Illustrative Sequence Diagrams
for Residential Broadband vCPE Use Case
vCPE Use Case
Residential Broadband vCPE Use Case Model: Infrastructure Flows

<table>
<thead>
<tr>
<th>Service</th>
<th>Service Level Flow</th>
<th>Resource</th>
<th>Resource Level Flow</th>
<th>SDNC Northbound API</th>
<th>SDNC DG</th>
</tr>
</thead>
<tbody>
<tr>
<td>vCpeCoreInfra_X</td>
<td>Generic Service</td>
<td>vCpeCoreInfraVnfs_X (VNF)</td>
<td>Generic VNF</td>
<td>GENERIC-RESOURCE</td>
<td>Generic DG</td>
</tr>
<tr>
<td>&quot;</td>
<td>CPE_PUBLIC (Network)</td>
<td>Generic Ntw</td>
<td>GENERIC-RESOURCE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;</td>
<td>CPE_SIGNAL (Network)</td>
<td>Generic Ntw</td>
<td>GENERIC-RESOURCE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vGMuxInfra</td>
<td>Generic Service</td>
<td>vGMUX (VNF)</td>
<td>Generic VNF</td>
<td>GENERIC-RESOURCE</td>
<td>Generic DG</td>
</tr>
<tr>
<td>&quot;</td>
<td>&quot;</td>
<td>MUX_GW (Network)</td>
<td>Generic Ntw</td>
<td>GENERIC-RESOURCE</td>
<td>Generic DG</td>
</tr>
<tr>
<td>vBngInfra</td>
<td>Generic Service</td>
<td>vBNG (VNF)</td>
<td>Generic VNF</td>
<td>GENERIC-RESOURCE</td>
<td>Generic DG</td>
</tr>
<tr>
<td>&quot;</td>
<td>&quot;</td>
<td>BRG_BNG (Network)</td>
<td>Generic Ntw</td>
<td>GENERIC-RESOURCE</td>
<td>Generic DG</td>
</tr>
<tr>
<td>BNG_MUX</td>
<td>Generic Service</td>
<td>BNG_MUX (Network)</td>
<td>Generic Ntw</td>
<td>GENERIC-RESOURCE</td>
<td>Generic DG</td>
</tr>
<tr>
<td>BRG_EMU</td>
<td>Generic Service</td>
<td>BRG_EMU (VNF)</td>
<td>Generic VNF</td>
<td>GENERIC-RESOURCE</td>
<td>Custom Process (Event Handling)</td>
</tr>
<tr>
<td>&quot;</td>
<td>&quot;</td>
<td>vG (VNF)</td>
<td>Generic VNF</td>
<td>GENERIC-RESOURCE</td>
<td>Generic DG</td>
</tr>
<tr>
<td>&quot;</td>
<td>&quot;</td>
<td>BRG (PNF)</td>
<td>Custom [New]</td>
<td>GENERIC-RESOURCE</td>
<td>Custom DG [New]</td>
</tr>
</tbody>
</table>

Note that in this table "_X" indicates applicability to both the "_HEAT" and "_TOSCA" approaches.

For simple Services which include only simple networks and VNFs (e.g., with no multi-data instances that map to different VF Modules), there is an SO “Generic Service” flow (“top level flow”) that calls the Generic VNF and/or the Generic Network resource-level flows. The SDNC functionality is also “generic” such that only modeling and configuration is needed to drive SDNC behavior for a specific VNF type. For example, this SDNC generic VNF flow can automatically assign the IP Addresses if pre-loaded. For “Generic VNF” the IP Addresses are pre-loaded. E.g., pre-load 25 vG instances with their assignments pre-populated. SDNC keeps track of which instances have/have not been assigned. It is expected that these SO and SDNC assets will be leveraged.
Residential Broadband vCPE Use Case Model: vCpeResCust & vGMuxInfra Topology

vCpeResCust Service:
topology_template:
node_templates:
  TunnelXConn (AllotRes):
  BRG (PNF):
  vG (VNF):

vGMuxInfra Service:
topology_template:
node_templates:
  MUX_GW (Ntw):
  vGMux (VNF)
Capabilities:
  TunnelXConnCapability

TunnelXConn AllottedResource:
Requirement: TunnelXConnCapability

BRG PNF Resource:
vG VNF Resource:
vG (VFC)
vG VF Module:

MUX_GW Network Resource:
vGMUX VNF Resource:
vGMUX (VFC)
vGMUX VF Module:
Residential Broadband vCPE Use Case Model: vBngInfra and BNG_MUX Topology

vBngInfra Service:
topology_template:
  node_templates:
    vBNG (VNF):
    BRG_BNG (Network):

BNG_MUX Service:
topology_template:
  node_templates:
    BNG_MUX (Network):

vBNG VNF Resource:
vBNG (VFC)

vBNG VF Module:

BRG_BNG Network Resource:

BNG_MUX Network Resource:
The BRG_EMU Service is an artificial construct to leverage ONAP to instantiate the BRG Emulator, which is implemented as a VNF in this Use Case. This Service would have no analogue in the “real world”.

The vBRG_EMU VNF has no modeling relationship with the BRG PNF which is referenced in the vCpeResCust Service. The BRG PNF is designed to be useful in the “real world” in which a BRG is a true PNF, whereas the vBRG_EMU VNF is a construct that is useful only for standing up the BRG Emulator used in this Use Case.
Residential Broadband vCPE Use Case: vCpeCoreInfra Topology

vCpeCoreInfra_Heat Service:
topology_template:
  node_templates:
    CPE_PUBLIC:
    CPE_SIGNAL:
  vCpeCoreInfraVnfs_Heat:

vCpeCoreInfra_Tosca Service:
topology_template:
  node_templates:
    CPE_PUBLIC:
    CPE_SIGNAL:
    vCpeCoreInfraVnfs_TOSCA:
As part of the “Assign” operation, SDNC assigns the “10.1” subnet to this network instance.
In Release 1 this sequencing of the vGMuxInfra Resources will be accomplished via the Generic SO Service-Level which always instantiates Networks before VNFs.

As part of the “Assign” operation, SDNC assigns the “10.3” subnet to this network instance. As part of the “Assign” operation, SDNC determines that the VNF must connect to two networks: BRG_BNG “10.3” and BNG_MUX “10.1”. It also assigns the IPs “10.3.0.1” and “10.1.0.10” at these networks. As part of the “Activate” operation, SDNC configures a static route between the 10.3 and the 10.1 subnets.

Robot will remember…
Residential Broadband vCPE Use Case Model: Instantiation Sequencing

Service Level (TOSCA)
- BRG_EMU Service: topology_template: node_templates: BRG_EMU (VNF):
- VID: Manual sequencing via VID

VNF Level (TOSCA)
- BRG_EMU VNF:
- Generic Service Flow [Existing]
- VID: 13

PNF Level (TOSCA)
- BRG PNF:
- Generic VNF Flow [Existing]

Allotted Resource (TOSCA)
- TunnelXConn AllottedResource:
- Custom [New]

Robot is acting as a BSS emulator, and is kicked off automatically upon detecting that the BRG MAC Address has registered.

This sequencing of the vCpeResCust Resources will be accomplished via a custom vCpeResCust SO Service-Level workflow in Release 1. In Release 2+ we hope a generic Service-Level workflow can perform sequencing from dependencies modeled in SDC.

vCpeResCust Service: topology_template: node_templates:
- TunnelXConn (AllottedResource):
- vG (VNF):
- BRG (PNF):
- Custom [New]

Depends On
- TunnelXConn (AllottedResource): 16
- vG (VNF): 17
- BRG (PNF): 18

Custom [New]
Residential Broadband vCPE Use Case Model: vCpeResCust & vGMuxInfra Inventory

Instance Example

- **vCpeResCust Service Instance**
  - Instance “A”

- **vGMuxInfra Service Instance**
  - Instance “B”

- **vG VNF Instance**
  - Instance “X”

- **BRG PNF Instance**
  - Instance “Y”

- **vGMUX VNF Instance**
  - Instance “Z”

- **TunnelXConn AllottedResource Instance**
  - Instance “N”

- **vG VF Module Instance**
  - Instance “J”

- **vGMUX VF Module Instance**
  - Instance “K”

- **MUX_GW Network Instance**
  - Instance “Z”
Residential Broadband vCPE Use Case Model: BRG_EMU and vCpeCoreInfra Inventory Instance Example
Residential Broadband vCPE Use Case Model: BRG_EMU and vCpeCoreInfra Inventory Instance Example

vCpeCoreInfra_HEAT Service Instance
  Instance “C”

vCpeCoreInfraVnfs_HEAT VNF Instance
  Instance “X”

vCpeCoreInfraVnfs_HEAT VF Module Instance
  Instance “L”

