Maxim Orlov - LFID mxmlv - GigaSpaces
  - Alex Vul - LFID avul - Intel
  - Marcus Williams - LFID mgkwill - Intel

Function test

- **Suggestion**
  Amir Levy: Suggest to focus on func-test interfaces - and leave the actual func-test to VNF's that specialized

- **Suggestion**
  VNF Product Package Designer
  VNF Package Designer, provides VNF product DevOps engineers with a graphical tool to define the VNF product model and package content. It’s made available as part of the VNF Supplier SDK tools. The package designer makes use of the VNF SDK command line interfaces (CLIs) and client-side API language bindings in order to define the model and the package content. As such, it is functionally equivalent to the VNF SDK tools.

- **Suggestion**
  VNF SDK Tools
  VNF SDK tools provide VNF product DevOps engineers with command line tools and client side API language bindings to define the VNF product model and package content. The following tools are included...
  - VNF Package Builder - creates a CSAR file based on inputs provided by the VNF product DevOps engineer
  - VNF Package Validator - validates the content of the VNF packages to ensure that everything has been built correctly
  - VNF Package Extractor - extracts VNF product model and executables from the CSAR file
  - VNF Package Parser - translates VNF product blueprint into a format consumable by OPEN-O components
  - VNF Package Dry Run - performs a "dry run" install to ensure that the package can be deployed during instantiation

- **Suggestion**
  NF Marketplace

- **Suggestion**
  OPEN-O Marketplace Tests

- **Suggestion**
  OPEN-O VNF SDK Microservice

- **Suggestion**
  OPEN-O NFV-O workflow extensions (in collaboration with the NFV-O project)

Interfaces

- **Suggestion**
  Sdk-Md
  Exposed by the VNF SDK as commands and client side language bindings. Enables use of VNF SDK functions from graphical user interfaces, shell scripts, programs and command line interfaces. Consumed by the VNF Package Designer.

- **Suggestion**
  Sdk-Mp
  Exposed by the VNF SDK as commands and client side language bindings. Enables use of NF Marketplace functions from graphical user interfaces, shell scripts, programs and command line interfaces. Consumed by the VNF Package Designer and OPEN-O VNF SDK micro-service adapter.

- **Suggestion**
  Mp-Api
  Exposed by the NF Marketplace as a programmatic API. Enables use of NF marketplace functions from command and graphical user interfaces. Consumed by the marketplace portal.

- **Suggestion**
  Mp-Tf
  Exposed by the NF Marketplace as a programmatic API. Enables use of the NF Marketplace’s PnP test framework for validation of uploaded VNF products. Consumed by the OPEN-O functional tests.
Suggestion

Artur Tyloch: For functest we should consider ABOT framework. It is open source VNF functional testing framework maintained by Rebaca https://jujucharms.com/u/deban-ch/abot-ims-basic/trusty/35

Contributor/Committer:

Marketplace

- Suggestion:
- Contributor/committer:

VNF SDK Contributions (PROPOSED)

The following diagram illustrates the proposal for contribution of the VNF SDK components.

Project Plan

Error rendering macro 'viewxls' : The viewfile macro is unable to locate the attachment "VNF-SDK Plan V0.1 .xlsx" on this page

OPEN-O VNF SDK Design(11.15)

Error rendering macro 'viewppt' : The viewfile macro is unable to locate the attachment "OPEN-O_VNF_SDK_Design(11.15).pptx" on this page

Mercury Plan 2016-12-14

Error rendering macro 'viewppt' : The viewfile macro is unable to locate the attachment "OPEN-O VNF-SDK Planning in Mercury Release(2106.12.15).pptx" on this page

VNF Design Tool 2016-1-22

Error rendering macro 'viewppt' : The viewfile macro is unable to locate the attachment "VNF SDK Design&Package Tool.pptx" on this page

Recent space activity

- gao weitao
  - Validate & Lifecycle Test updated Apr 28, 2017 • view change
VNF SDK Project Proposal

