

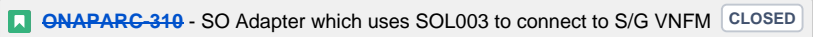
SO Plug-in Support for VNFM (SO VNFM Adapter)

- **Resource commitment:**
 - Ericsson: primary contact: [Byung-Woo Jun](#)
- **Usecase Lead:**
 - Ericsson: [Byung-Woo Jun](#)
- **TSC Contact:**
 - Ericsson: [Stephen Terrill](#)
- **Participating ONAP Projects:**
 - Implementation: SO, AAI, SDC (not in Dublin)

- **SO-ETSI-VNFM Adapter for Dublin Presentation slide deck at ONAP Paris 2019**



- **Associated JIRA tickets**

- JIRA ONAPARC-310  (SO Adapter which uses SOL003 to connect to S/G VNFM)
- JIRA SO-1508 (SO SOL003 plugin support to connect to an external VNFM): Epic
- See the user stories below for other JIRA tickets

- **SO-ETSI Alignment Use Cases for Dublin**

- Leverage ETSI standards for VNF LCM in SO
- Build SO VNFM Adapter
 - Use SOL003 APIs (2.5.1) for VNF LCM
 - Support operations such as create, instantiate, grant, terminate, delete, LCN subscription and LCN
- Enhance SO BPMN workflows & recipes
 - VNF-level Building Block workflows, leveraging VNFM Adapter
 - Passing VNF LCM requests to VNFM using VNFM Adapter

- Note: the followings are candidates for the EI Alto release.

- Provide VNF package management for VNFM
- Policy-based VNF scaling thru VNFM Adapter
- Support of remaining SOL003 APIs
- VNF Package handling (Download & Parse VNF Package)
 - Get package files from the SDC repository thru SO
 - Provide VNF package(s), VNFDs and Artifacts to VNFM
 - SO Catalog DB enhancement for SOL001/SOL004 is identified as future release work

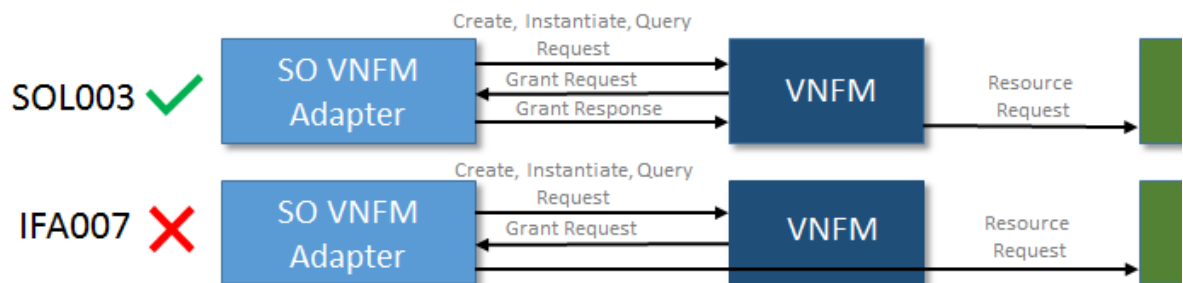
- **SO VNFM Adapter Requirements for Dublin**

- A new SO sub-component, following ONAP Microservice Architecture
- A Generic VNFM Adapter, supporting SOL003-compliant SVNFM
- Support of SOL003 APIs for VNFM LCM

- Invoking SVNFM based on SOL003 VNF LCM APIs as a client
 - use <https://forge.etsi.org/gitlab/nfv/SOL002-SOL003/2.5.1/master/src/SOL003/VNFLifecycleManagement> swagger to generate a client
 - support Create VNF, Instantiate VNF, Terminate/Delete VNF operations as a client
 - collect data for SOL003 API parameters from SDNC, A&AI and OOF (for models with homing: OOF-based granting might be supported in EI Alto)
 - Granting, based on ETSI VNFLifecycleOperationGranting
 - use <https://forge.etsi.org/gitlab/nfv/SOL002-SOL003/tree/master/src/SOL003/VNFLifecycleOperationGranting> swagger to generate grant services
 - Grant decisions based on the data either from 1) OOF (based on location, inventory data, resource availability, business rules, etc.) or 2) VIM registration, cloud region, etc.
 - In Dublin, the option #2 will be supported first.
 - Subscription to SVNFM for getting notifications
 - STARTING, PROCESSING, COMPLETED
- SVNFM selection based on configuration values that are configured during VNF on-boarding and VNFM registration. Two methods are considered:
 - Correlation between VNF NF Type and VNFM Type (Nokia method)
 - Utilizing VNFD vnf_info:type, VNFM registration values: VNFM type, Cloud region, vendor

• SVNFM Requirements for Dublin (SVNFM Vendor Responsibilities)

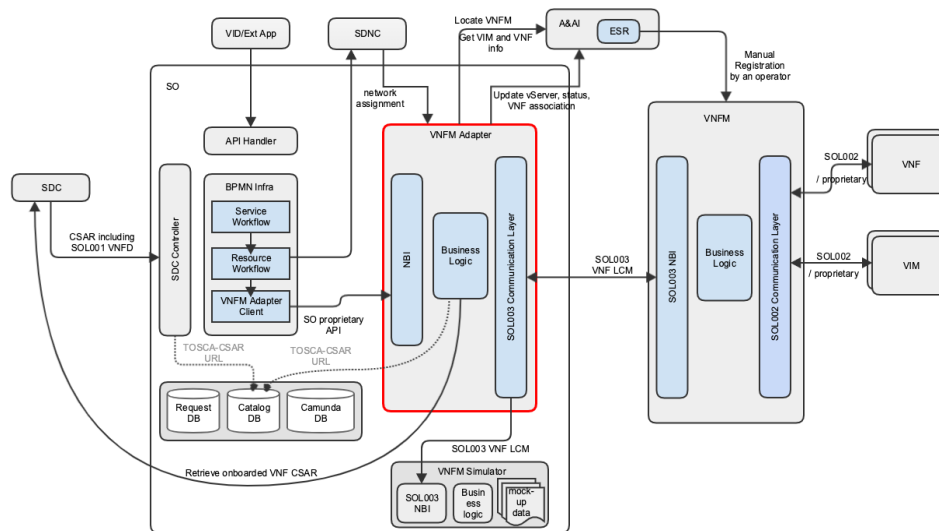
- Vendor SVNFM must be "SOL003-compliant"
- Providing SOL003 APIs for VNFM LCM, based on ETSI VNFLifecycleManagement
 - Use <https://forge.etsi.org/gitlab/nfv/SOL002-SOL003/2.5.1/master/src/SOL003/VNFLifecycleManagement> swagger for providing services
 - Create
 - Instantiate
 - Grant request to SO VNFM Adapter, as a client
 - Life cycle notification
 - Terminate/Delete
- Registration itself to ONAP (thru A&AI ESR) - Name, Type, Vendor, Version, URL, VIM, Username and Password
- Providing Subscription Services for Life-cycle Management Notifications
- Due to SDC SOL004 package support issues in Dublin, manual onboarding VNF Packages are needed for SVNFM.*
- Support of the "Direct Mode" of Resource Management only
 - After receiving a grant permission, the VNFM sends requests for resources directly to VIM**
 - Invoking MultiCloud from VNFM is under discussion, but not for Dublin
 - The "Indirect Mode" of Resource Management is being discussed, but not for Dublin



• VNFM Adapter Component Architecture

- The following diagram depicts the component architecture.
- The VNFM Adapter will be a SO sub-component; packaged as a docker and running in a container.
- VNFM will register into A&AI ESR and VNFM Adapter will locate a proper VNFM based on VNF NF Type or others.
- Communications between the VNFM Adapter and SVNFM will be SOL003 API-based.
- Communications between the SO BPMN Infra VNFM Adapter REST Client and the VNFM Adapter NBI will be SO-specific; i.e., it does not follow ETSI standards at this time.
- SO SDC Controller receives SOL001/SOL004-based CSAR from SDC, and stores the CSAR reference URLs to TOSCA_CSAR database table (see the note below).
 - Currently, VNF package output from SDC is not ETSI alignment. If SDC does not support it Dublin, use the standard package under the artifact directory (e.g., Artifact/Deployment/Others)
 - In Dublin release, there could be two CSAR file contents: one for the original CSAR and one for ONAP-compliant CSAR (maybe the ONAP-compliant CSAR includes the original package)
 - It is assumed that SDC keeps the original vendor VNF packages in their repository, and VNFM Adapter retrieves the original vendor CSAR files from the repository.
 - Note: if SDC does not support SOL004 VNF package in Dublin, the SO VNFM Adapter will retrieve the VNFD from SDC directly, bypassing SO TOSCA_CSAR database table.
- New VNF-level workflows that use VNFM Adapter will be implemented, and these new workflows will be invoked from 1) the (Network) Service-level workflows based on VNF type and other criteria or 2) the al carte VNF invocation from VID.
 - In Dublin, the second option will be supported.

- Unlike the existing SO VNF/VF-Module workflows, the new ETSI VNF-level workflows will NOT interact with OpenStack directly. The workflows will delegate the VNF LCM requests to VNFM Adapter, and VNFM Adapter will delegate the requests to VNFM further. Then, VNFM will interact with VIM directly. In Dublin release, the direct mode of resource management will be supported as depicted above.



• SOL001/SOL004 Standard Conformance

◦ VNF Package

- A VNF Package is a compressed file that contains the following:
 - One VNF descriptor (VNFD)
 - One or more Software Image files
 - Zero or more manifest files
 - Other files
- Package structure is SOL004 2.5.1 compliant.
- Note: SDC in Dublin has limitations and restrictions on SOL004 support:
 - SDC does not allow custom directories such as Images, Scripts, Licenses and HOT in the root directory.
 - SDC requires MainServiceTemplate.mf and MainServiceTemplate.yaml in the root directory
 - As a result, in Dublin, the vendor VNF package should follow SDC onboarding rules. These SDC restrictions and limitations plan to be removed in the EI Alto release.

◦ Cloud Service Archive (CSAR) format

- It is a packaging construct defined in SOL004 2.5.1, identified by a .csar suffix on the package file. In SOL004, there are two package options; one with TOSCA-Metadata, one without TOSCA-Metadata directories:
 - The CSAR file does not contain TOSCA-Metadata directory, the descriptor yaml file is in the root directory of the CSAR.
 - The CSAR file contains TOSCA-Metadata directory, the TOSCA meta file in this directory contains the location and name of the descriptor file denoted by Entry-Definitions
- In ONAP, the second option (with TOSCA-Metadata) is supported.
- Note: in SDC EI Alto, if the VNF package with certificate and/or signature will be packaged as a zip file. The csar format continues to be used for the package without certificate and/or signature. The zip file without certificate and/or signature will be considered as an HEAT-based package.

