verizon

Application Configuration Proposal

Fernando (Fred) Oliveira

Orchestration Scenarios relevant for this proposal

Confluence Page: https://wiki.onap.org/display/DW/Orchestration+Scenarios

- ONAP with Vendor VNFM, EMS, SA and Inventory #1a (VNF deployment)
- ONAP with Vendor VNFM, EMS, SA and Inventory #1b (VNF Scale Out)
- ONAP with Vendor VNFM, EMS, SA and Inventory #1c (VNF Scale In)
- **ONAP with Vendor VNFM, EMS, SA and Inventory #1d (VNF deployment with Application Configuration via SOL003 ModifyVnfInfo interface)**
- **ONAP with Vendor VNFM, EMS, SA and Inventory #1e (VNF deployment with Application configuration via SOL003 ModifyVnfInfo interface and SOL002 ConfigureVNF interface)**
- **ONAP with Vendor VNFM, EMS, SA and Inventory #1f (VNF deployment with Application configuration via ONAP Netconf/Restconf)**
- ONAP with Legacy Orchestration, EMS, SA and Inventory #2 (NS deployment)
- **ONAP with Legacy Orchestration, EMS, SA and Inventory #3a&b (Service deployment with VNF with ONAP App config and NS deployment)**

Orchestration Scenarios derived requirements

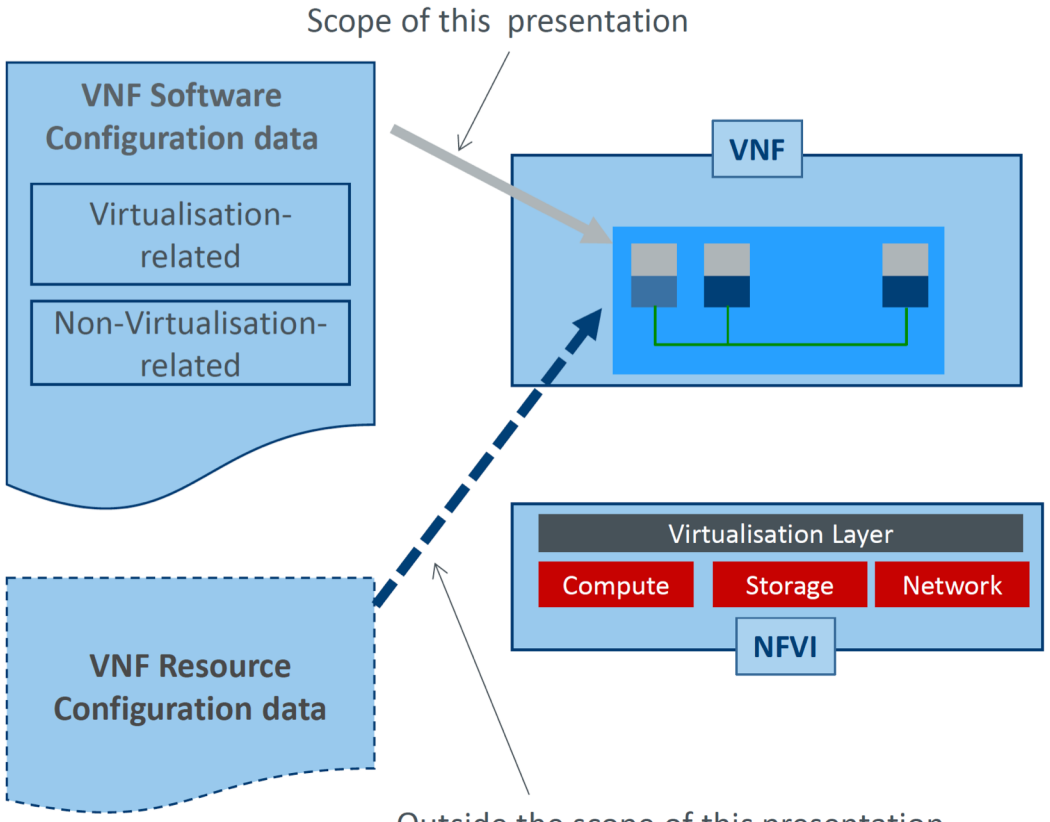
- VNF Application Parameter definition
  - ONAP needs a way for a VNF package to supply the definition of the application parameters
    - Approach A; ONAP needs to ingest and interpret a SOL001 compliant VNF Descriptor that includes the Configurable Properties (VNFSDK, SDC)
    - Approach B: ONAP needs to ingest and interpret an artifact in a SOL004 package that includes the Configurable Properties (VNFSDK, SDC)
  - ONAP needs to present these Configurable Properties in the design of an ONAP Service (SDC)
- VNF Application Configuration over SOL003 ModifyVnfInfo interface using a vendor specific VNFM (1d)
  - Upon Service deployment, ONAP needs to pass the Application Configurable Properties with the supplied values to the external VNFM that was used to deploy the VNF using the SOL003 ModifyVnfInfo interface (SO)
  - As part of a Change Management operation, ONAP needs to pass the updated Configurable Properties and new values to the external VNFM that was used to deploy the VNF using the SOL003 ModifyVnfInfo interface (SO)
  - As part of an ONAP recovery operation, ONAP needs to query the VNF to get the current state and values of the Configurable Properties using the external VNFM that was used to deploy the VNF using the SOL003 Query interface (SO)