1 Project Description
2 Project Status
3 Architecture
   3.1 ARC Overview
   3.2 Flow Diagram
   3.3 Open-O-VNF-Supplier-v5
   3.4 OPEN-O_VNF_Supplier_APIs&_SDK_Project_Proposal(2016116)
   3.5 References

Project Description

The "VNF Supplier APIs and SDK" (short name: VNF SDK) project delivers code, documentation and best practices for the following functionality:

- Functionality for automated delivery and publication of VNF products with explicit VNFI and VIM dependencies in a supplier/provider neutral and standardized way
- Functionality to enable automated download and CI/CD of VNF products in a supplier/provider neutral and standardized way
- Hooks for automated KPI specification, telemetry generation and VNF performance testing
- Functionality to enable a rich OPEN-O community ecosystem of VNF product suppliers and operators, with high degree of interoperability

This functionality is intended for use by VNF product developers, VNF product DevOps teams, as well as operators of OPEN-O managed infrastructure. As a stretch goal, this project will deliver authoring tools to ease the creation of TOSCA blueprints by VNF developers.

Project Status

Approved on August 31, 2016

Architecture

ARC Overview
Flow Diagram

You can find flow diagram here.

Open-O-VNF-Supplier-v5
References

VNF SDK References

VNF SDK Functional Components & Interfaces

The following diagram illustrates the VNF SDK functional components and their public interfaces. Additional information about each project can be found by following the links below.

Functional Components

VNF Product Model/Blueprint

VNF product model/blueprint provides a declarative way to define deployment, operational and functional attributes of a VNF product. The VNF product is defined in terms of deployment time requirements and dependencies and exposed telemetry indicator definitions.

The deployment time requirements and dependencies define any and all compute infrastructure needs of the VNF product, such as specific hardware architecture, on-chip features, instruction set availability and hypervisor capabilities.

The telemetry indicator definitions define a set of default indicators exposed by a given VNF product for use by monitoring and assurance
tools. This list can be extended and customized once a given VNF product is on-boarded and instantiated at run-time.

The VNF product model is specified using the TOSCA NFV simple profile. It is persisted, along with the product executables and data, using TOSCA CSAR files.

**VNF Product Package Designer**

VNF Package Designer, provides VNF product DevOps engineers with a graphical tool to define the VNF product model and package content. It is made available as part of the VNF Supplier SDK tools. The package designer makes use of the VNF SDK command line interfaces (CLIs) and client-side API language bindings in order to define the model and the package content. As such, it is functionally equivalent to the VNF SDK tools.

**VNF SDK Tools**

VNF SDK tools provide VNF product DevOps engineers with command line tools and client side API language bindings to define the VNF product model and package content. The following tools are included...

- VNF Package Builder - creates a CSAR file based on inputs provided by the VNF product DevOps engineer
- VNF Package Validator - validates the content of the VNF packages to ensure that everything has been built correctly
- VNF Package Extractor - extracts VNF product model and executables from the CSAR file
- VNF Package Parser - translates VNF product blueprint into a format consumable by OPEN-O components
- VNF Package Dry Run - performs a “dry run” install to ensure that the package can be deployed during instantiation

**NF Marketplace**

TBD

**OPEN-O Marketplace Tests**

TBD

**OPEN-O VNF SDK Microservice**

TBD

**OPEN-O NFV-O workflow extensions (in collaboration with the NFV-O project)**

TBD

**Interfaces**

**Sdk-Md**

Exposed by the VNF SDK as commands and client side language bindings. Enables use of VNF SDK functions from graphical user interfaces, shell scripts, programs and command line interfaces. Consumed by the VNF Package Designer.

**Sdk-Mp**

Exposed by the VNF SDK as commands and client side language bindings. Enables use of NF Marketplace functions from graphical user interfaces, shell scripts, programs and command line interfaces. Consumed by the VNF Package Designer and OPEN-O VNF SDK micro-service adapter.

**Mp-Api**

Exposed by the NF Marketplace as a programmatic API. Enables use of NF marketplace functions from command and graphical user interfaces. Consumed by the marketplace portal.