◦ VNFD

- It is specified by the SOL001 2.5.1.
- Note 1: that the input and get_input function would be used by a TOSCA orchestrator at run time to access the selected input parameter. If the deployment is not done by a TOSCA orchestrator, the inputs and get_input function may not be needed. The VNFD design should follow the vendor SVNFM orchestration capabilities.
- Note 2: in Dublin, SDC will convert SOL001 VNFD to SDC AID DM, but it is not complete. More mapping design discussions are necessary in EI Alto.

• VNFM Adapter conforms to the SOL001/SOL004 standards of specification and package management and SOL003 lifecycle operations.


- Note: in Dublin, SDC does not support SOL004 VNF package yet. SO and SO VNFM Adapter design around SOL004 handling is simplified or deferred to the EI Alto release.

• Design Scope for Dublin





- Epic and Use Stories
- VNFM Adapter Design
- SOL001/SOL004 Support & Design
- SO BPMN Infra & VNFM Adapter Run-time Scenario
- SOL003 API Support











- SO VNFM Adapter SOL003 API Support Design
- VNFM Adapter VNF Package Management (Not part of Dublin)
- SDNC Assignment Management
- VNFM Adapter Locating SVNFM
- VNF Life-cycle Granting
- VNFM Adapter Homing Decision for VNF Granting (TBD)
- SVNFM Simulator








• Epic

Epic	Feature	Description
 SO-4508 - ETSI Alignment - SO SOL003 plugin support to connect to external VNFM's CLOSED	ETSI Alignment - SO SOL003 plugin support to connect to external VNFM's	<p>ETSI Alignment - SO SOL003 plugin support to connect to an external VNFM.</p> <ul style="list-style-type: none"> • Leverage ETSI standards for VNF LCM in SO • Build SO VNFM Adapter <ul style="list-style-type: none"> ◦ Use SOL003 APIs (2.5.1) for VNF LCM ◦ Support operations such as create, instantiate, grant, query, terminate/delete, LCN subscription, LCN and VNF package management ◦ Support of Delete VNF is a stretch goal in Dublin • Enhance SO BPMN workflows & recipes <ul style="list-style-type: none"> ◦ VNF-level and VF-Module workflows, leveraging VNFM Adapter ◦ Passing VNF LCM requests to VNFM using VNFM Adapter • Provide VNF package management for VNFM (Stretch Goal; under investigation)

• User Stories

User Stories	Feature	Description
 SO-4538 - Integration Test for SO VNFM Adapter - Perform the functional test to validate VNFM Adapter NBI and SOL003-based SBI CLOSED	Create the Functional test case to validate VNFM Adapter NBI and SOL003-based SBI	Validate VNFM Adapter NBI and SOL003-based SBI
 SO-4618 - SVNFM Simulator CLOSED	SVNFM Simulator	<p>For integration testing in ONAP, vendor-neutral SVNFM is needed, This SVNFM Simulator supports SOL003-based interfaces and message exchange sequences for interface verification.</p> <ul style="list-style-type: none"> • vCPE VNF packages plan to be used for this validation testing.
 SO-4619 - Create SO VNFM Adapter Northbound Interface using Swagger CLOSED	Create SO VNFM Adapter Northbound Interface using Swagger	Create SO VNFM Adapter Northbound Interface using Swagger
 SO-4620 - Create Shell Adapter CLOSED	Create Shell Adapter	<ul style="list-style-type: none"> • Deployable VNFM Adapter container in ONAP (including docker image and helm chart) • Register VNFM Adapter with MSB
	Create placeholder implementation for create VNF (without	<ul style="list-style-type: none"> • Create Override YAML in OOM project

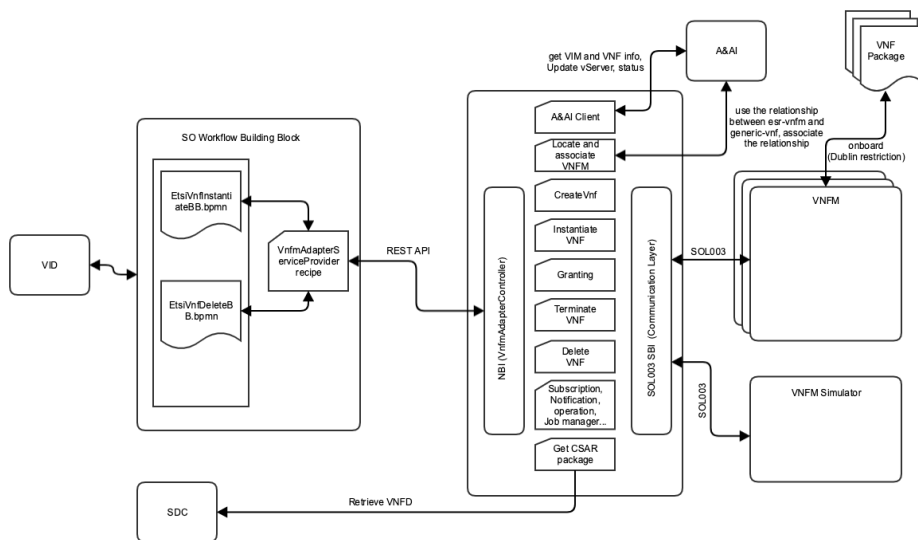
 SO-1621 - Create placeholder implementation for create VNF (without SVNFM interaction) CLOSED	SVNFM interaction)	<ul style="list-style-type: none"> Define Create/instantiate VNF interface which log the request in SVNFM adapter Create request to VNFM adapter for VNF creation <p><u>Note: manual database update to trigger new BB flow and no pre-load</u></p>
 SO-1622 - Check for existing VNF (with SVNFM Interaction) CLOSED	Check for existing VNF (with SVNFM Interaction)	<ul style="list-style-type: none"> Update Override YAML to add A&AI basic auth and URL Generate 003 APIs using swagger Get the generic-vnf from A&AI Select a VNFM from A&AI (if not already associated with a VNFM) Check for existing VNF
 SO-1623 - Handle Create VNF request in VNFM adapter CLOSED	Handle Create VNF request in VNFM adapter	<ul style="list-style-type: none"> Get VNFD Id from original csar Send create request to the SVNFM Set self-link based on result of create operation
 SO-1624 - Instantiate VNF (with SVNFM Interaction) CLOSED	Instantiate VNF (with SVNFM Interaction)	<ul style="list-style-type: none"> With pre-load data from SDNC based on model name and VNF-type Get the flavor Id from the CSAR Get the VIM info from A&AI Send request to SVNFM Update generic-vnf orchestration status A&AI
 SO-1625 - Handle Grant Request (Without Homing/OOF) CLOSED	Handle Grant Request (Without Homing/OOF)	Reply to grant request based on given VIM info in request
 SO-1626 - Monitor Node Status CLOSED	Monitor Job Status	<p>Monitor Job Status</p> <ul style="list-style-type: none"> Adapter Store and return job Id (job ids stored in cache) Introduce Job monitoring handling in flow Handle time out for Job monitoring (hard coded/configure in yaml timeout) Identify the VNFM and operation Id for the job Send get operation status request to VNFM Return status
 SO-1627 - Create relationship between esr-vnfm and generic-vnf in AAI CLOSED	Create relationship between esr-vnfm and generic-vnf in AAI	<p>Create relationship between esr-vnfm and generic-vnf in AAI</p> <ul style="list-style-type: none"> add a rule to AAI DBEdgeRule ESR <pre>{ "from": "generic-vnf", "to": "esr-vnfm", "label": "tosca.relationships.DependsOn", "direction": "OUT", "multiplicity": "MANY2ONE", "contains-other-v": "NONE", "delete-other-v": "NONE", "prevent-delete": "NONE", "default": "true", "description":"" }</pre> <ul style="list-style-type: none"> read the relationship in the SO VNFM Adapter Adapter
 SO-1628 - Handle Notification Subscription CLOSED	Handle Notification Subscription	<p>Notification Subscription</p> <ul style="list-style-type: none"> Update generic-vnf status Create vServers Set OAM IP address - source of which needs to be configurable Update Orch status in A&AI to completed
 SO-1629 - Notification Handling - Instantiate CLOSED	Notification Handling - Instantiate	<p>Notification Handling - Instantiate</p> <ul style="list-style-type: none"> Update generic-vnf status Create vServers Set OAM IP address - source of which needs to be configurable Update Orch status in A&AI to completed
 SO-1630 - Monitor Job Status-Create CLOSED	Monitor Node Status	<p>Monitor Node Status</p> <ul style="list-style-type: none"> Introduce Node monitoring handling in flow which periodically check orchestration status in A&AI Handle time out for node status handling (hard coded/configurable timeout)

 SO-1631 - VNFM Simulator Enhancement and Refactoring CLOSED	Handling Homing in Flow	Handling Homing in Flow
 SO-1632 - Handle VNF delete and termination (without SVNFM integration) CLOSED	Handle VNF delete and termination (without SVNFM integration)	Deleting/Terminating VNF (without SVNFM integration) <ul style="list-style-type: none"> Define Terminate/Delete VNF interface in VNFM adapter Update or introduce new building block which invoke VNFM adapter for termination
 SO-1633 - Terminate VNF (with SVNFM interaction) CLOSED	Terminate VNF (with SVNFM interaction)	Terminate VNF (with SVNFM interaction) <ul style="list-style-type: none"> Identify the SVNFM to use from A&AI Send terminate request to SVNFM Send delete request to SVNFM Return a job Id Check termination job status in flow
 SO-1634 - Notification Handling - Terminate CLOSED	Notification Handling - Terminate	Notification Handling - Terminate <ul style="list-style-type: none"> Delete vServers Update generic-vnf orchestration status Check node termination status in flow
 SO-1635 - Preload using user_param (without UI changes) CLOSED	Remove SDNC pre-load and introduce user_param handling	Remove SDNC pre-load and introduce user_param handling
 SO-1636 - SOL003 Adapter - VNF Instances Query Support CLOSED	Handle Failure case where notification is missed (Query VNF)	Handle Failure case where notification is missed (Query VNF) <ul style="list-style-type: none"> VNFM Adapter expose interface to get of VNF info Flow use VNF info to check status at timeout
 SO-1637 - OAM IP address handling for generic-vnf CLOSED	Spike - investigate OAM IP address handling for generic-vnf	Investigate OAM IP Address handling for generic-vnf

• Testing

- Ericsson Internal Test:
 - A vendor provides their VNF Package and SVNFM for managing the vendor-specific VNF package.
 - For the SO VNFM Adapter NBI and SOL003-SBI interface validation, Ericsson SVNFM is used for the internal testing
- Integration Test:
 - For the integration testing, generic/dummy VNF package and VNFM simulator are provided.
 - The generic/dummy VNF package is used to extract SOL001 VNFD parameters for SOL003-based API parameters, such as descriptor_id, flavor, etc.
 - VNFM simulator is a vendor-neutral SOL003-compliant VNFM, which supports SOL003 responses and message exchanges.
 - vCPE VNF packages would be used for this integration testing.