Proposal for Application Configuration using ETSI mechanisms; Add SOL003 adapter as an APP-C/GNF-C plugin.

- Leverage SOL001 VNF-D ConfigurableProperties to describe all of the Application Configuration that the VNF supports
- Use SOL003 ModifyVnfInfo interface to program the application configuration
  - Initial Deployment related data (DNS, EMS, NTP, …)
  - Initial Application Configuration (Cluster partner(s), profile, users, ..)
  - Ongoing configuration updates (new profile, new users, …)
- Agnostic to VNFM→ VNF configuration mechanism
  - Restconf/Netconf
  - SOL002 VNF Configuration interface
  - Proprietary

ETSI defined configuration data (SOL001 ConfigurableProperties)

Classification of VNF configuration data



Virtualisation-related configuration parameters

- Parameters whose value is or can be influenced by processing functions in the NFVI and/or NFV-MANO
- e.g. IP address of a Connection Point of a VNFC to be configured on another VNFC

Non-virtualisation-related configuration parameters

- Parameters whose value cannot be influenced by processing functions in the NFVI and/or MANO
- e.g. APN-to-GGSN mappings in a SGSN VNF

https://docbox.etsi.org/ISG/NFV/Open/other/Tutorials/201810-Tutorials-SDN_NFV_World_Congress-The_Haque/ETSI_NFV_Layer123_VNF%20Configuration-v2_RX15009.pdf

SOL001 VNF-D Configurable Properties Example:

```
tosca_definitions_version: tosca_simple_yaml_1_2

node_types:
  MyCompany.SunshineDB.1_0.1_0:
    derived_from: tosca.nodes.nfv.VNF
    properties:
      flavour_id:
        constraints:
          - valid_values: [ simple, complex ]
      configurable_properties:
        type: MyCompany.datatypes.nfv.VnfConfigurableProperties


data_types:
  Mycompany.datatypes.nfv.VnfConfigurableProperties:
    derived_from: tosca.datatypes.nfv.VnfConfigurableProperties
    properties:
      additional_configurable_properties:
        type: MyCompany.datatypes.nfv.VnfAdditionalConfigurableProperties

  MyCompany.datatypes.nfv.VnfAdditionalConfigurableProperties:
    derived_from: tosca.datatypes.nfv.VnfAdditionalConfigurableProperties
    properties:
      name_prefix_in_vim:
        type: string
        required: false
      dns_server:
        type: string
        required: true
```

ETSI defined configuration data (SOL001 ConfigurableProperties)

CONFIGURATION PATH B: (OSS -> NFVO or EM ->) VNFM -> VNF software instance

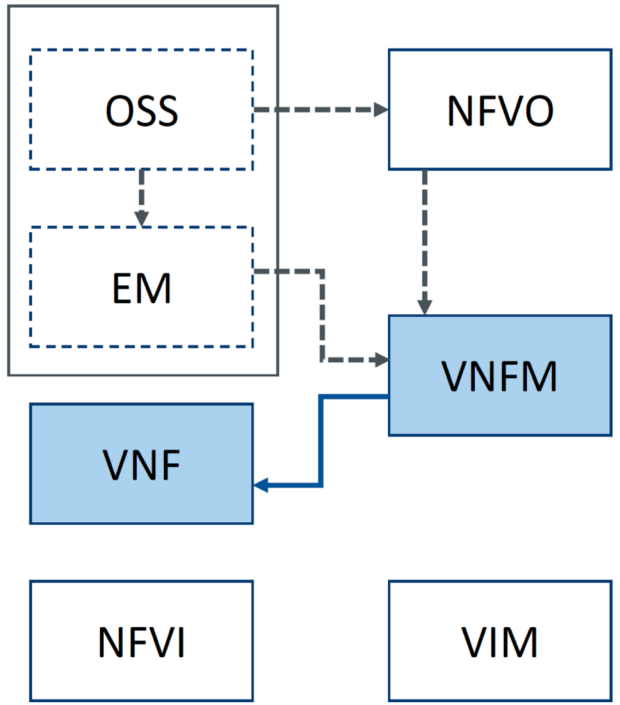
Relies on the support of the optional VNF Configuration interface (push mode) or on VNF LCM notifications followed by a Query VNF operation (Pull Mode) defined in ETSI GS NFV-IFA 008.