**Mp-Tf**

Exposed by the NF Marketplace as a programmatic API. Enables use of the NF Marketplace’s PnP test framework for validation of uploaded VNF products. Consumed by the OPEN-O functional tests.
VNF SDK Contributions (PROPOSED)

The following diagram illustrates the proposal for contribution of the VNF SDK components.

VNF SDK Sub Projects
VNF SDK Flow Diagram

This is draw by visio 2007, you can find the source file in attachment *.vsd.

Design&Package Tool

Project structure

- VNF Package Designer
- VNF SDK tools
  - VNF Model: VNF SDK Blueprint
  - VNF Package Builder
  - VNF Package Validator
  - VNF Package Extractor
  - VNF Package Parser
  - VNF Package Dry Run

VNF SDK tools

VNF Model

The model information can be found under the following link: VNF SDK Blueprint

VNF Package tools

Provided tools

VNF Package Builder - creates a CSAR file based on inputs provided by the VNF product DevOps engineer
VNF Package Validator - validates the content of the VNF packages to ensure that everything has been built correctly
VNF Package Extractor - extracts VNF product model and executables from the CSAR file
VNF Package Parser - translates VNF product blueprint into a format consumable by OPEN-O components
VNF Package Dry Run - performs a "dry run" install to ensure that the package can be deployed during instantiation
The tools are provided in a form of a shared library (Python module) that can be used in other projects. A CLI is also provided out-of-the box for DevOps to use the library with their scripts and autoamtion framework.

Repository
Name: vnf-sdk-design-pkg
Clone command: git clone https://gerrit.open-o.org/r/vnf-sdk-design-pkg

Installation
Python module with CLI is installed by Python pip command. It is possible to install into a virtual environment (virtualenv).

The following commands are executed in the cloned repository directory
1. pip install -r requirements.txt
   Install all required dependencies
2. pip install .
   Install VNF SDK tools package

Usage
- Create CSAR by specifying a directory
  vnf-sdk csar-create -d DESTINATION source entry
- Extract CSAR content
  vnf-sdk csar-open -d DESTINATION source
- Validate CSAR content
  vnf-sdk csar-validate source

All commands have -h switch which displays help and description of all parameters.

Function Test
- Abstract Design
- Function Test Requirements:
- Function Test Interface
  - Execute Function Test
  - Get Function Test Result
- Initial Design Flow
- Test Plan
  - Testing Plan:
- References:
  - Test Framework Selection
  - Enhanced REST Library for Robot framework

Abstract Design
Function Test Requirements:

- Support to integrate with different kinds of framework.
- Generate test result with different format.
- Deploy test framework and execute VNF's test scripts automatically
- Collect and display test result.
- Create or extend Robot Framework's test library.

Function Test Interface

Execute Function Test

<table>
<thead>
<tr>
<th>Interface Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI</td>
<td>/openoapi/vnfsdk/v1/functest</td>
</tr>
<tr>
<td>Operation Type</td>
<td>POST</td>
</tr>
<tr>
<td>Content-Type</td>
<td>multipart/form-data</td>
</tr>
</tbody>
</table>

Request Parameters:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Qualifier</th>
<th>Cardinality</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>M</td>
<td>1</td>
<td>InputStream</td>
<td>The CSAR file stream</td>
</tr>
</tbody>
</table>

Response:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Qualifier</th>
<th>Cardinality</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>functestId</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>Id of the function test</td>
</tr>
</tbody>
</table>
Get Function Test Result

<table>
<thead>
<tr>
<th>Interface Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI</td>
<td>/openoapi/vnfsdk/v1/functest</td>
</tr>
<tr>
<td>Operation Type</td>
<td>GET</td>
</tr>
<tr>
<td>Content-Type</td>
<td>multipart/form-data</td>
</tr>
</tbody>
</table>

**Request Parameters:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Qualifier</th>
<th>Cardinality</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>functestId</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>Id of function test</td>
</tr>
</tbody>
</table>

