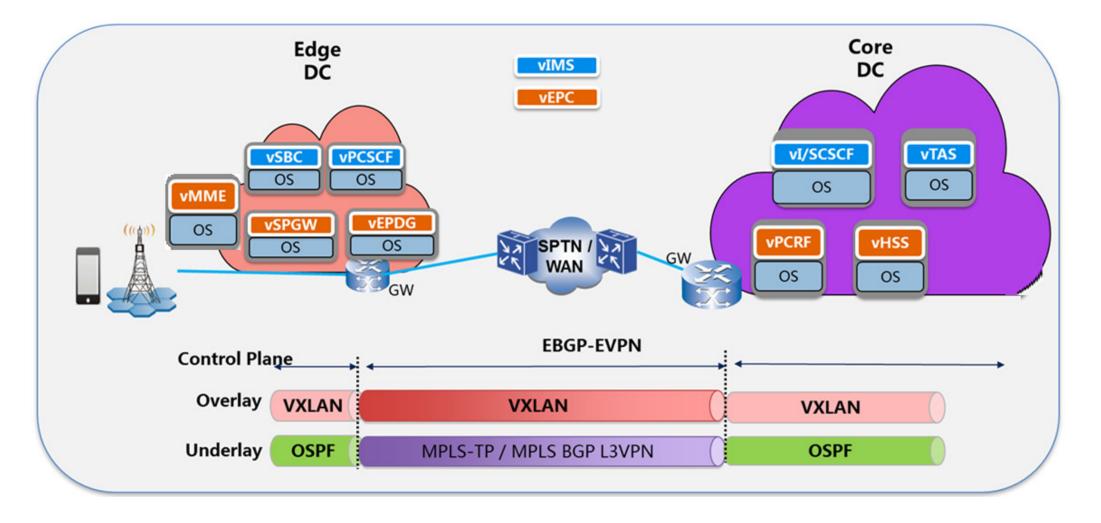
High Level Summary Points to Introduce the Sequence Diagrams

- The main SO participants in the sequence diagrams are a "service level" workflow, referred to as the "E2E Service Level Workflow" in the diagrams, and a "VNF-level" workflow referred to as the "Create VNF Instance BB", where "BB" stands for "Building Block".
- The VNF-level workflow execution threads across all the various VNF types share the same participant lifeline because they all are instances of the same generic SO BPMN based "building block" workflow. This building block's runtime behavior is model-driven, where the model is an instance structure derived from the corresponding VNF's TOSCA service template.
- In a similar manner, the E2E Service-level workflow is a generic SO BPMN based workflow whose runtime behavior is modeldriven, where the model is an instance structure derived from the corresponding Service's TOSCA service template.
- The functionality performed by the "Create VNF Instance BB" WF-level SO workflow includes much of that which is performed by the "VF-C" participant in the "Instantiate" sequence diagram better appears in the "Workflows" section above on this wiki page. Thus one can think of the "Create VNF Instance BB" VNF-level SO workflow has having genericized this functionality and pulled it into the SO component.
- Recognizing that different Service Providers may want to model VoLTE differently, I have provided two different TOSCA modeling
 alternatives. The sequence diagrams are organized according to these TOSCA models. Note, however, that irrespective of the
 TOSCA model, the sequence diagrams look basically the same. This is intended to illustrate the point that the workflows are
 intended to be implemented in a "generic" manner, and not specific to any particular Service or VNF type. The model-driven
 behavior of ONAP as illustrated in the sequence diagrams would support either TOSCA modeling approach with no code impacts.
- The sequence diagrams assume that there is a private VPN already established in the WAN, and that network already has presence at each Cloud Zone via a set of pre-configured VLAN tags. Thus, all that is required is to assign one of these tags to each instantiated VNF, and to configure the Compute Host via HEAT to access this network via that VLAN tag.
- Note that the TOSCA models interpret the VoLTE network VNF architecture such that vMME is considered to be part of the vEPC "Edge".

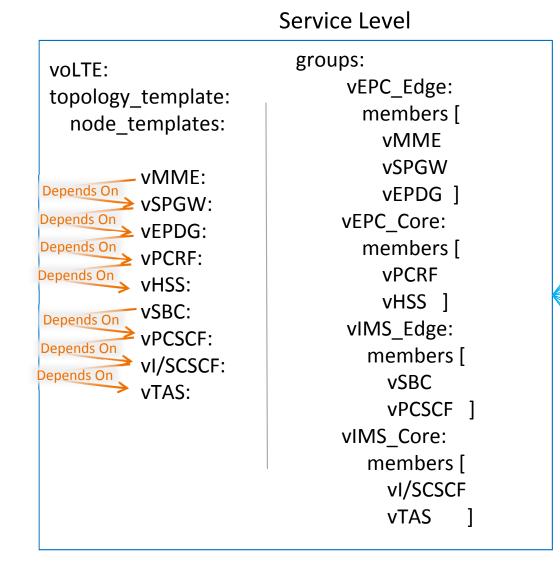
VoLTE Use Case Variation Used in These Examples

In the associated sequence diagrams, we assume that there is a private VPN already established in the WAN, and that network already has presence at each Cloud Zone via a set of pre-configured VLAN tags. All that is needed is to assign one of these tags to each instantiated VNF, and to configure the Compute Host via HEAT to access this network via that VLAN tag.