• VNFM Adapter Sub-components



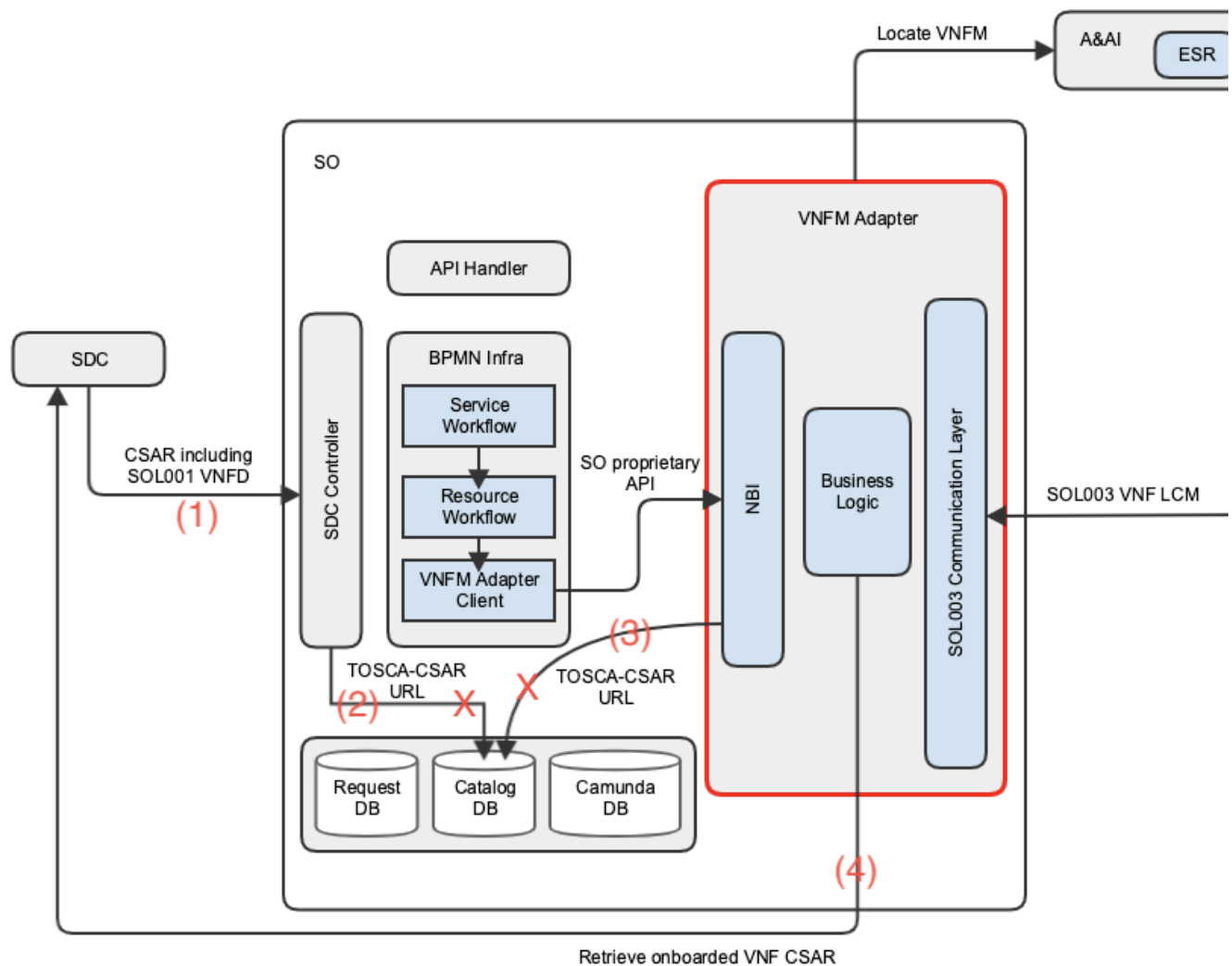
- SO VNFM Adapter component is a sub component of SO; utilizing docker image and container managed.
 - North Bound Interface (NBI)
 - RESTful APIs that support createVnf (invokes both createVnf and instantiateVnf), grantVnf, TerminateVnf/DeleteVnf
 - Its APIs are SO specific; i.e., not SOL003-based ones; for the NB API details, see the [SO VNFM Adapter APIs](#) page.
 - Business Logic layer
 - It is invoked by the NBI and provides business logic for createVnf, instantiateVnf, terminateVnf/DeleteVnf
 - SDNC and A&AI access to collect assignment and VimConnectionInfo
 - Accesses SdcPackageProvider for getting SOL004 package(s) and parameters
 - SdcPackageProvider
 - Supports SOL001/SOL004 package management
 - Provides getPackage, getVnfId, getFlavorId, getVnfNodeProperty
 - Provides getPackage(s), getVnf, getArtifactFile for SVNFM
 - Uses SDC Tosca Parser
 - GrantManager
 - Provides requestGrantForInstantiate REST API for SVNFM
 - Invokes OOF for homing decision; HPA support, and/or non-OOF decision
 - SOL003Lcn APIs
 - Supports VnfIdentifierCreationNotification, VnfIdentifierDeletionNotification, VnfLcmOperationOccurrenceNotification
 - SOL003 Communication Layer
 - It is a thin REST client layer, which sends SOL003-compliant requests to SVNFM and receives responses /notifications from SVNFM.
 - For Grant, it provides the Grant REST endpoint for SVNFM
 - For the SOL003 Southbound API details, see the [SO VNFM Adapter APIs](#) page.

• SOL001/SOL004 Support & Design

- **SDC VNF SOL001/SOL004 Support**
 - VNFM Adapter depends on the following SDC capabilities:
 - SOL004-based VNF CSAR package onboarding, including storing the original VNF package in SDC output.
 - Manual onboarding of VNF package thru SDC UI.
 - Mapping VNF (SOL001) to SDC AID DM, including VF-Module deduction based on ScaleAspect + Delta
 - SDC TOSCA Parser for SDC/ONAP Internal model
 - TOSCA Parser for SOL001
 - VNF SDK support of VNF package verification
 - SDC support scope for VNF SOL001/SOL004 is under discussion (Ericsson, Nokia contribution)
 - In Dublin, SDC does not support SOL004 VNF package fully.
 - The SDC package structure support is not flexible and requires proprietary convention.
 - SDC limitations and restrictions have been identified and under discussion for the EI Alto release.

• CSAR Import, Store and Retrieve Sequences

- SDC stores the original vendor VNF package along with the transformed ONAP-compliant package.
 - Note: this will not supported SDC Dublin.
- SO uses SDC-transformed CSAR packages and VNFM Adapter uses the original Vendor CSAR package.
 - Note: since SDC Dublin does not support the original vendor CSAR package, SO VNFM Adapter will retrieve VNFs from SDC.
 - For that reason, the following steps 1, 2, 3 and 4 have been simplified and adjusted.
- In Dublin, SVNFM needs to onboard its VNF packages.



Design

Note 1: the following design will be completed in the EI Alto release. For Dublin, the steps 2 and 3 are not used, and the step 4 will be used to retrieve VNFDs from SDC

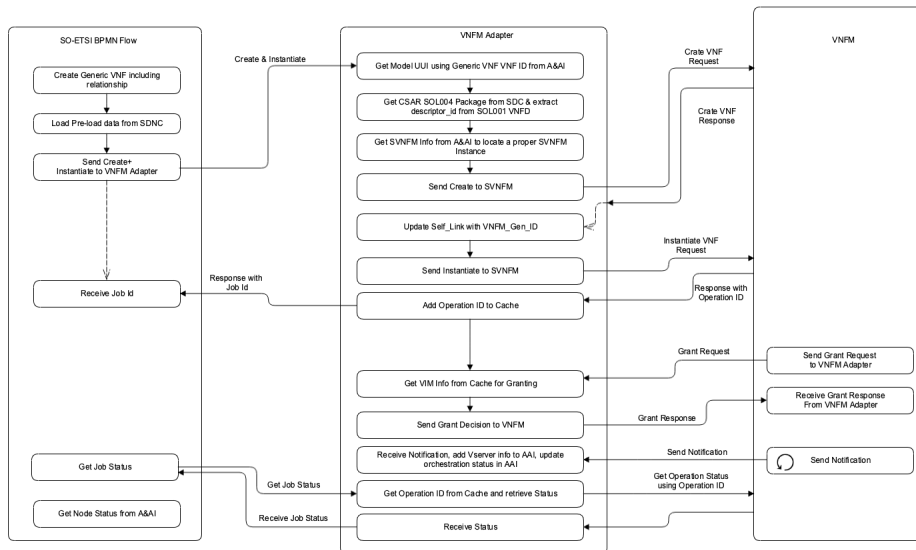
Note 2: the step 1 is not important for SO VNFM Adapter in Dublin since the Adapter will get VNFD from SDC directly.

- SO SDC Controller gets a SOL004 VNF package with an SOL001 VNFD (for EI Alto)
 - SDC could generate two output: one ONAP-compliant CSAR and one original CSAR (maybe the first file includes the second one)
 - SO will use the ONAP-compliant CSAR
 - VNFM Adapter will use original CSAR
- SO SDC Controller stores a VNF CSAR file reference to the SO Catalog DB (e.g., TOSCA_CSAR database table) - for EI Alto
- VNFM Adapter gets a CSAR package URL from the SO TOSCA_CSAR database table - for EI Alto
- VNFM Adapter gets an original CSAR package file from the SDC repository
 - It is assumed that the Adapter retrieves the original vendor provided CSAR package from SDC repository directory before it passes the package to SVNFM, where SVNFM handles the original CSAR. For that, SDC copy the full original package.
 - There would be two CSAR packages for a service: one original package, one SDC transformed package.
 - VNFM Adapter passes the original CSAR package to SVNFM because the SVNFM is outside of ONAP and is designed to handle the vendor CSAR package.