**Response:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Qualifier</th>
<th>Cardinality</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>M</td>
<td>1</td>
<td>Stream</td>
<td>result folder is zipped and sent out as stream</td>
</tr>
</tbody>
</table>

**Initial Design Flow**

![Initial Design Flow Diagram]

**Test Plan**

**Testing Plan:**

1. Test Interface between Market Place and Function Test
2. Execute a simple test and check whether tests are executed in remote machine and results are transferred to local machine.
3. Results must be displayed in the portal
References:

Test Framework Selection

Automatic Test Framework Selection for Current Release.pptx

Enhanced REST Library for Robot framework

Enhanced REST Library for Robot.pptx

Suggestion

Amir Levy: Suggest to focus on func-test interfaces - and leave the actual func-test to VNF's that specialized

Artur Tyloch: For functest we should consider ABOT framework. It is open source VNF functional testing framework maintained by Rebaca https://juju charms.com/u/debayan-ch/abot-ims-basic/trusty/35

Market Place

Market Place API

Upload VNF Package

<table>
<thead>
<tr>
<th>Interface Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI</td>
<td>/openoapi/marketplace/v1/vnf</td>
</tr>
<tr>
<td>Operation Type</td>
<td>POST</td>
</tr>
<tr>
<td>Content-Type</td>
<td>multipart/form-data</td>
</tr>
</tbody>
</table>

Request Parameters:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Qualifier</th>
<th>Cardinality</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>M</td>
<td>1</td>
<td>InputStream</td>
<td>The CSAR file stream</td>
</tr>
<tr>
<td>file</td>
<td>M</td>
<td>1</td>
<td>FormDataContentDisposition</td>
<td>The detail of CSAR file</td>
</tr>
</tbody>
</table>

FormDataContentDisposition

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>the disposition type. will be &quot;form-data&quot;</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>the control name</td>
</tr>
<tr>
<td>fileName</td>
<td>String</td>
<td>the file name</td>
</tr>
<tr>
<td>creationDate</td>
<td>Date</td>
<td>the creation date</td>
</tr>
<tr>
<td>modificationDate</td>
<td>Date</td>
<td>the modification date</td>
</tr>
</tbody>
</table>
Response:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Qualifier</th>
<th>Cardinality</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>csarId</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>The CSAR identifier</td>
</tr>
</tbody>
</table>

Query VNF information by csarId

**Interface Definition**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI</td>
</tr>
<tr>
<td>/openoapi/marketplace/v1/csars/{csarId}</td>
</tr>
</tbody>
</table>

**Operation Type**

GET

**Request Parameters:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Qualifier</th>
<th>Cardinality</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>csarId</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>The id of CSAR package</td>
</tr>
</tbody>
</table>

Response:

**CSARPackage**

Sample:

```json
{
    "csarId": "78ede6f3-66cc-46ab-b748-38a6c010d272",
    "name": "NanocellGateway",
    "provider": "ZTE",
    "version": "V1.0",
    "createTime": "2016-06-29 03:33:15",
    "modifyTime": "2016-06-29 09:33:15",
    "size": "0.93M",
    "type": "CSAR"
}
```

Download VNF package files download URI

**Interface Definition**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI</td>
</tr>
<tr>
<td>/openoapi/marketplace/v1/csars/{csarId}/files?relativePath=xxx</td>
</tr>
</tbody>
</table>

**Operation Type**

GET

**Request Parameters:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Qualifier</th>
<th>Cardinality</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>csarId</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>The id of CSAR</td>
</tr>
<tr>
<td>relativePath</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>The file relative path in CSAR package</td>
</tr>
</tbody>
</table>

Sample:

/openoapi/marketplace/v1/csars/78ede6f3-66cc-46ab-b748-38a6c010d272/files?relativePath="/images/segw.img"

Response:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Qualifier</th>
<th>Cardinality</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>downloadUri</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>The download URI of file</td>
</tr>
</tbody>
</table>
Validate & Lifecycle Test