CPE_PUBLIC Network Instance
  Instance “Y”

CPE_SIGNAL Network Instance
  Instance “Z”

vCpeCoreInfraTOSCA Service Instance
  Instance “D”

vCpeCoreInfraVnfs_TOSCA VNF Instance
  Instance “X”

CPE_PUBLIC Network Instance
  Instance “Y”

CPE_SIGNAL Network Instance
  Instance “Z”

No VF Modules on the TOSCA VNF.
Residential Broadband vCPE Use Case Model: vCpeResCust & vGMuxInfra Topology

- vBngInfra Service Instance
  - Instance “B”
- vBNG VNF Instance
  - Instance “V”
- vBNG VF Module Instance
  - Instance “K”
- BRG_BNG Layer 3 Network Instance
  - Instance “K”
- BNG_MUX Service Instance
  - Instance “C”
- BNG_MUX Layer 3 Network Instance
  - Instance “K”
The Cloud Region object in A&AI is created via the ESR (A&AI) portal:

- User inputs backend Cloud information into ESR portal.
- ESR stores the backend Cloud information as auth modal into A&AI, key is cloudowner_cloudregion.
- User triggers VIM register service exposed by Multi VIM which will trigger registry implements in different VIM plugins to fill in VIM Model information into A&AI.
- Each plugin handles A&AI query about the backend Cloud information for backend Cloud operations.

### Key:
- **SO**: Created by...
- **ESR**
- **Multi-VIM**

---

Multi-VIM plans to create the Tenant, Flavor, and Image objects in A&AI when a new VIM is instantiated.

---

"Cloud Region" is a representation of a unique cloud instance which has geographical significance. Multi-VIM expects SO to pass the Cloud Region ID in with the request along with the Keystone Identity URL.
As part of the Network-level “Assign” operation, SDNC assigns the network name and the “10.2” subnet to this network instance.

As part of the VNF-level “Assign” operation, SDNC determines that this VNF must connect to three network types. There is no infrastructure-level configuration of the VNFs needed.

As part of the VF Module-level “Assign” operation, SDNC assigns the IPs on each of the networks.

The Generic Service flow doesn’t call “homing”. Rather it works off of separate (non-TOSCA) fielded attributes in the Svc Instantiation API for Cloud Configuration Data, which includes Cloud_Region and Cloud_Tenant.

The “10.4” subnet is assigned at the Network level.
Residential Broadband vCPE Use Case Model: BRG_EMU Service Data Mappings

5. **Service Level**
   - BNG_MUX Service:
     - topology_template:
       - node_templates:
         - BNG_MUX (Ntw):
           - Input Attributes:
             - {NULL Set}

6. **Resource Level**
   - BNG_MUX Network:
     - Input Attributes:
       - {NULL Set}
     - Assignable Attributes:
       - Subnet

The Generic Service flow doesn’t call “homing”. Rather it works off of separate (non-TOSCA) fielded attributes in the Svc Instantiation API for Cloud Configuration Data, which includes Cloud_Region and Cloud_Tenant.

As part of the Network-level “Assign” operation, SDNC assigns the network name and the “10.1” subnet to this network instance.
Residential Broadband vCPE Use Case Model: vGMuxInfra Service Data Mappings

Service Level

vGMuxInfra Service:
topology_template:
node_templates:
  vGMux (vGMUX):
  MUX_GW (Network):
Input Attributes:
{NULL Set}

As part of the VNF-level “Assign” operation, SDNC determines that the vGMUX must connect to three network types. SDNC assigns the network names for each. SDNC uses the TOSCA to determine that these connection points need to be assigned; SDNC returns a structure to SO, which SO maps to the HEAT. There is no infrastructure-level configuration of the vGMUX needed.

Resource Level

vGMUX VNF:
Input Attributes:
{NULL Set}
Assignable Attributes:
Ext_Conn_Pt (BNG_MUX)
Ext_Conn_Pt (MUX_GW)
Ext_Conn_Pt (ONAP_OAM)
VNI Pool
Configuration Attributes:
{NULL Set}

As part of the Network-level “Assign” operation, SDNC assigns the “10.5” subnet to this network instance.