Note: SO future release could consider SOL001/SOL004 internal representation in its Catalog DB, or using the Run-time Catalog DB

• VNFM Adapter VNF Flow Design (Run-time Scenarios)

- The following diagram depicts VNF flows thru SO VNFM Adapter.



1. SO BPMN Service workflows dispatch new resource-level workflows based on VNF request parameters (e.g., type, others).
 - a. Once a Service workflow chooses a new workflow path for VNFM Adapter, the subsequent requests for the same VNF will follow the new path.
 - b. an association between VNF and VNFM will be made in A&AI.
2. SO BPMN VNF-level resource workflows handle:
 - a. Assign VNF to SDNC
 - b. Retrieve the VNF Assignment from SDNC
 - c. Invoke VNFM Adapter Client with required parameters
3. VNFM Adapter Client manages:
 - a. Populate parameter structures based on data from SO workflows
 - b. Invoke VNFM Adapter NBI with required parameters
4. VNFM Adapter gets GenericVNF from A&AI
5. VNFM Adapter locates the corresponding VNF and VNFM registration info from A&AI (ESR). Two methods are suggested
 - a. Current one: based on VNF NF Type and VNFM Type in A&AI
 - b. Could use VNFD vnf_info:type, VNFM registration values: VNFM type, Cloud Region, vendor - logic is being designed
6. VNFM Adapter gets VimConnectionInfo from A&AI
 - a. Queries A&AI based on the cloud region and tenant id
 - b. Builds the VimConnectionInfo based on the type, service-url, user-name, password, cloud-domain, etc.
7. VNFM Adapter uses network assignment (e.g., IP Address) from SO (thru SDNC) and builds the extVirtualLinks and other parameters.

• VF-Module Deduction from SOL001 (It is a candidate for the EI Alto release)

- Note: VF-Module deduction will not be supported in Dublin. ETSI-based scaling will not be supported in Dublin.
- There is an assumption that SDC transforms the vendor provided VNF package into ONAP-compliant one; i.e., deducing VF Modules based on VNFD ScalingAspects and Delta.
- If SDC supports the transformation in Dublin time-frame, the transformed CSAR will be imported to SO, and SO VNF- and VF-Module-level workflows will manage VNF and VF Module topology towards SDNC with the following changes - Input from Gil Bullard (AT&T)
 - Today the VNF-level workflow has an embedded per-VF Module loop that a) retrieves the SDNC assignments for that VF Module, and then b) sends those VF Module-level assignments down to the VIM (e.g., OpenStack); the loop then moves to repeat "a" with the next VF Module.
 - The new VNF-level flow will have the following sequences:
 1. an embedded per-VF Module loop that only retrieves the SDNC assignments for each VF Module; because the VIM is hidden from SO's sight, beneath the VNFM Adapter/VNF.
 2. After finishing the loop, the SO workflows will send a structure to the VNFM Adapter that includes the aggregate assignments at the VNF level.
 3. The VNFM Adapter aggregates all the VF-Module level assignments and transforms the assignment data into SOL003 API parameters before sending them to SVNF
 - a. The VNFM Adapter would need to be able to parse the VNF-level assignments structure received from SO to obtain the per-VDUconnection point assignments information and any per-VDU parameter information (e.g., hostnames)
 - b. In doing so, the VNFM Adapter would need to know to ignore the VF Module groupings of these assignments
 - c. Further know how to map the ONAP data structure and parameter names into the ETSI (e.g., VM=VDU, VNFC=VNFC, vNIC=vNIC, etc.). Note that the above assumes that in ONAP, as in ETSI, there will be a one-to-one correspondence between VM/VDU and VNFC.

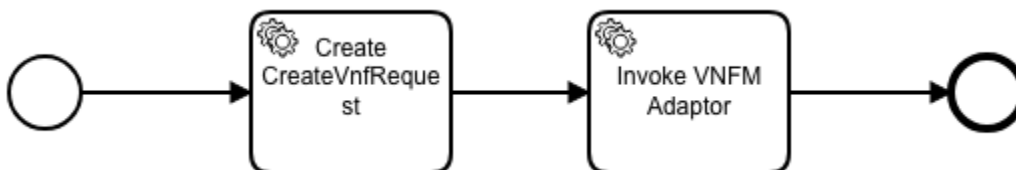
◦ Assumptions for deducing VF-Module from SOL001 (Gil Bullard's input)

- SOL001 concept of Aspect+ScalingDelta combination is one to one with the ONAP concept of VF Module.
- SOL001 concept of VDU is one to one with the ONAP concept of A&AI vServer

- SOL001 concept of a connection point associated with a VDU corresponds to the ONAP concept of the same name, as does the understanding of the meaning of "internal" versus "external" connection point.
- ONAP-compliant SOL001 VNF Vendors will be obliged to name their HEAT files using a naming convention that encodes the SOL001 Aspect+ScalingDelta names
- ONAP-compliant SOL001 VNF Vendors will be obliged to name their SOL001 Aspect+ScalingDelta parameters using a naming convention that readily maps to the corresponding HEAT properties.
- In addition, if AT&T has already deployed such a vendor's VNF into its network, the HEAT naming will remain invariant, and (at least) the (AT&T version of that) SOL001 be written to match it.
- What to do
 - ONAP will be extended to incorporate the constructs of Aspect and Scaling Level. This includes SDC's, SOs, and A&AI's modeling of these constructs and A&AI's ability to capture and SO's ability to set/update the "current scaling level" of a VNF for a given Aspect.
 - If ETSI in their SOL001 VNFD had defined a "ScalingDelta" in a straightforward manner, i.e., in terms of the VNFCs that comprise it, then it would be very easy to extract VF Module information from the VNFD. (I would like to see ETSI define "ScalingDelta" in this manner, as opposed to the current way they do so.) However, given that they did not, ONAP SDC would need to be extended to derive the VF Module "structure" from the SOL001 document through the algorithm below. "Structure" in this case includes the VDU topology, connection points and associated parameters. This algorithm will:
 - Determine the set of Aspects and corresponding VDUs and associated ScalingDeltas (step_deltas) from the SOL001.
 - Determine the set of ScalingLevels associated with each Aspect and the ScalingDeltas associated with each.
 - Translate the VDU-centric representation of ScalingDeltas (step_deltas) as per SOL001 to come up with a ScalingDelta-centric representation that captures the number and type of VDUs associated with that ScalingDelta across the various VDU types.
 - Create a VF Module object that corresponds to each ScalingDelta-centric representation of a ScalingDelta calculated above.
 - Fill in the details of the VF Module object based on the SOL001 data to represent the VDU connection points, associated Networks (internal or external), and associated Parameters.
 - Determine if there is an artifact in the SOL004 package that is a HOT YAML whose file name corresponds (through a VNF vendor obligatory naming convention) to the Aspect+ScalingDelta from which this VF Module object was derived. If so, associate that HOT file with the VF Module.
 - Name the VF Module based on a naming convention to capture the Aspect+ScalingDelta names
 - Determine and capture the mapping from each Aspect + ScalingLevel model for the VNF to the corresponding VF Module.
 - Given a desired state Aspect+ScalingLevel, will be able to calculate (from the SDC distributed mapping of Aspect+ScalingLevel to VF Module along with the current Scaling Level for this Aspect as per A&AI) the (ordered set of) VF Module(s) needed to take that VNF from the "current scaling level" to the desired level for that Aspect.
 - Note: As an aside, SDC enhancements are being discussed. It is not clear if the SDC changes will be available in the Dublin time frame. some "stubbing off" SDC with a simulator could be suggested to at least prove in the run-time aspects of the solution.

• SO BPMN VNF Workflow

- New VNF-level Building Block workflows will be created.
- Create and Instantiate VNF workflow (EtsiVnfInstantiateBB.bpmn)



- Delete VNF workflow (EtsiVNFDeleteBB.bpmn)



• SO BPMN VF-Module Workflow (not for Dublin)

- For ETSI-based VNF package, VF-Module level workflows will NOT be used.
- SO VNFM Adapter will interface with SVNFM at the VNF level.

• VNFM Adapter Client

- SO Workflows (Building Blocks) will invoke Java-based VNFM Adapter Client.
- This Adapter client will invoke the SO VNFM Adapter NBI thru SO proprietary interfaces.

• SDNC Assignment

- Even though SO Building Blocks are used for the SO VNFM Adapter, assignment data will be pre-loaded into SDNC for the Dublin release.
- In EI Alto release, preload data could be replaced with SO request data (TBD)
- Possible preload data structure. The additionalParams and extVirtualLinks data will be preloaded to SDNC, and the SO VNFM Adapter will retrieve the data from SDNC.

```

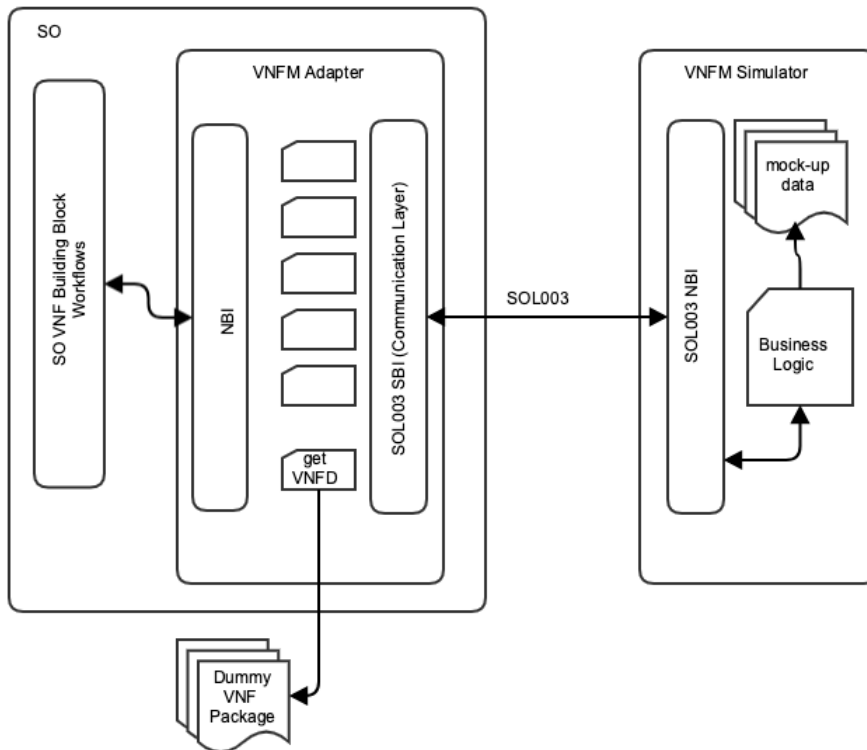
{
  "additionalParams": {"param_1": "value_1", "param_2": "value_2", "param_3": "value_3"},
  "extVirtualLinks": [{"id": "vlaue", "resourceId": "value", "extCps": [{"cpdId": "cpdId_value", "fixedAddresses": [{"macAddress": "macAddress_1", "macAddress_2"}]}]}]
}