- Background
- Overall Design
  - Validate
  - Lifecycle Test
- Interface Design (draft - need to be confirmed)
  - VNF Package LifecycleTest and Validate
  - VNF Package Delete
  - Update Validation/Lifecycle Test interface
  - VNF Instantiation
  - VNF Termination
  - Validate
- Resource

Background

Validate & Lifecycle Test as one of the parts of VNF SDK project in OPEN-O, there are two main functions in this part, one is validating the package from vendors, and the other is doing the lifecycle test for the package (e.g. instantiation, termination, scaling, etc.). The scope of Validate & Lifecycle Test component is shown in the figure with purple blocks.

Overall Design

Validate

In Mercury Release, by reusing the onboarding operation for VNF package in NFVO, the package from vendors can be validated. There are three main functions listed below.
Format checking
Integrity checking
Tamper proofing

Lifecycle Test

In Mercury Release, by reusing the instantiation and termination operation for VNF in NFVO, the lifecycle test for the package can be done. There are two main functions listed below.

- The instantiation of VNF
- The termination of VNF

In other words, in mercury release, NFVO will provide the functional entity for the validation module, and the operational entity for the lifecycle test module.

Interface Design (draft - need to be confirmed)

Refer to NS LCM API Definition of NFVO.

1. here VNF-SDK use NFVO onboarding interface
2. NFVO onboarding interface will cover package Validation and Lifecycle test functionality. VNF-SDK need package the package(or Package Id) to NFVO do onboarding operation

VNF Package LifecycleTest and Validate

<table>
<thead>
<tr>
<th>Interface Definition</th>
<th>Description</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI</td>
<td>/openoapi/vnfsdk/v1/vnfpackage</td>
<td>MarketPlace-&gt;Lifecycle-&gt;NFVO</td>
</tr>
<tr>
<td>Method</td>
<td>POST</td>
<td></td>
</tr>
</tbody>
</table>

Request:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Qualifier</th>
<th>Cardinality</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package Id</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>VNF package id in Marketplace</td>
</tr>
</tbody>
</table>

Response:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Qualifier</th>
<th>Cardinality</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jobId</td>
<td>M</td>
<td>1</td>
<td>Identifier</td>
<td>The identifier of the VNF package operation occurrence job.</td>
</tr>
</tbody>
</table>

VNF Package Delete

<table>
<thead>
<tr>
<th>Interface Definition</th>
<th>Description</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI</td>
<td>/openoapi/vnfsdk/v1/vnfpackage/{csarId}</td>
<td>MarketPlace-&gt;Lifecycle-&gt;NFVO</td>
</tr>
<tr>
<td>Method</td>
<td>DELETE</td>
<td></td>
</tr>
</tbody>
</table>

Request:

none

Response:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Qualifier</th>
<th>Cardinality</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jobId</td>
<td>M</td>
<td>1</td>
<td>Identifier</td>
<td>The identifier of the VNF package operation occurrence job.</td>
</tr>
</tbody>
</table>

Update Validation/Lifecycle Test interface
### Request:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Qualifier</th>
<th>Cardinality</th>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jobId</td>
<td>M</td>
<td>1</td>
<td>Identifier</td>
<td>The identifier of the VNF package operation occurrence job.</td>
</tr>
<tr>
<td>Validate_status</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>Indicate Validation success or not</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Should be: &quot;Pass&quot; or &quot;Fail&quot;</td>
</tr>
<tr>
<td>Lifecycle_status</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>Indicate Lifecycle Test success or not</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Should be: &quot;Pass&quot; or &quot;Fail&quot;</td>
</tr>
<tr>
<td>Vnf_info</td>
<td>O</td>
<td>1</td>
<td>String</td>
<td>The information about the Package deployed, only valid when Lifecycle_status is &quot;Pass&quot;</td>
</tr>
<tr>
<td>Vms</td>
<td>M</td>
<td>1-n</td>
<td>String</td>
<td>The Virtual Machine information inside the vnf</td>
</tr>
<tr>
<td>--ip</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>The management ip of VM</td>
</tr>
<tr>
<td>--username</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>The username that can access the VM</td>
</tr>
<tr>
<td>--Password</td>
<td>M</td>
<td>1</td>
<td>String</td>
<td>The Password that can access the VM</td>
</tr>
</tbody>
</table>

```json
{
    "jobId": "1",
    "validate_status": "Pass",
    "lifecycle_status": "Pass",
    "vnf_info": {
        "vms": [{
            "ip": "1.1.1.1",
            "username": "open-o",
            "password": "*******"
        },
        {
            "ip": "1.1.1.2",
            "username": "open-o-2",
            "password": "*******"
        }]
    }
}
```