VF Module Level

vGMUX VF Module:
Input Attributes:
{NULL Set}
Assignable Attributes:
vGMUX_WAN_IP (BNG_MUX)
vGMUX_LAN_IP (MUX_GW)
OA&M_IP (ONAP_OAM)

As part of the VF Module-level “Assign” operation, SDNC assigns the IPs “10.1.0.20”, “10.5.0.20” and “10.0.101.20” on these three networks.

The Generic Service flow doesn’t call “homing”. Rather it works off of separate (non-TOSCA) fielded attributes in the Svc Instantiation API for Cloud Configuration Data, which includes Cloud_Region and Cloud_Tenant.
Residential Broadband vCPE Use Case Model: vBNG Service Data Mappings

Service Level

10 vBngInfra Service:
topology_template: node_templates:
vBNG: BRG_BNG
Input parameters: {NULL Set}

Resource Level

11 BRG_BNG Network:
Input Attributes: {NULL Set}
Assignable Attributes: Subnet

As part of the Network-level “Assign” operation, SDNC assigns the “10.3” subnet to this network instance.

12 vBNG VNF:
Input Attributes: {NULL Set}
Assignable Attributes:
- Ext_Conn_Pt (CPE_SIGNAL)
- Ext_Conn_Pt (BNG_MUX)
- Ext_Conn_Pt (BRG_BNG)
- Ext_Conn_Pt (ONAP_OAM)
Configuration Attributes: {NULL SET}

As part of the VNF-level “Assign” operation, SDNC determines that the vBNG must connect to four network types. SDNC assigns the network names for each. SDNC uses the TOSCA to determine that these connection points need to be assigned; SDNC returns a structure to SO, which SO maps to the HEAT. There is no infrastructure-level configuration of the vBNG needed.

As part of the VF Module-level “Assign” operation, SDNC assigns the IPs “10.4.0.3”, “10.1.0.10”, “10.3.0.1”, and “10.0.101.10” to these networks, respectively.

The Generic Service flow doesn’t call “homing”. Rather it works off of separate (non-TOSCA) fielded attributes in the Svc Instantiation API for Cloud Configuration Data, which includes Cloud.Region and Cloud.Tenant.
The BRG_WAN_MAC_Address is communicated via the BRG HEAT.

The BRG_WAN_MAC_Address is communicated via the BRG HEAT.

There is a feature in 1710 whereby the generic Service Level flow converts Service-Level Input Attributes into a MAP that is then sent as an input into every subtending building block (VNF level and VF Module level). We will rely on that feature. If for some reason that feature doesn’t make it into ONAP R1, we will depend on the “a la carte” method to instantiate the BRG_EMU Service, VNF, and VF Module. In an “a la carte” method, the MAC Address would only be on the VF Module level input.

The Generic Service flow doesn’t call “homing”. Rather it works off of separate (non-TOSCA) fielded attributes in the Svc Instantiation API for Cloud Configuration Data, which includes Cloud_Region and Cloud_Tenant.
Residential Broadband vCPE Use Case Model: vCpeResCust Input Data Mappings (R1)

**Service Level**
- vCpeResCust Service:
  - topology_template:
    - node_templates:
      - vG (VNF): TunnelXConn (AllottedResource):
        - BRG (PNF):
          - Input Attributes:
            - BRG_WAN_MAC_Address
            - vGMxuInfra_UUID
            - vGMux_Cloud_Region
          - Homing Attributes:
            - vGMuxInfra_UUID
          - Service Level VNF Level
          - Allotted Resource
          - Configuration Attributes:
            - vBRG_WAN_IP
            - VNI
          - Mapped From
          - PNF Level
          - BRG PNF:
            - Key: BRG_WAN_MAC_Address
            - Input Attributes:
              - vGMUX_Bearer_IP
              - VNI
            - Assignable Attributes:
              - BRG_WAN_MAC_Address
            - Configuration Attributes:
              - vGMUX_LAN_IP
              - VNI
              - vGMUX_Bearer_IP
          - Mapped From
          - VNF Level
          - vG VNF:
            - Input Attributes:
              - VNI
              - vGMUX_LAN_IP
            - Assignable Attributes:
              - vG_LAN_IP
              - VNI
              - vGMUX_Bearer_IP
          - Mapped From
          - VF Module Level
          - vG VF Module:
            - Input Attributes:
              - {NULL Set}
            - Assignable Attributes:
              - vG_LAN_IP
              - VNI
              - vGMUX_LAN_IP
          - Mapped From

The Input Data for the vCpeResCust Resources will be mapped by the custom vCpeResCust SO Service-Level workflow in Release 1. In Release 2+ we hope a generic Service-Level workflow can derive this data mapping from information contained in the SDC model.