```

• VNFM Simulator

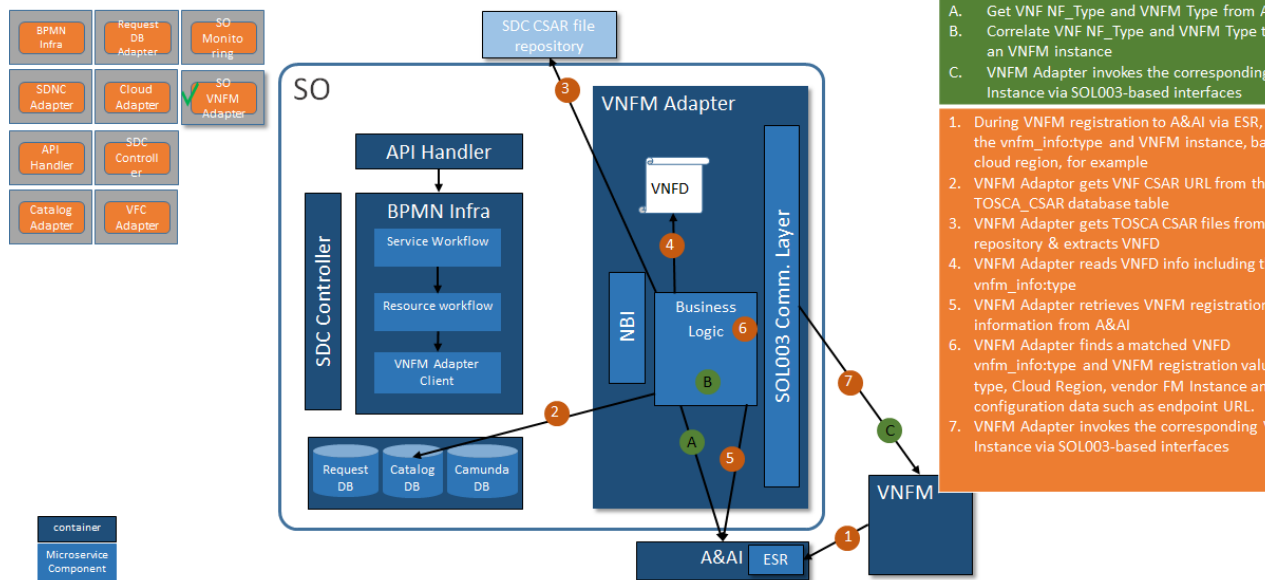
SVNFM is provided by its vendor, which is vendor-specific by nature. For ONAP integration testing for SO VNFM Adapter, a generic VNFM Simulator is needed. This generic VNFM Simulator will support SOL003-compliant interfaces.

In Dublin, the Simulator will support Create, Instantiate, Terminate and Delete VNF operations and Grant request/response handling. Simulated mock-up request and response data will be exchanged.



• VNFM Adapter Locating SVNFM

- VNFM Adapter locates a proper SVNFM based on VNF NF Type/VNFD and VNFM registration
- Two methods are suggested as follows, and one of them will be chosen.

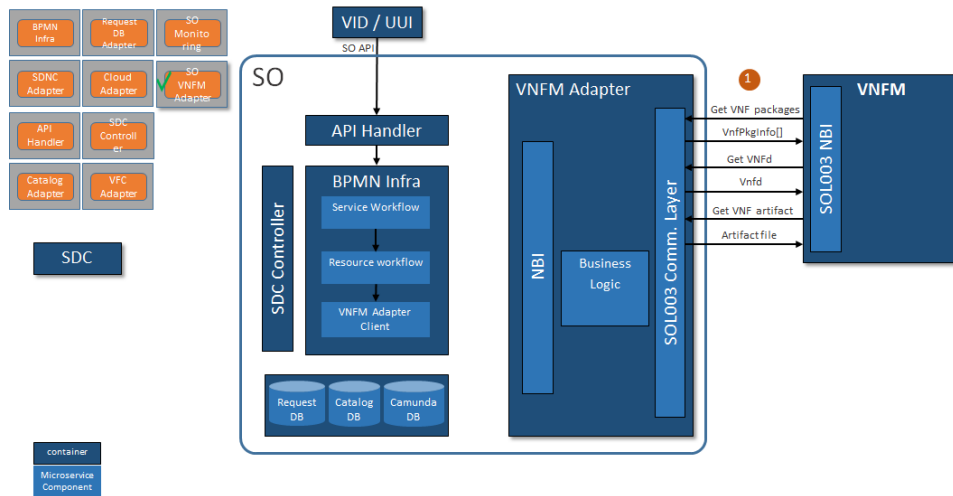


Design

- Current method (Green Dots)
 - VnfLCM::locateVnf(GenericVnf vnf)
 - Get a vnf list from AAI ESR
 - Find a matched Vnf, where vnfinfo.getType() == vnf.getNfType
- New method (Orange Dots)
 - Under development (TBD)

VNFM Adapter VNF Package Management (It is deferred to the next release, e.g, EI Alto)

- In Dublin, VNF packages will be onboarded into SVNFM separately/manually. The SVNFM will onboard SOL004 VNF package in their own way.
- In the future release (e.g., EI Alto), VNFM Adapter will support VNF package management for SVNFM.



VNF Package Management Execution Flow

- VNFM Adapter supports VNF Package Management Interface
 - Accepts the "Get VNF packages" request and returns "200 OK" with VnfPkgInfo[]
 - Accepts the "Get VNFD" request and returns "200 OK" with Vnfd
 - Accepts the "Get VNF artifact" request and returns "200 OK" with Artifact file

Design

Getting multiple VNF packages

- VNFM queries information about multiple VNF packages (VNFM VNFM Adapter); GET .../vnf_packages
- VNFM Adapter returns a "200 OK" response and includes in the payload body zero or more data structures of type "VnfPkgInfo" (VNFM Adapter VNFM); 200 OK (VnfPkgInfo[])

- **Getting a particular VNF Package**

- VNFM sends a GET request to the "Individual VNF package" resource (VNFM VNFM Adapter); GET ... /vnf_packages/{vnfPkgId}
- VNFM Adapter returns a "200 OK" response and includes in the payload body a data structure of type "VnfPkgInfo" (VNFM Adapter VNFM); 200 OK (VnfPkgInfo)

- **Reading the VNFD of an onboarded VNF Package**

- VNFM sends a GET request to the "VNFD in an individual VNF package" resource (VNFM VNFM Adapter); GET .../vnf_packages/{vnfPkgId}/vnfd
- VNFM Adapter returns a "200 OK" response and includes a copy of the VNFD from the VNF package in the payload body (VNFM Adapter VNFM); 200 OK (Vnfd)

- **Fetching a VNF package artifact**

- VNFM sends a GET request to the "Individual VNF package artifact" resource (VNFM VNFM Adapter); GET .../vnf_packages/{vnfPkgId}/artifacts/{artifactPath}
- VNFM Adapter returns a "200 OK" response, and includes a copy of the applicable artifact file from the VNF package in the payload body (VNFM Adapter VNFM); 200 OK (artifact file)

- **Note: Package management subscription and notification are not consider at this time, future consideration.**

- ******* Design *******

- **SOL003 Lifecycle Management (LCM) Support and Design**

- **SOL003 Interfaces between VNFM Adapter (Client) and SVNFM (Provider)**

Note: for the interface details, see the VNFM Adapter APIs,

[SO VNFM Adapter APIs](#)

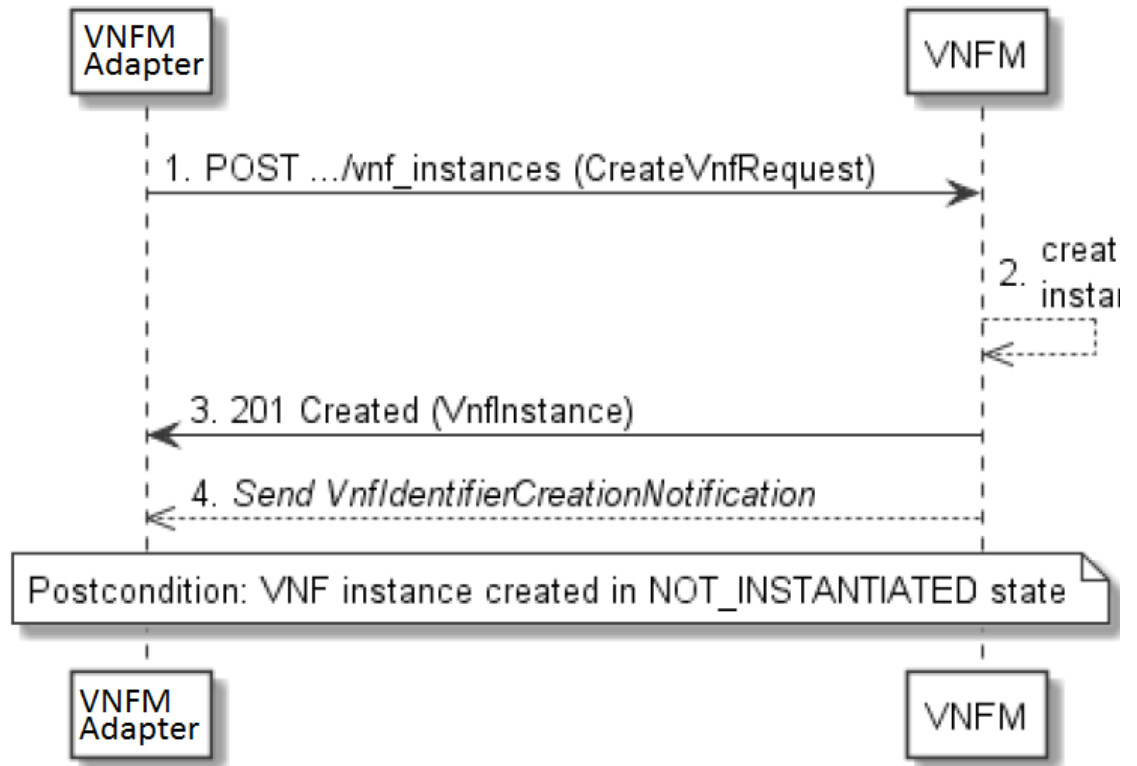
- **Create VNF**

- HTTP Method Type: POST
- VNFM Endpoint: /vnf_instances/
- Request Payload: CreateVnfRequest
- Response Header: 201 success
- Response Body: VnfInstance