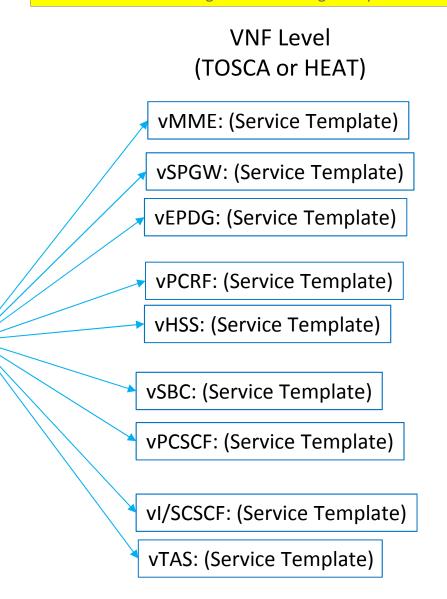


Note that this differs from the ONAP wiki diagram in that vMME has been moved to the Edge DC

Modeling Example A: VoLTE Modeled as a Service

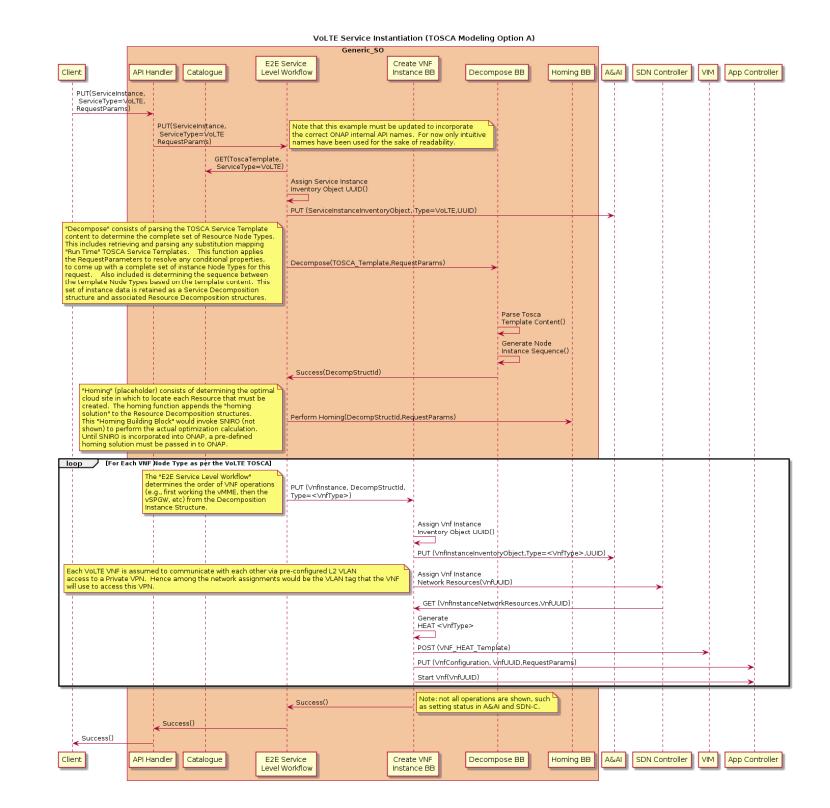


In this example "A", VoLTE itself is modeled as a Service. Thus, creation of an instance of "VoLTE" would entail creating instances the vIMS Core and Edge as well as the vEPC Core and Edge. See modeling example "B" for an alternative approach.