As part of the VNF-level “Assign” operation, SDNC determines that the vG must connect to three network types. SDNC assigns the network names for each. SDNC also looks up the vGMUX_LAN_IP (10.5.0.20). SDNC will configure the vG_LAN_IP, VNI, and vGMUX_LAN_IP.
Residential Broadband vCPE Use Case Model: vCpeResCust Input Data Mappings (R1)

Service Level

- **vCpeResCust Service:**
  - **topology_template:**
    - node_templates:
      - vG (VNF):
      - TunnelXConn (AllottedResource):
        - BRG (PNF):
  - **Input Attributes:**
    - BRG_WAN_MAC_Address
    - Customer_Location
  - **Homing Attributes:**
    - Customer_Location

PNF Level

- **BRG PNF:**
  - **Key:**
    - Mapped From
    - BRG_WAN_MAC_Address
  - **Input Attributes:**
    - vGMUX_LAN_IP
    - vGMUX_Bearer_IP
  - **Assignable Attributes:**
    - NULL Set
  - **Configuration Attributes:**
    - vBRG_WAN_IP
    - VNI
    - VNI
    - vGMUX_Bearer_IP

VF Module Level

- **vG VF Module:**
  - **Input Attributes:**
    - NULL Set
  - **Assignable Attributes:**
    - vG_LAN_IP
    - OA&M_IP

VF Module Level

- **vG VF Module:**
  - **Input Attributes:**
    - NULL Set
  - **Assignable Attributes:**
    - vG_LAN_IP
    - OA&M_IP

VNF Level

- **vG VNF:**
  - **Input Attributes:**
    - VNI
    - vGMUX_LAN_IP
  - **Assignable Attributes:**
    - Ext_Conn_Pt (CPE_PUBLIC)
    - Ext_Conn_Pt (MUX_GW)
    - Ext_Conn_Pt (ONAP_OAM)
  - **Configuration Attributes:**
    - vG_LAN_IP
    - VNI
    - vGMUX_LAN_IP

Allotted Resource

- **TunnelXConn AllottedResource:**
  - **Input Attributes:**
    - BRG_WAN_MAC_Address
  - **Assignable Attributes:**
    - VNI
    - vGMUX_LAN_IP
    - vGMUX_Bearer_IP
  - **Configuration Attributes:**
    - Xconn: {BRG_WAN_IP, VNI}

The Input Data for the vCpeResCust Resources will be mapped by the custom vCpeResCust SO Service-Level workflow in Release 1. In Release 2+ we hope a generic Service-Level workflow can derive this data mapping from information contained in the SDC model.

As part of the VNF-level “Assign” operation, SDNC determines that the vG must connect to three network types. SDNC assigns the network names for each. SDNC also looks up the vGMUX_LAN_IP (10.5.0.20). SDNC will configure the vG_LAN_IP, VNI, and vGMUX_LAN_IP.
Homing_Policy: Latency \{Customer\_Location, TunnelXConn\} < X ms

Homing_Policy: Affinity \{TunnelXConn, vG\} <Same Cloud Zone>

SDC should, in the future, be able to derive the “Homing Input” attributes for a given Service type by determining which homing policy variables have no defined source.
Generic Service Level Processing

SO (Service Level):
GenSvc: <SvcName>

Create Generic Service Instance
(Type=<SvcName>, Cloud_Config_Data)

Get SvcTemplate
(Type=<SvcName>)

Assign Service Instance
Inventory Object UUID()

Ack (UUID=<SvcName> UUID)

Create Service Instance
Inventory Object (Type=<SvcName>, UUID)

Decompose

Internal Homing

SO Generic Service Level flow calls the "Decomposition Building Block". *Decompose* consists of parsing the TOSCA Service Template content to determine the complete set of Resource Node Types. Also included is determining the sequence between the template Node Types based on the template content, which we will assume to be as follows:
* <VnFName> VNF
* <NtNName> Network

The Generic Service flow doesn't call "homing". Rather it works off of separate (non-TOSCA) fielded attributes in the Svc Instantiation API for Cloud Configuration Data, which includes Cloud Region and Cloud Tenant.