- **Design**

- VNFM Adapter sends a POST request to the "VNF Instances" resource including in the payload body a data structure of type "CreateVnfRequest" (VNFM Adapter VNFM); POST ../vnf_instances (CreateVnfRequest)
- VNFM creates a new VNF instance resource in NOT_INSTANTIATED state, and the associated VNF instance identifier (VNFM VNFM Adapter);
- VNFM returns a 201 Created response containing a representation of the VNF Instance resource just created by the VNFM, and provides the URI of the newly-created resource in the "Location" HTTP header (VNFM VNFM Adapter); 202 Created (VnfInstnace)
- VNFM sends a VNF Identifier Creation Notification to the VNFM Adapter to indicate the creation of the VNF instance resource and the associated VNF instance identifier (VNFM VNFM Adapter); Send VnfIdentifierCreationNotification
- Parameters and data source

Parameters	Required?	Data Source	Note
vnfdId	Required	descriptor_id from VNFD	
vnfInstanceName	Optional	User Input	
vnfInstanceDescription	Optional	User Input	



▪ Instantiate VNF

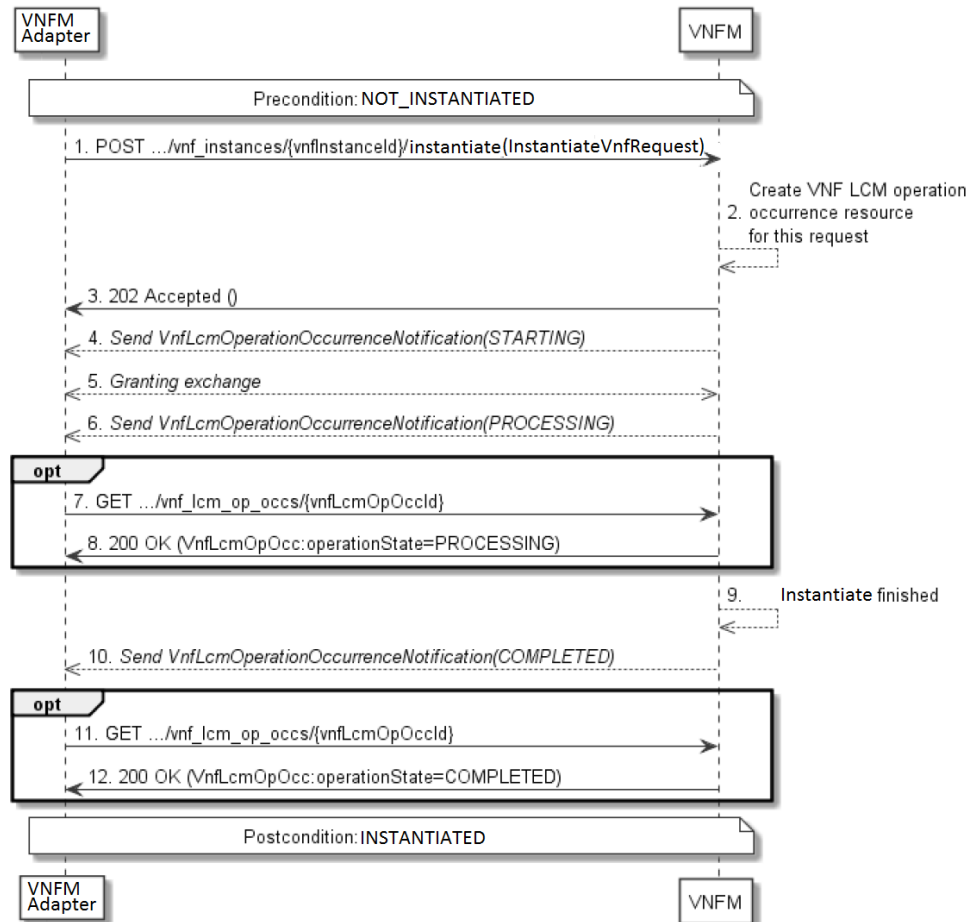
- HTTP Method Type: POST
- VNFM Endpoint: /vnf_instances/{vnfInstanceId}/instantiate
- Request Payload: InstantiateVNFRequest
- Response Header: 202 success
- Response Body: not applicable

• Design

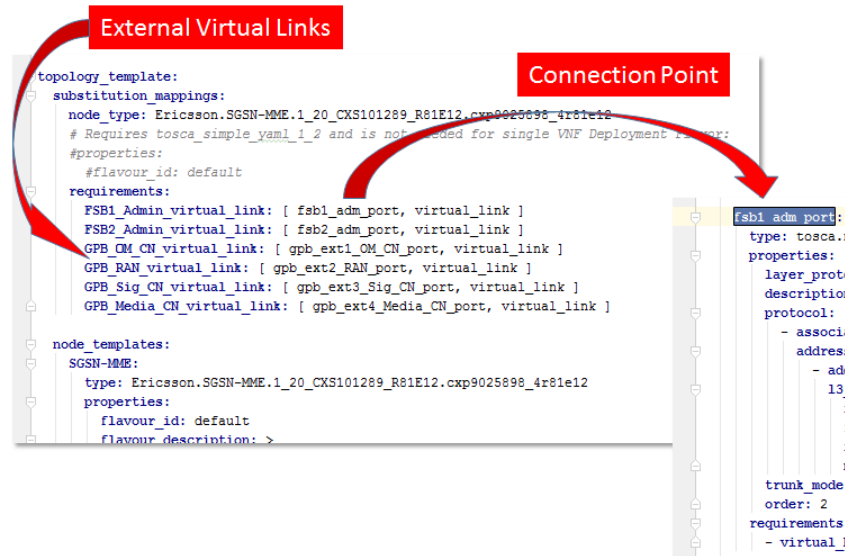
- VNFM Adapter sends a POST request to the Task resource that represents the lifecycle operation to be executed on the VNF instance, and includes in the payload body a data structure of type InstantiateVNFRequest (VNFM Adapter VNFM); POST ../vnf_instances/{vnfInstanceId}/instantiate
- VNFM Creates a "VNF LCM operation occurrence" resource for the request (VNFM VNFM Adapter)
- VNFM returns a "202 Accepted" response with an empty payload body and a "Location" HTTP header that points to the new "VNF LCM operation occurrence" resource (VNFM VNFM Adapter); ../vnf_lcm_op_occs/{vnfLcmOpOccId}
- VNFM sends to the VNFM Adapter a VNF lifecycle management operation occurrence notification to indicate the start of the lifecycle management operation occurrence with the "STARTING" state
- VNFM and VNFM Adapter exchange granting information (see Granting section)
- VNFM sends to the VNFM Adapter a VNF lifecycle management operation occurrence notification to indicate that the VNF LCM operation occurrence enters the "PROCESSING" state
- VNFM Adapter polls the "VNF LCM operation occurrence" resource to obtain information about the ongoing operation by sending a GET request to the resource that represents the VNF LCM operation occurrence.
- VNFM returns to the VNFM Adapter information of the operation, such as the operation status, by providing in the payload body a data structure of type "VnfLcmOpOcc"
- VNFM has finished the operation <<Operation>>
- VNFM sends a VNF lifecycle management operation occurrence notification to VNFM Adapter to indicate the completion of the lifecycle management operation occurrence with the success state "COMPLETED"
- Parameters and data source

Parameter	Required?	Data Source	Note
flavorId	Optional	From user input or from the VNFD	This parameter is optional for NBI but it is mandatory for southbound. The value from the user; otherwise it takes the default value from VNFD
instantiationLevelId	Optional	From VNFD	
extVirtualLinks	Optional	From preload data or user input	The user input requires UI enhancement - See below for design proposal If the external connection point ip_address_assignment is set to false, extVirtualLink is not necessary since the ip address is set by VIM dynamically.

extManagedVirtualLinks	Optional	from user input	Externally-managed internal VL; Not supported in Dublin
vimConnectionInfo	Optional	From AAI	In Dublin, the direct resource mode is supported, that means all the VIM resources are created directly by VNFM
localizationLanguage	Optional		Not supported in Dublin
additionalParams	Optional	From VNFD	It is a mechanism to pass vendor-specific parameters



- Population of User Inputs for SOL003 Instantiation. Two options are discussed and the Option #1 is chosen for Dublin for simplicity. It will be revisited for the EI Alto release.
 - Option #1. Preload configuration solution (it would be an option for the Dublin release)
 - For the VNFD, pre-configure the mapping between the external virtual links and the ip addresses
 - VNFM Adapter retrieves the mapping from preload data and fill up the extVirtualLink parameters based on the mapping
 - Option #2: model-driven user data population from UI (which is attached to VID) - deferred to the next release.
 - The external virtual links user input is shown for illustration purposes.
 - VNFD does not define external virtual links, but it lists the external virtual links as requirements for the VNF.
 - If the connection point ip_address_assignment is set to false, no extVirtualLinks ip address assignment is necessary.
 - In this case, VIM will assign IP addresses dynamically.
 - This could be an option for the Dublin release for simplifying the solution.



- If the connection point `ip_address_assignment` is set to true, set `extVirtualLink` ip address assignment with configuration data from the user input or a preload file.
 - UI solution (build an UI configuration page and invoke it from VID; it would be an option for the post Dublin release): Impact on VID (open issue)
 - Parse VNFD and extract a list of external virtual links
 - Map the external virtual links to the corresponding connection points, and read `ip_address_assignment` and `number_of_ip_address` value
 - Render the external virtual links
 - For each external virtual link, render the `ip_address_assignment` entry fields based on the `number_of_ip_address` value
 - User configures the mapping and the UI stores the mapping in the database
 - VNFM Adapter retrieves the mapping from database and fill up the `extVirtualLink` parameters based the mapping
 - UI Example:

External Virtual Link Configuration

External Virtual Link	Connection Point
FSB1_Admin_virtual_link	fsb1_adm_port +
FSB2_Admin_virtual_link	fsb2_adm_port +
GPB_OM_CN_virtual_link	gpb_ext1_CM_CN_port +
GPB_RAN_virtual_link	gpb_ext2_RAN_port +
GPB_Sig_CN_virtual_link	gpb_ext3_Sig_CN_port +
GPB_Media_CN_virtual_link	gpb_ext4_Media_CN_port +

External Virtual Link:Connection Point = 1:N
 Connection Point:IP Address = 1:N

• VNFM and GenericVNF relationship

- Trace which VNFM instance managed which VNF instance, the following rule will be added to A&AI DB Edge (DbEdgeRules_esr_v15.json, DbEdgeRules_esr_v16.json).
 - {
 - "from": "generic-vnf",


```

    "to": "esr-vnfm",
    "label": "tosca.relationships.DependsOn",
    "direction": "OUT",
    "multiplicity": "MANY2ONE",
    "contains-other-v": "NONE",
    "delete-other-v": "NONE",
    "prevent-delete": "NONE",
    "default": "true",
    "description":""
  }
}

```

- SO VNFM Adapter will update the relationship between generic-vnf and esr-vnfm based on the rule.