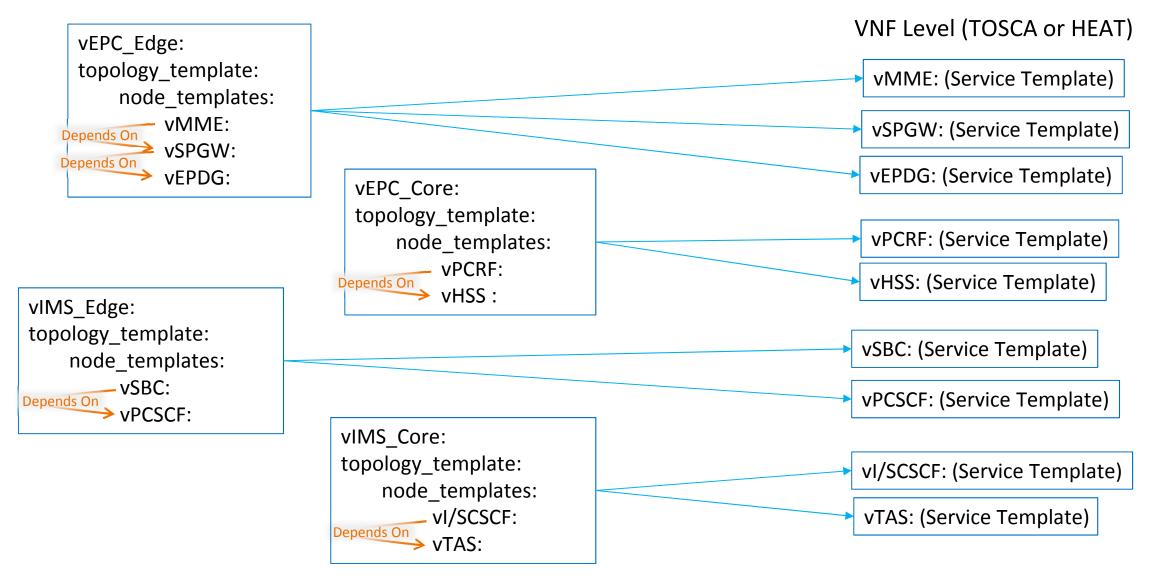
VoLTE Service Instantiation (TOSCA Modeling Example A)

This sequence diagram illustrates SO behavior driven from a VoLTE service instantiation request, assuming TOSCA modeling example A



Modeling Example B: vEPC and vIMS Edge/Core Modeled as Services

In this example "B", vIMS Core et al are modeled as individual Services. We want to show this example because we believe some service providers will want to reuse this infrastructure for wireline and so would not want to tightly couple them to a VoLTE Service.



Service Level

vIMS Service Instantiation (TOSCA Modeling Example B)

This sequence diagram illustrates SO behavior driven from a vIMS service instantiation request, assuming TOSCA modeling example B.

vIMS_Edge Service Instantiation (TOSCA Modeling Option B) Generic SO E2E Service Create VNF Client API Handler Catalogue Level Workflow Instance BB Decompose BB Homing BB A&AI SDN Controller VIM App Controller PUT(ServiceInstance, ServiceType=vIMS_Edge, RequestParams) PUT(ServiceInstance, Note that this example must be updated to incorporate ServiceType=vIMS_Edge the correct ONAP internal API names. For now only intuitive RequestParams) names have been used for the sake of readability. GET(ToscaTemplate, ServiceType=vIMS_Edge) Assign Service Instance Inventory Object UUID() PUT (ServiceInstanceInventoryObject, Type=vIMS_Edge,UUID) "Decompose" consists of parsing the TOSCA Service Template content to determine the complete set of Resource Node Types his includes retrieving and parsing any substitution mapping "Run Time" TOSCA Service Templates. This function applies the RequestParameters to resolve any conditional properties, Decompose(TOSCA Template,RequestParams) to come up with a complete set of instance Node Types for this request. Also included is determining the sequence between he template Node Types based on the template content. This set of instance data is retained as a Service Decomposition structure and associated Resource Decomposition structures Parse Tosca Template Content() $\overline{}$ Generate Node Instance Sequence() -Success(DecompStructId) "Homing" (placeholder) consists of determining the optimal cloud site in which to locate each Resource that must be created. The homing function appends the "homing solution" to the Resource Decomposition structures. Perform Homing(DecompStructId,RequestParams) This "Homing Building Block" would invoke SNIRO (not shown) to perform the actual optimization calculation Until SNIRO is incorporated into ONAP, a pre-defined homing solution must be passed in to ONAP. loop / [For Each VNF Node Type as per the vIMS_Edge TOSCA] The "E2E Service Level Workflow" determines the order of VNF operations PUT (Vnfinstance, DecompStructid, (e.g., first working the vMME, then the Type=<\/nfType>) vSPGW, etc) from the Decomposition Instance Structure. Assign Vnf Instance Inventory Object UUID() PUT (VnfinstanceInventoryObject,Type=<VnfType>,UUID) Each VIMS_Edge VNF is assumed to communicate with each other via pre-configured L2 VLAN Assign Vnf Instance access to a Private VPN. Hence among the network assignments would be the VLAN tag that the VNF Network Resources(VnfUUID) will use to access this VPN. GET (VnfInstanceNetworkResources,VnfUUID) Generate HEAT <\/nfType> < POST (VNF HEAT Template) PUT (VnfConfiguration, VnfUUID,RequestParams) Start Vnf(VnfUUID) Note: not all operations are shown, such Success() as setting status in A&AI and SDN-C. Success() Success() Client API Handler Catalogue E2E Service Create VNF Decompose BB Homing BB A&AI SDN Controller VIM App Controller Level Workflow Instance BB

vEPC Service Instantiation (TOSCA Modeling Example B)

This sequence diagram illustrates SO behavior driven from a vEPC service instantiation request, assuming TOSCA modeling example B.