Non-virtualisation-related configuration data

- Relies on the configurable properties declared in the VNFD
- NFVO and VNFM are used as a "tunnelling" mechanism between the OSS and the VNF Application

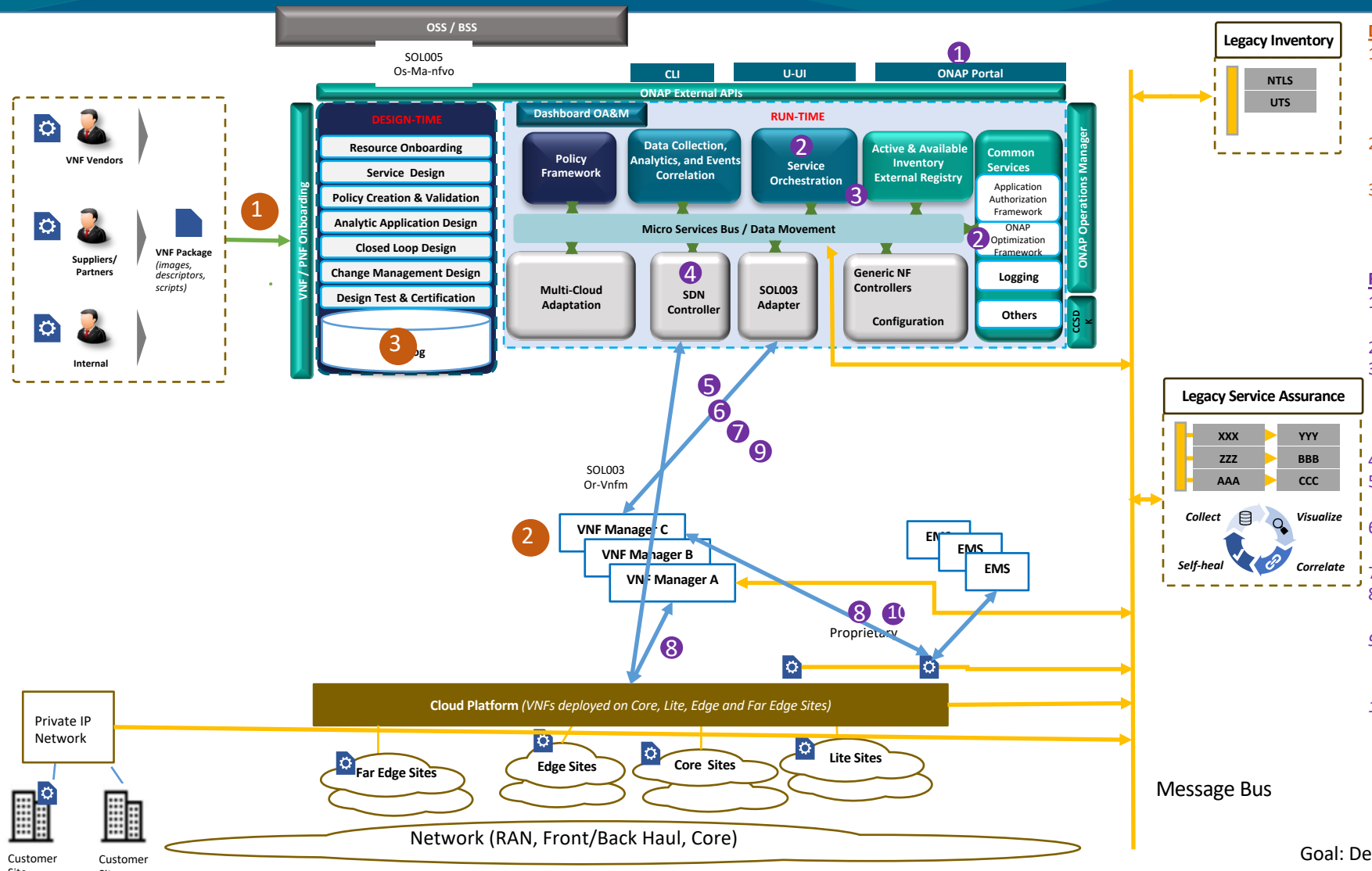
Virtualisation-related configuration data

- Relies on the configurable properties declared in the VNFD and/or on pre-defined parameters available in VnfInfo (DHCP server address to use, Addresses and ports assigned to the Connection Points)



VnfInfo represents the runtime view of a VNF instance exposed by a VNFM.