▪ Query VNF Instances (deferred to EI Alto)

- HTTP Method Type: GET
- VNFM Endpoint: /vnf_instances (for multiple VNFs), /vnf_instances/{vnfInstanceId} (for single VNF)
- Request Payload: not applicable
- Response Header: 200 success
- Response Body: VnfInstance[] (for multiple VNFs), VnfInstance (for single VNF)
- Design
 - If the VNFM Adapter intends to query all VNF instances, it sends a GET request to the "VNF instances" resource
 - The VNFM returns a "200 OK" response to the VNFM Adapter, and includes zero or more data structures of type "VnfInstance" in the payload body
 - If the VNFM Adapter intends to read information about a particular VNF instance, it sends a GET request to the "Individual VNF instance" resource, addressed by the appropriate VNF instance identifier in its resource URI
 - The VNFM returns a "200 OK" response to the VNFM Adapter, and includes one data structure of type "VnfInstance" in the payload body

▪ SOL003 Interfaces between SVNFM (Client) and VNFM Adapter (Provider)

• Grant VNF Request

- HTTP Method Type: POST
- VNFM Endpoint: /grants
- Request Payload: GrantRequest
- Response Header: 201 success
- Response Body: not applicable

• VNF Life-cycle Granting

- The purpose of the grant request is to have the VNFM's resource request authorized by VNFM Adapter/SO and to get advice on where to allocate resources such as virtual machines based upon capacity.

Grant requests are also useful to ensure that all resources (capacity, quota, flavors, SRTs, etc.) required for successful VNF deployment are available.

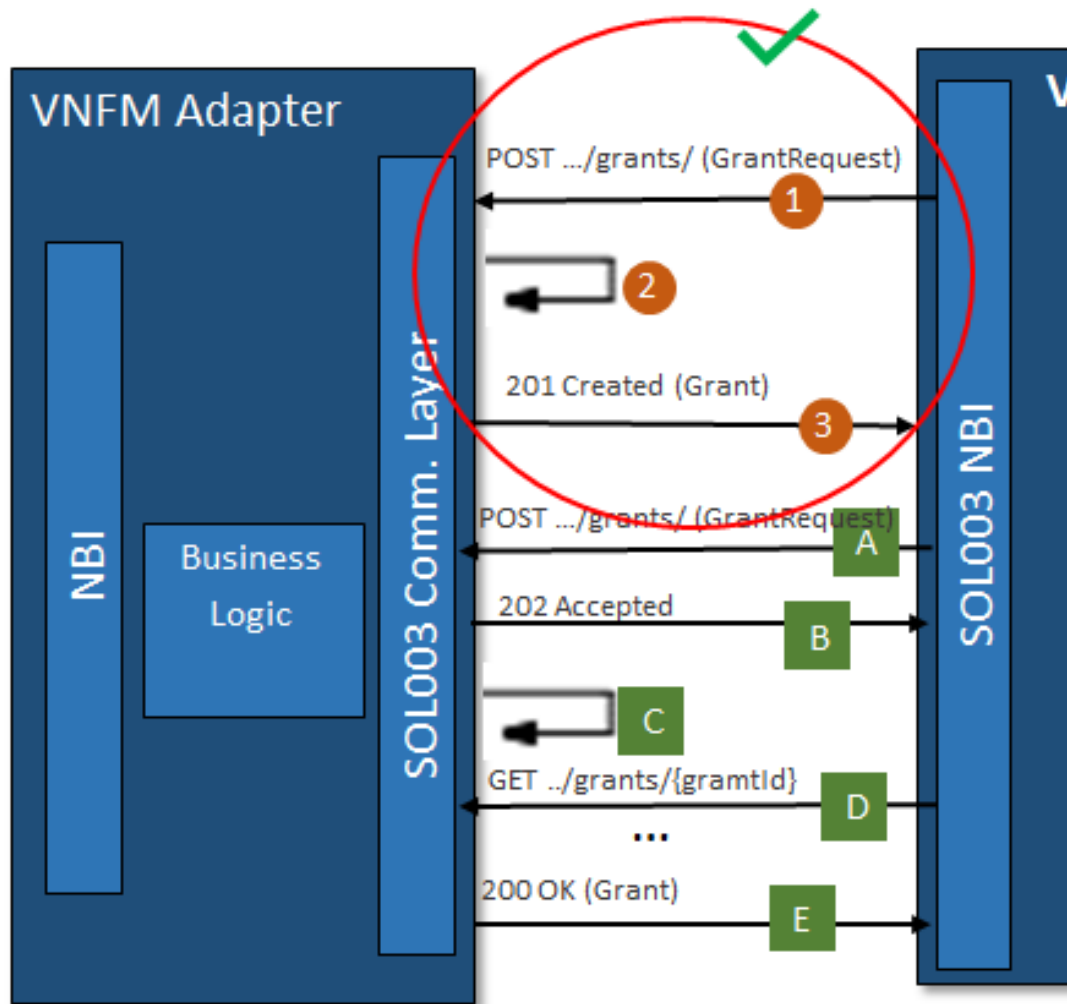
- There are two Grant Response modes, synchronous and asynchronous. **Synchronous response mode will be supported for Dublin.**

▪ Synchronous Mode

1. VNFM sends a POST request to the Grant resource with a "GrantRequest" in the body
2. VNFM Adapter with SO makes the granting decision
3. VNFM Adapter with SO returns to VNFM a "201 Created" response with a "Grant" data structure in the body

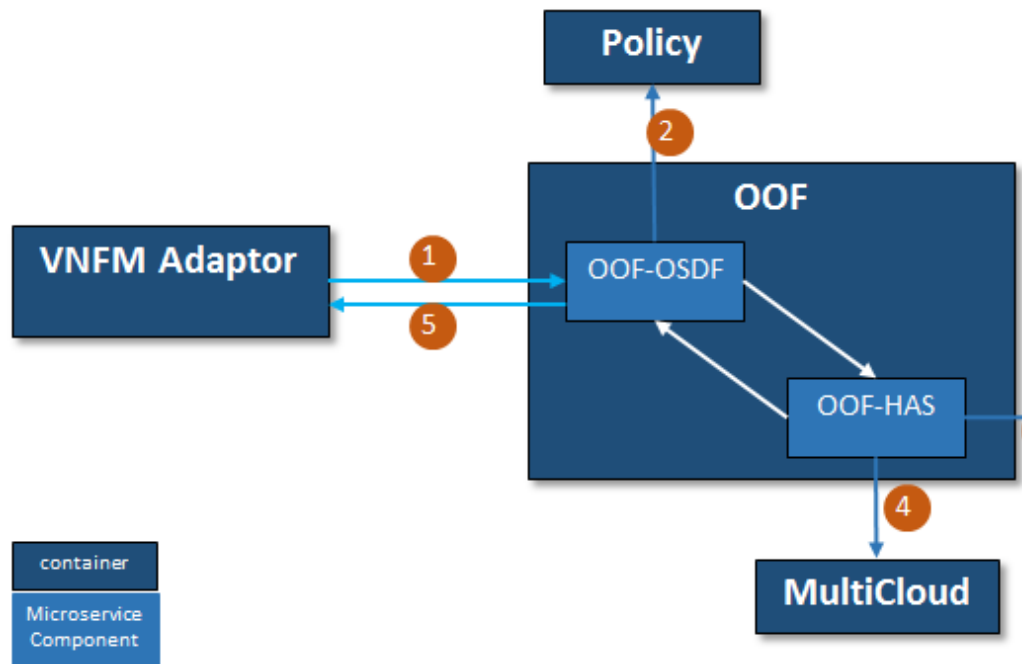
▪ Asynchronous Mode

- i. VNFM sends a POST request to the Grant resource with a "GrantRequest" in the body
- ii. VNFM Adapter with SO returns to VNFM a "202 Accepted" response with an empty body, and a "Location" header indicates a callback URL
- iii. VNFM Adapter with SO makes the granting decision
- iv. VNFM keeps trying to obtain the grant by sending a GET request to VNFM Adapter until a "200 OK" response with a "grant" data in the body
- v. VNFM Adapter finishes the granting process and returns a "200 OK" response with a "Grant" data in the body



• VNFM Adapter VNF Granting

- In Dublin, the homing data will be collected during the service decomposition procedures in SO workflows. The SO VNFM Adapter will receive the homing data from the SO workflows.
- Some models do not have homing models and policies; i.e., use of OOF is optional. In that case, the SO VNFM Adapter will make a grant decision based on VIM registration, cloud region.
- The following concept is based on the current VFC granting mechanism. *In the Dublin, the mechanism is not used.*