Note that this diagram must be updated to incorporate the correct ONAP internal API names. For now, only intuitive names have been used for the sake of readability.
First bring the `<NtwName>` network instance online.

Create Generic Ntw Instance (Type=<NtwName>)

Assign Ntw UUID

Create Ntw Inventory Instance (Type=<NtwName> UUID)

Assign Generic Network (Type=<NtwName> UUID)

Get Generic Network Assignments (UUID)

Generate HEAT (Type=<NtwName>)

Instantiate Workload (Template=HEAT, cloud=<Cloud_Region>)

Activate Ntw (Type=<NtwName> UUID)

Update Ntw Instance Inventory Object (UUID)

Success

The Cloud Region was passed in the original Service Instantiation request input.

The Service Orchestrator will populate the HEAT.

In the current implementation, the Generic Service-Level flow is coded to instantiate all networks prior to instantiating any VNIs. If there are multiple Networks in the decomposed Service Structure, then the Networks are instantiated in an indeterminate order. In the (hopefully near) future this will be changed such that the Service-Level flow sequencing is driven from the Service-Level TOSCA.

EDNC assigns the "name" for the network instance as well as the subnet. Assignments are driven from the TOSCA Model of the Network.
Generic Resource Level Processing: VNFs
vCpeCoreInfra_HEAT
And
vCpeCoreInfra_TOSCA
Service Level Processing

Note that this diagram must be updated to incorporate the correct ONAP internal API names. For now only intuitive names have been used for the sake of readability.

SO Generic Service Level flow calls the "Decomposition Building Block". *Decompose* consists of parsing the TOSCA Service Template content to determine the complete set of Resource Node Types. Also included is determining the sequence between the template Node Types based on the template content, which we will assume to be as follows:
- CPE_PUBLIC
- CPE_SIGNAL
- vCpeCoreInfraVnfs VNF

The Generic Service flow doesn't call "homing". Rather it works off of separate (non-TOSCA) fielded attributes in the Svc Instantiation API for Cloud Configuration Data, which includes Cloud_Region and Cloud_Tenant.
As per the Node Type dependencies captured in the vCpeCoreInfra TOSCA, next bring the CPE_PUBLIC_HEAT network instance online.

1. Create Generic Ntw Instance (Type=CPE_PUBLIC_HEAT)
   - Assign Ntw UUID
2. Create Ntw Inventory Instance (Type=CPE_PUBLIC_HEAT, UUID)
3. Assign Generic Network (Type=CPE_PUBLIC_HEAT, UUID)
4. Get Generic Network Assignments (UUID)
5. Generate HEAT (Type=CPE_PUBLIC_HEAT)
6. Instantiate Workload (Template=HEAT, Cloud=<Cloud_R...>)
7. Activate Ntw (Type=CPE_PUBLIC_HEAT, UUID)
8. Update Ntw Instance Inventory Object (UUID)

SDNC assigns the "name" for the network instance as well as the subnet (*10.2.0.0/24*).

The Service Orchestrator will populate the HEAT.
vCpeCoreInfra_HEAT
Resource Level
Processing
(CPE_SIGNAL_HEAT)
As per the Node Type dependencies captured in the vCpeCoreInfra.HEAT TOSCA, next bring the vCpeCoreInfraVnfs.HEAT VNF instance online.

- SDNC selects the network type(s) to connect to this VNF as per the VNF Model, in this case the "CPE_PUBLIC", "CPE_PUBLIC", and "GNAR_OAM" networks.
- For each VF Module in the VNF Model definition:
  - Assign Vf Module UUID
  - Create Vf Module Inventory Instance (Type=vCpeCoreInfraVnfs.HEAT, UUID)
  - Assign Generic Vf Module Network Resources (UUID)
  - SDNC assigns the CPE_PUBLIC_IP, CPE_SIGNAL_IP, and QAM_IP
  - Generate HEAT
  - Instantiate Workload (Template=HEAT, Cloud=Cloud.Region)
  - Update Vf Module Instance Inventory Object (UUID)
  - Activate Vf [UUID]
  - Update Vf Instance Inventory Object (UUID)

Note: not all operations are shown, such as setting status in AGAI and SDNC.