ONAP with Vendor VNFM, EMS, SA and Inventory #1d (VNF deployment with Application configuration via SOL003 ModifyVnfInfo interface)



Design/Develop Time:

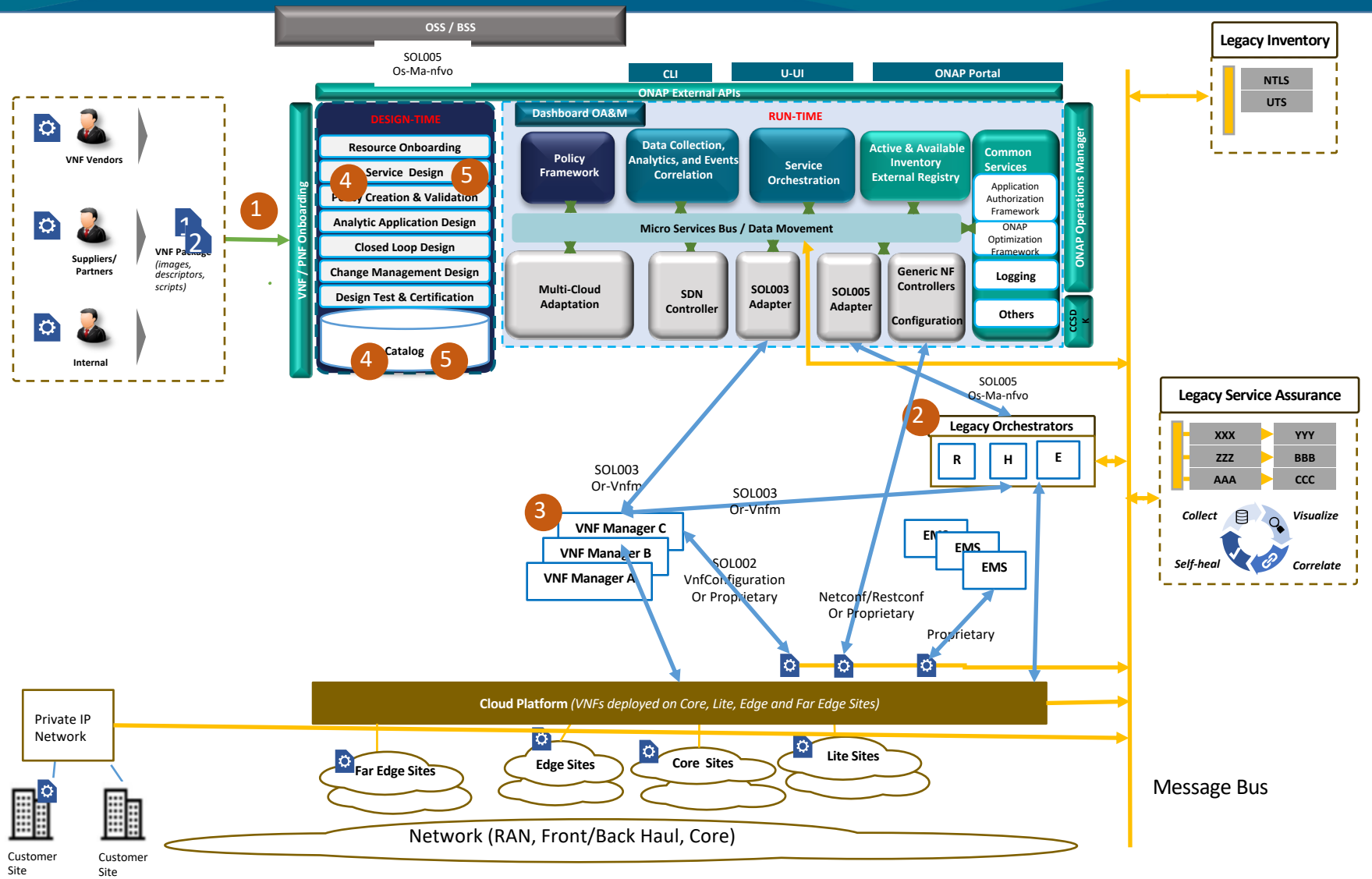
1. Onboard and catalog a SOL004 VNF package with a SOL001 VNFD.
   a) *Application level ConfigurableProperties attributes included in the VNF-D from the vendor*
   b) *Could also be provide in a separate artifact in the SOL004 package*
2. VNFM C registers as a SOL003 compliant VNFM
   a) *Support "ModifyVnfInfo" operation*
3. Design an ONAP Service A with deployment *and application* configuration referencing the onboarded VNF and VNFM type.
   a) *ConfigurableProperties extracted from the VNF-D(s) or alternate artifact and composed into Service level configuration*