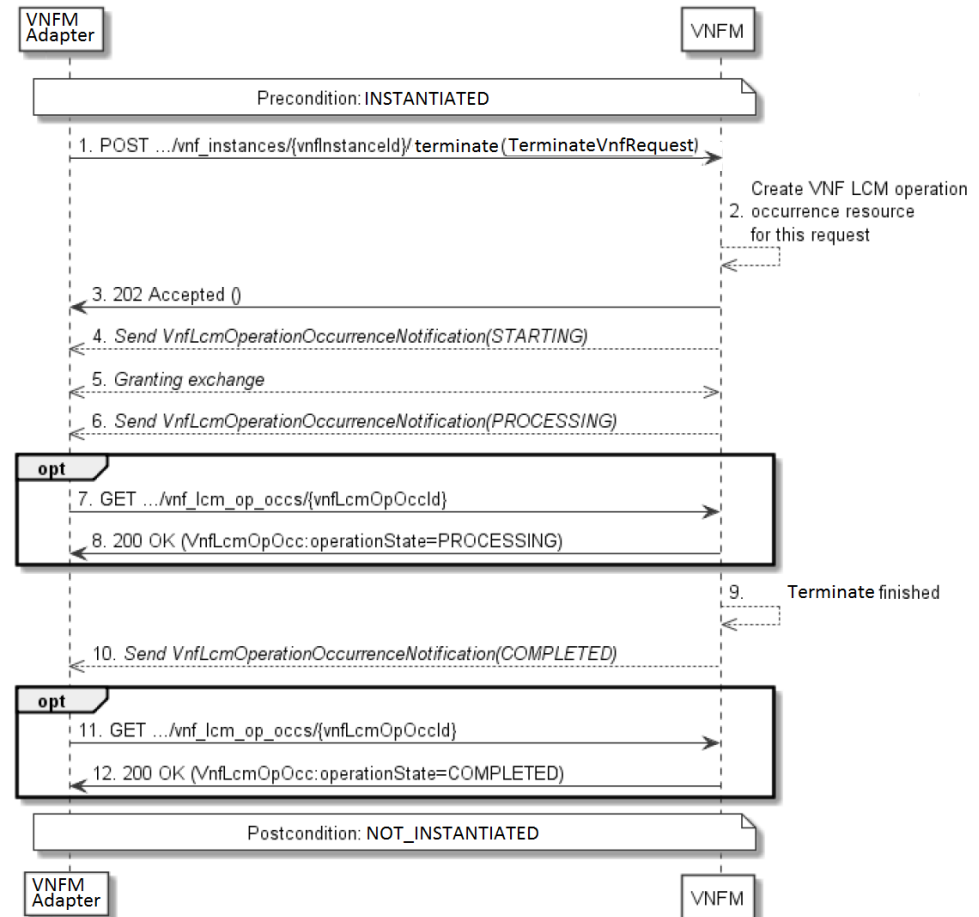


1. VNFM Adapter sends out homing requests to OOF (OSDF) containing resource info
2. OOF (OSDF) pulls all the related homing constraints from Policy
3. OOF (HAS) checks AAI database to pull region (flavor) information
4. OOF (HAS) communicates with Multi-Cloud to check cloud capacity (vims which fulfill the requirements)
5. OOF (OSDF) returns homing allocation solution to VNFM Adapter
 - OOF collects information as following:
 - Service and Resource Info, from: AAI
 - HPA Flavors/Capabilities/Capacity Info, from: AAI
 - Policy Models (Homing, PCI) from: Policy
 - Infrastructure Metrics Info (capacity), from: MultiCloud
 - Cloud Agnostic Intent Info, from: MultiCloud
 - PCI configuration data (not yet a part of SDC model)

▪ Terminate VNF

- HTTP Method Type: POST
- VNFM Endpoint: /vnf_instances/{vnfInstanceId}/terminate
- Request Payload: TerminateVnfRequest
- Response Header: 202 success
- Response Body: not applicable
- Design
 - VNF precondition = INSTANTIATED state
 - After the operation, VNF state = NOT_INSTANTIATED
 - VNFM Adapter sends a POST request to the Task resource that represents the lifecycle operation to be executed on the VNF instance, and includes in the payload body a data structure of type TerminateVnfRequest (VNFM Adapter VNFM); POST ../vnf_instances/{vnfInstanceId}/terminate
 - VNFM Creates a "VNF LCM operation occurrence" resource for the request (VNFM VNFM Adapter)
 - VNFM returns a "202 Accepted" response with an empty payload body and a "Location" HTTP header that points to the new "VNF LCM operation occurrence" resource (VNFM VNFM Adapter); ../vnf_lcm_op_occs/{vnfLcmOpOccId}
 - VNFM sends to the VNFM Adapter a VNF lifecycle management operation occurrence notification to indicate the start of the lifecycle management operation occurrence with the "STARTING" state
 - VNFM and VNFM Adapter exchange granting information (see Granting section)
 - VNFM sends to the VNFM Adapter a VNF lifecycle management operation occurrence notification to indicate that the VNF LCM operation occurrence enters the "PROCESSING" state
 - VNFM Adapter polls the "VNF LCM operation occurrence" resource to obtain information about the ongoing operation by sending a GET request to the resource that represents the VNF LCM operation occurrence.
 - VNFM returns to the VNFM Adapter information of the operation, such as the operation status, by providing in the payload body a data structure of type "VnfLcmOpOcc"
 - VNFM has finished the operation <<Operation>>
 - VNFM sends a VNF lifecycle management operation occurrence notification to VNFM Adapter to indicate the completion of the lifecycle management operation occurrence with the success state "COMPLETED"
 - Note: its communication exchange pattern is the same as the Instantiate VNF.
 - Parameters and Data source

Parameter	Required?	Data Source	Note
terminationType	Yes	from user input	
gracefulTerminationTimeout	Optional	from user input	This attribute is only applicable in case of graceful termination. The unit is seconds
additionalParams	Option	from VNFD	

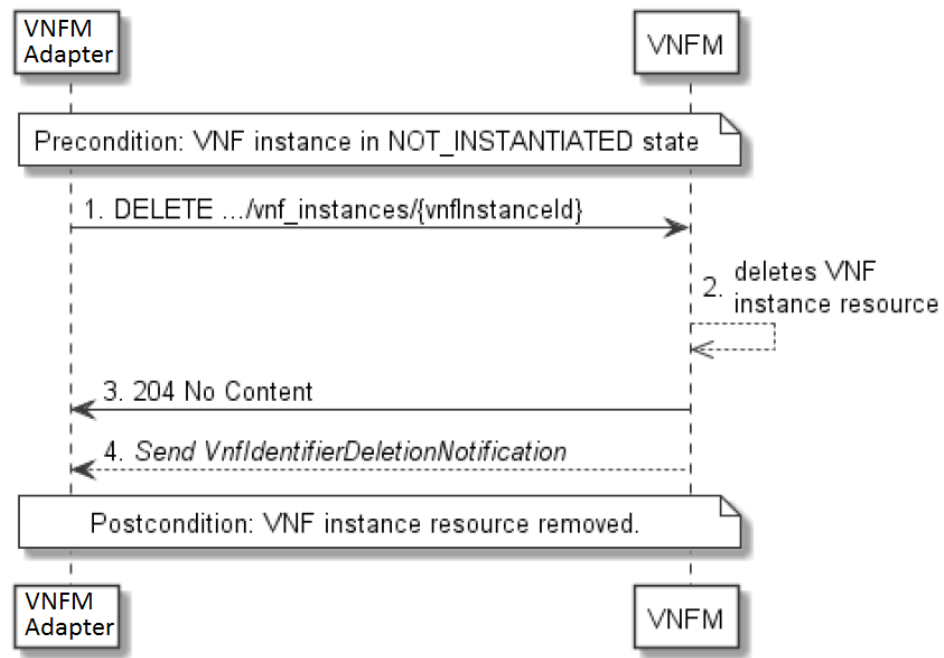


▪ Delete VNF (Stretch Goal in Dublin)

- HTTP Method Type: DELETE
- VNFM Endpoint: /vnf_instances/{vnfInstanceId}
- Request Payload: not applicable
- Response Header: 204 success
- Response Body: not applicable

• Design

- VNF precondition = NOT_INSTANTIATED
- After the operation, VNF resource has been removed from the list of VNF instance resources
- VNFM Adapter sends a DELETE request to the "Individual VNF Instance" resource.
- The VNFM deletes the VNF instance resource and the associated VNF instance identifier.
- The VNFM returns a "204 No Content" response with an empty payload body.
- The VNFM sends to the VNFM Adapter a VnfIdentifierDeletionNotification to indicate the deletion of the VNF instance resource and the associated VNF instance identifier.
- If the VNF instance is not in NOT_INSTANTIATED state, the VNFM rejects the deletion request.



▪ More SVFM SOL003 Interfaces for Future Release

- Scale VNF
 - HTTP Method Type: POST
 - VNFM Endpoint: /vnf_instances/{vnfInstanceId}/scale
 - Request Payload: ScaleVnfRequest
 - Response Header: 202 accepted
 - Response Body: not applicable
- Scale to Level Vnf
 - HTTP Method Type: POST
 - VNFM Endpoint: /vnf_instances/{vnfInstanceId}/scale_to_level
 - Request Payload: ScaleVnfToLevelRequest
 - Response Header: 202 accepted
 - Response Body: not applicable
- Delete VNF
 - HTTP Method Type: DELETE
 - VNFM Endpoint: /vnf_instances/{vnfInstanceId}
 - Request Payload: not applicable
 - Response Header: 204 success
 - Response Body: not applicable
- Operate VNF
 - HTTP Method Type: POST
 - VNFM Endpoint: /vnf_instances/{vnfInstanceId}/operate
 - Request Payload: OperateVnfRequest
 - Response Header: 202 success
 - Response Body: not applicable

◦ Impacted ONAP components

- SO
 - SO Catalog DB for SOL001/SOL004 support
 - BPMN Workflows and Recipes
 - VNFM Adapter
- SDC
 - Support SOL001/SOL004
 - Leave the current TOSCA SDC AID DM mapping as is
 - store the original vendor-provided VNF package
 - VF Module deduction based on SOL001 (out of scope from Dublin)
- SDNC
 - VNF-level Network Assignment, instead of VF-Module
- A&AI
 - VNF-level Inventory Update
 - VNFM location
- Policy (Stretch goal) - out of scope from Dublin
 - Scale-Out support for ETSI-based scaling

- Modeling
 - Support SOL001/SOL004
 - ETSI vs. ONAP-compliant VNFD (out of scope from Dublin)
- VID
 - External Virtual Link Configuration UI to map External CPs and Virtual Links such as IP Address (out of scope from Dublin)

○ Open Items

- VNF Application Configuration thru VNFM Adapter and VNFM is under discussion (out of scope from Dublin)
 - Architecture subcommittee is defining orchestration scenarios for application configuration
- Better mechanism for VNFM Adapter to locate VNFs (two methods are suggested)
 - How do we identify which VNFM to use?
- Modelling for VNF and VF Modules; mapping between VF Modules and VNF (out of scope from Dublin)
 - Assigning Network resources to SDNC; do we use preload data?
- Continue to support existing non-ETSI SO workflows along with ETSI-based workflows
- SOL001/SOL004 SO Representation (for Dublin, the second option was chosen)
 - Option #1: Enhance SO Catalog Database?
 - VNF_Package (vnfdid, vnfm_info, version, vnf_provider, product, vendor, etc.)
 - VNF_Package_Artifact (child database for VNF_Package, VNFD_URL, SoftwareImage_URL, Additional Files etc)
 - **Option #2: Store minimum reference in TOSCA_CSAR database table? This was chosen for Dublin**
- MultiCloud use (not for Dublin)
- SO VNFM Adapter High Availability and error handling are not fully covered.