Success()
To incorporate ARIA option, need to show that the SO Adaptor determines from the SDC Model that the VNF is described using TOSCA versus HEAT. In the former case the Adaptor will generate a TOSCA document to forward to ARIA, rather than what is shown here.

Need to flesh out the interactions between HEAT and OS and ARIA and OS.
BNG_MUX Service and Resource Level Processing

As per the Node Type dependencies captured in the BNG_MUX Service TOSCA, next bring the BNG_MUX Network Instance online.

- Create Generic Service Instance (Type=BNG_MUX)
- Create New Inventory Instance (Type=BNG_MUX, UUID)
- Assign Generic Network (Type=BNG_MUX, UUID)
- Generate Service Level (Type=BNG_MUX, UUID)
- InstANTIATE Workload (Template=HEAT, Cloud=Cloud_Tenant)
- Activate Ntw (Type=BNG_MUX, UUID)
- Update NtwInstance Inventory Object (UUUID)

SDN assigns the "name" for the network instance as well as the subnet ("10.1.6.0/24").
vGMuxInfra Service Level Processing

SO [Existing] (Service Level): GenSvc: vGMuxInfra
SO [Existing] (Resource Level): vGenVnf; vGMUX

Create Generic Service Instance (Type=vGMuxInfra, Cloud_Config_Data)
Get SvcTemplate (Type=vGMuxInfra)
Assign Service Instance Inventory Object UUID()

Ack (UUID=vGMuxInfra UUID)

Note that this diagram must be updated to incorporate the correct ONAP internal API names. For now, only intuitive names have been used for the sake of readability.

For purposes of the Use Case, Robot retains the vGMuxInfra Service Instance UUID for purposes of sending with the vCpeResCust request. In the real world, a high-volume consumer service like vCpeResCust would require automated "homing", which will not be available until the introduction of SNRIO in ONAP R2.

Create Service Instance Inventory Object (type=vGMuxInfra.UUID)

SO Generic Service Level flow calls the "Decomposition Building Block". "Decompose" consists of parsing the TOSCA Service Template content to determine the complete set of Resource Node Types. Also included is determining the sequence between the template Node Types based on the template content, which we will assume to be as follows:

* vGMUX VNF
* MJX_GW Network

The Generic Service flow doesn’t call "homing". Rather, it works off of separate (non-TOSCA) fielded attributes in the SvcInstantiation API for Cloud Configuration Data, which includes Cloud Region and Cloud_Tenant.
vGMuxInfra Resource Level Processing (MUX_GW Network)

As per the Node Type dependencies captured in the vGMuxInfra TOSCA, next bring the MUX_GW network instance online.

- Create Generic Ntw Instance (Type=MUX_GW)
- Assign Ntw UUID()
- Create Ntw Inventory Instance (Type=MUX_GW, UUID)
- Assign Generic Network (Type=MUX_GW, UUID)
- Get Generic Network Assignments (UUID)
- Generate HEAT (Type=MUX_GW)
- Instantiate Workload (Template=HEAT, Cloud=<Cloud Region>)
- Activate Ntw (Type=MUX_GW, UUID)
- Update Ntw Instance Inventory Object (UUID)
- SDNC assigns the "name" for the network instance as well as the subnet ("10.5.0.0/24")

Success

The Service Orchestrator will populate the HEAT.

The Cloud Region was passed in the original Service Instantiation request input.
vGMuxInfra
Resource Level Processing (VNF)

As per the Node Type dependencies captured in the vGMuxInfra TOSCA, next bring the vGMUX VNF Instance online:

1. Create Generic Vnf Instance (Type=vGMUX)
2. Assign Vnf UUID
3. Create Vnf Inventory Instance (Type=vGMUX_UUID)
4. SDNC selects the network type(s) to connect to this VNF as per the VNF Model, in this case the "BNG_MUX", "MUX_CLK", and "ONAP_GAM" networks. It also assigns a "VNI Pool".
5. Assign Generic VNF Network Resources (UUID)

For each Vf Module in the Vnf Model definition:

1. Assign Vf Module UUID
2. Create Vf Module Inventory Instance (Type=vGMUX_UUID)
3. SDNC assigns the vGMUX WAN_IP, vGMUX LAN_IP, and OAM_IP
4. The Service Orchestrator will populate the HEAT
5. Get Generic Vf Module Network Resources (UUID)
6. Generate HEAT
7. The Cloud Region was passed in the original Service instantiation request input.
8. Instantiate Workload (Template=HEAT, Cloud=<Cloud_Region>)
9. Instantiate vGMUX infrastructure
10. Update Vf Module Instance Inventory Object (UUID)

Activate Vnf [UUID]
Success

Success

Note: not all operations are shown, such as setting status in A&I and SDN-C.
Note that because the BRG_EMU Service is an artificial construct, this orchestration flow would have no analogue in the "real world".
As per the Node Type dependencies captured in the BRG_EMU TOCSA, next bring the vBRG_EMU VNF instance online

SO maps the "VBRG_MAC_Address" Service Level Input Attribute to the vBRG_EMU "VBRG_MAC_Address" Resource Level Input Attribute.

SDNC selects the network type(s) to connect to this VNF as per the VNF Model, in this case the "VBRG (RNO)" network.

For each VF Module in the VNF Model definition

The Service Orchestrator will populate the HEAT, including the "VBRG_MAC_Address" which was originally sent in the Service Level Input Attributes.

The Cloud Region was passed in the original Service Instantiation request input.

onap_uc_res_vcpe_brg_emu_r1_p2.html
Assumptions

Once the BRG_EMU VNF Controller configures the vBRG_EMU VNF (see prior slide), that VNF will initiate DHCP interactions. This will result in an event being generated, which will be intercepted by the BRG PNF Controller. Note that the BRG_EMU VNF Controller shown in the prior sequence diagram is a *different* Controller function, and hence may be a different Controller instance, than the BRG PNF Controller which receives this event.

Being independent of each other, the vCpeResCust service instantiation request may be received in ONAP before the BRG PNF Controller receives this event notification, or it may be received after. The following sequence diagrams show both possibilities. However, in Release 1, only the latter will be supported.
In Release 1 Use Case we will support only this alternative.

We will defer this alternative until Release 2 or beyond.
Because the vCpeResCust service involves an Allotted Resource and a PNF, the generic Service Level ("top level") SO flow will not support this Service. So in Release 1 we either need to extend the generic Service Level flow to support Allotted Resources/PNFs or build a Custom flow for vCpeResCust.
Aspirational goal: In Release 2 we will have support for a Generic Service-Level (“top level”) flow that handles both Allotted Resources and PNFs, as well as integration with SNIRO.
As per the Node Type dependencies captured in the vCpeResCust TOSCA, first make the TunnelXConn network assignments.

1. **Create Generic Allotted Res Instance (Type=TunnelXConn, Input=BRG_WAN_MAC_Address)**
   - The "Custom" TunnelXConn Resource-Level flow "just knows" that the "BRG_WAN_MAC_Address" from the Resource Level Input must be sent to SDNC in the "Assign" request. In the future, this determination should be made based on the TunnelXConn Model.

2. **Assign Allotted Resource (UUID)**
   - Assign Allotted Resource Network Resources (Type=TunnelXConn, UUID, Params={BRG_WAN_MAC_Address})
   - SO will associate the TunnelXConn object with both the vCpeResCust service instance and the vGMLX_Wan Service Instance in A&AI.

3. **Get BRG_WAN_IP**
   - The Network Controller will assign a VNI value from the associated vGMLX's VNI Pool. It will also look up the vGMLX_Bearer_IP and vGMLX_LAN_IP of that vGMLX and make it part of the TunnelXConn context. The associated vGMLX for this TunnelXConn is known from the Pending Solution. In addition, SDNC uses the BRG_MAC_Address as the index into the BRG Controller local inventory to find the BRG_WAN_IP value.

4. **Create Allotted Resource (UUID)**
   - The action of configuring the vGMLX affects the creation of the Allotted Resource

5. **Configure (XConn: (BRG_WAN_IP, VNI))**
   - Update Allotted Resource instance inventory Object (UUID)

6. **Success**
vCpeResCust Resource Level Processing:

- vG VNF
vCpeResCust Resource Level Processing:

- **BRG PNF**

Future:
- Add step to get monitoring in place at the Service and Resource level (via REST call)
- Add step to get Security in place
- Add step to notify turn up to Ops; need to define a REST call to an external system.

We will defer this alternative until Release 2 or beyond.

In Release 1 Use Case we will support only this alternative.