Run Time:

1. ONAP receives request to create an instance of Service A *with appropriate configuration data*.
2. ONAP "decomposes" request into VNF 1 and homes it.
3. ONAP makes resource assignments for deployment of VNF 1 based on information in the VNF-D
   a) #vCPU, RAM, Network, Storage, CPU pinning, SR-IOV, EPA
   b) Issue on if/how resources are "reserved" by OOF/A&AI
4. ONAP makes assignments (IP Address) for the VNF
5. ONAP selects VNFM C (match type and homing) and calls via SOL003 API, requesting creation of VNF 1, passing "deployment data" values.
6. VNFM C upcalls ONAP asking for a "grant" of resources based on information in the VNF-D.
7. ONAP responds with resources allocated in #3 along with VIM credentials.
8. VNFM C calls VIM to create VNF 1, and applies any needed deployment data via proprietary API.
9. *ONAP decomposes service config data for this VNF and updates the application configuration data over SOL003 ModifyVnfInfo interface with VNFM C*
10. *VNFM C updates the VNF via a proprietary mechanism*

Goal: Design and deploy a composite Service consisting of multiple vendor VNFs in an automated and repeatable way leveraging vendor provided SOL003 compliant VNF Manager(s) *and ONAP Application configuration*.

ONAP with Legacy Orchestration, EMS, SA and Inventory #3a (Service deployment with VNF (with ONAP App config) and NS deployment) (Design Time)