### Response:

None

### VNF Instantiation

As previous discussion, the VNF Package Onboarding interface includes the process of onboarding and the instantiation of VNF.

### VNF Termination

Whether the VNF Package Delete interface including the termination of VNF or not needs to be confirmed.

### Validate

NFVO onboarding interface will cover validate functionality
VNF SDK Testing

- VNF SDK Function Test CSIT use cases
- VNF SDK Marketplace CSIT use cases
- VNF SDK package tool CSIT use cases
- VNF SDK Validation / Lifecycle Test CSIT use cases

VNF SDK Function Test CSIT use cases

Test Plan

Testing Plan:

1. Use Robot Framework scripts to test each REST Interface.
2. All Test cases must be running in Jenkins.

Sample Test Script Link:

Sample CSIT cases

Use Case 1 – Done (Running in Jenkins)

Basic Health Check for the service including MSB registration

Use Case 2 - Done (Running in Jenkins)

Upload the CSAR package to VNF Function test with Firewall VNF test cases. Result should return the funcTestId.

Use Case 3 - Done (Running in Jenkins)

Get the function test result using funcTestId. Result should return result zip file in Stream format.

VNF Vendor Script

Firewall function test use case.

a. Ping remote machine
b. Connect to remote machine
c. execute firewall command to stop accepting icmp packets from test machine
d. Ping remote machine and it should fail
e. execute firewall command to remove the above firewall command to block icmp ping.

f. Ping remote machine and it should pass now.

VNF SDK Marketplace CSIT use cases

Use Case 1 - Done (Running in Jenkins)

query and publish CSAR list in UI

Use Case 2 - Done (Running in Jenkins)

upload CSAR to marketplace repository which includes validation, life cycle test and functional test

Use Case 3 - Done

download CSAR from marketplace repository

VNF SDK package tool CSIT use cases

Use Case 1

Package a properly structured files into CSAR

Preconditions

A directory that contains a proper YAML file. Additional files may be provided

Tests steps

1. call csar-create procedure to create a temporary CSAR file from input directory
2. check for errors
3. check if the file is not empty

Expected result

A non-empty CSAR file is created

Use Case 2

Validate a properly structured CSAR

Preconditions

A reference CSAR file provided for the test case

Tests steps

1. call csar-validate procedure using the provided CSAR file
2. check for errors

Expected result

Operation ends with no errors
Use Case 3

Extract proper CSAR and confirm consistency with original files

Preconditions

A reference CSAR file and reference files to compare with the extracted ones

Tests steps

1. extract the reference CSAR file to a temporary directory
2. compare the extracted files with the reference ones (they should be identical)

Expected result

Extracted files are identical as the reference files

Use Case 4

Package invalid structure into CSAR - Attempt should end with error information

Preconditions

A malformed CSAR file is provided

Tests steps

1. attempt to extract the malformed CSAR file

Expected result

Operation ends with error

VNF SDK Validation / Lifecycle Test CSIT use cases

Use Case 1 - Done (Running in Jenkins)

Call VNF Package Onboarding interface of NFVO to validate the VNF package.

Use Case 2 - Done (Running in Jenkins)

Call Instantiate VNF interface of NFVO to instantiate VNF.

Use Case 3 - Done

Call Terminate VNF interface of NFVO to terminate VNF.