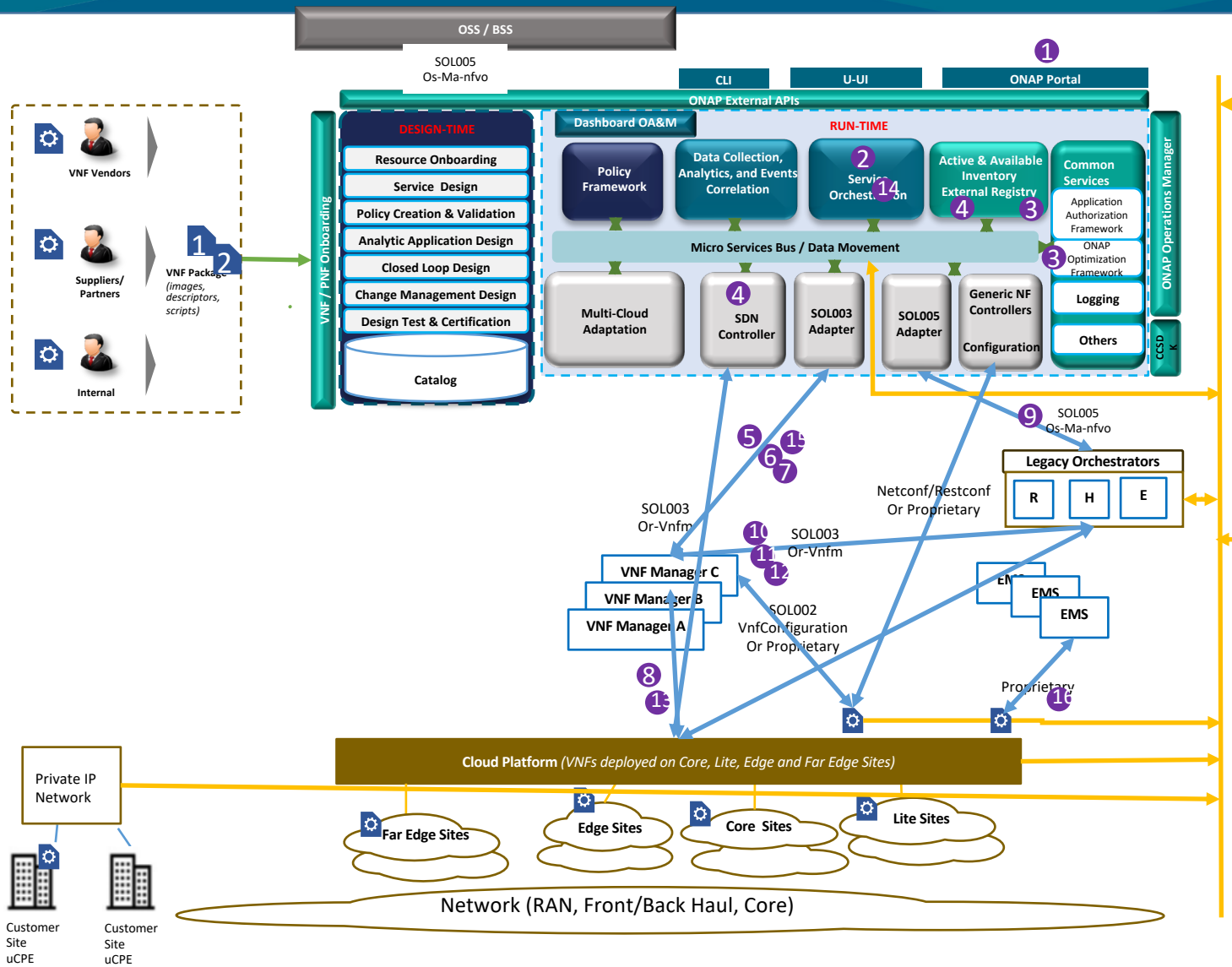


Design/Develop Time:

1. Onboard SOL004 VNF packages with a SOL001 VNFD.
   a) *Application level ConfigurableProperties included in the VNFD from the vendor*
   b) *Could also be provide in a separate artifact in the SOL004 package*
2. NFVO R registers as a SOL005 compliant NFVO
3. VNFM C registers as a SOL003 compliant VNFM
   a) *Supporting "ModifyVnfInfo" operation*
4. Design an ONAP Service A referencing VNF 1 and and NFVO R.
   a) *ConfigurableProperties extracted from the VNFD or alternate artifact and composed into Service level configuration*
5. Design an ONAP Service B with deployment *and application* configuration referencing VNF 2 and Service A.
   a) *ConfigurableProperties extracted from the VNFD or alternate artifact along with Service A configuration properties and composed into Service B configuration*

Goal: Leverage ONAP VNF onboarding and Service Design capabilities to develop a hierarchical composite service parts of which can be directly instantiated by a VNFM along with a subservice that can be instantiated and managed by an existing SOL005 compliant orchestrator.

ONAP with Legacy Orchestration, EMS, SA and Inventory #3b (Service deployment with VNF with ONAP App config and NS deployment) (Run Time)



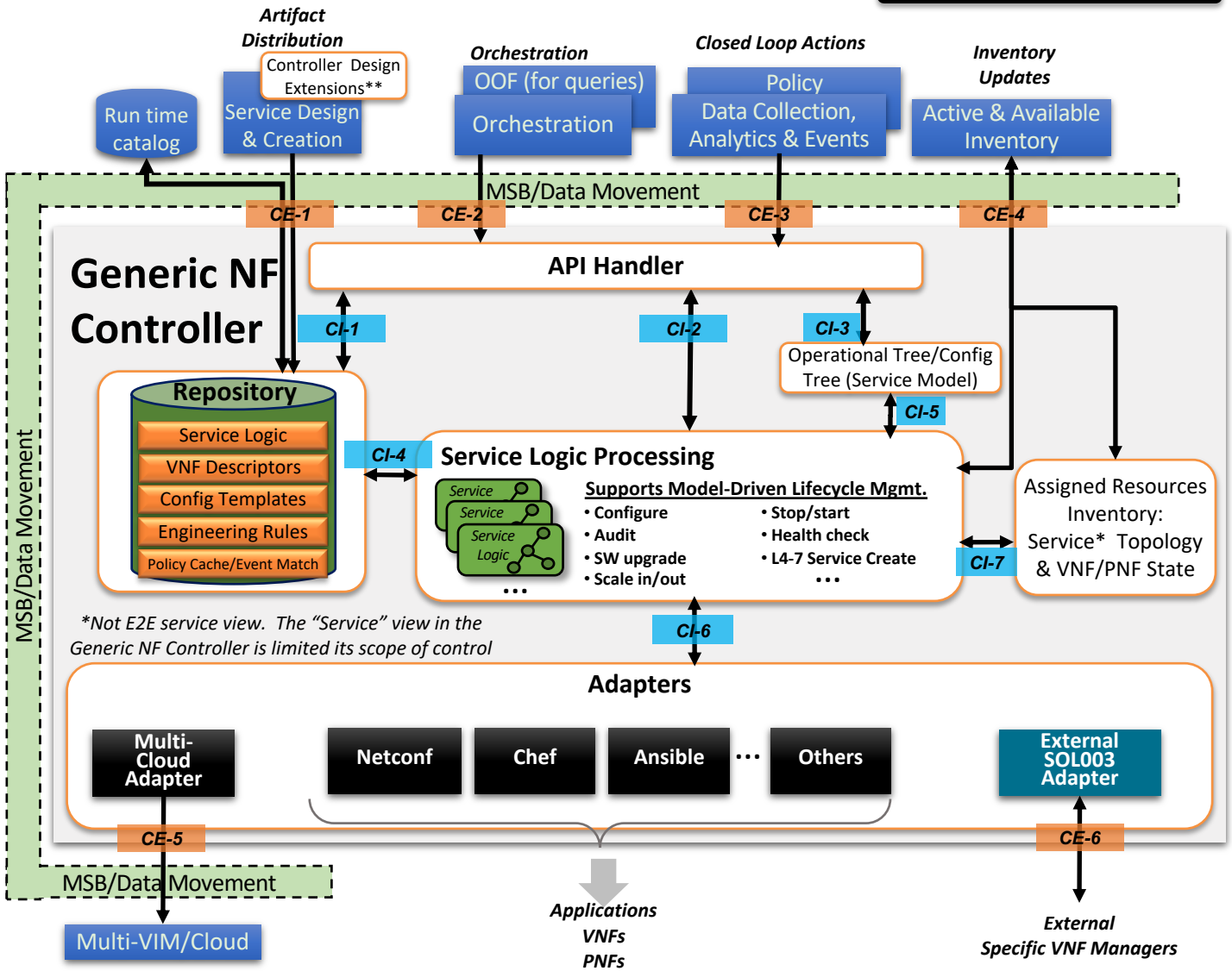
Run Time:

1. ONAP receives request to create an instance of Service B.
2. ONAP "decomposes" the request into an NS and a VNF.
3. ONAP makes resource assignments for deployment of VNF 1 based on information in the VNF-D
   a) #vCPU, RAM, Network, Storage, CPU pinning, SR-IOV, EPA
4. ONAP makes assignments (IP Address) for the VNF and the NS
5. ONAP selects VNFM C (match type and homing) and calls via SOL003 API, requesting creation of VNF 1, passes "deployment data" values.
6. VNFM C upcalls ONAP asking for a "grant" of resources based on information in the VNF-D.
7. ONAP responds with resources allocated in #3 along with VIM creds.
8. VNFM C calls VIM to create VNF 1, and applies any needed deployment data via proprietary API.
9. ONAP calls NFVO R via SOL005 API, requesting creation of NS, passing configuration values.
10. NFVO R calls VNFM C requesting an instance of VNF 2.
11. VNFM C upcalls NFVO R asking for a "grant" of resources.
12. NFVO R responds with resources and VIM creds.
13. VNFM C calls VIM to create VNF 2, and applies any needed deployment data via proprietary API.
14. ONAP decomposes service config data for VNF 1
15. ONAP updates VNF 1 application configuration data over SOL003 ModifyVnfInfo interface with VNFM C
16. NFVO R updates VNF 2 application configuration data over SOL003 ModifyVnfInfo interface with VNFM C.

Goal: Leverage ONAP VNF onboarding and Service Design capabilities to develop a hierarchical composite service, parts of which can be directly instantiated by a VNFM along with a subservice that can be instantiated and managed by an existing SOL005 compliant orchestrator.

Generic NF Controller Architecture

- **Generic NF Controller configures and maintains the health of VNFs/PNFs/services* (L0-7) throughout their lifecycle.**
  - **The Lifecycle Management and Configuration Functions are a normalization of the controller functions into a common, extensible library**
- **Programmable network application management platform**
  - **Behavior patterns programmed via models and policies**
  - **Standards based models & protocols for multi-vendor implementation**
  - **Extensible SB adapter set including vendor specific VNF-Managers**
  - **Operational control, version management, software updates, etc.**
- **Manages the health of VNFs/PNFs within its scope**
  - **Policy-based optimization to meet SLAs**
  - **Event-based control loop automation to solve local issues near real-time**
- **Local source of truth**
  - **Manages inventory within its scope**
  - **All stages/states of lifecycle**
  - **Configuration audits**
- **Key Attributes of Generic NF Controllers**
  - **Intimate with network protocols**
  - **Manages the state of services**
  - **Provide Deployment Flexibility to meet user scalability / resilience needs**

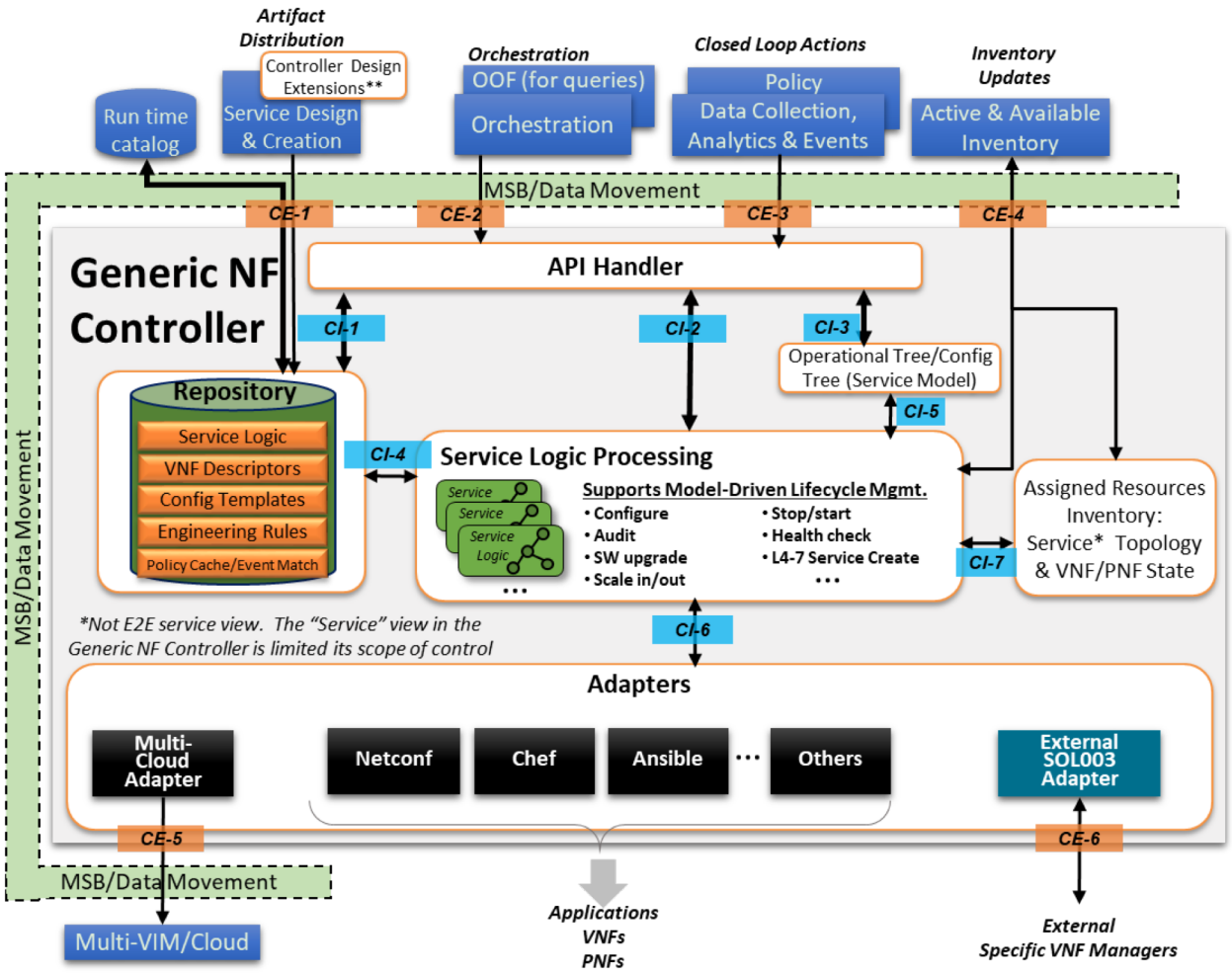


*How the services are to be handled is for further study

** Controller Design plugin extensions to be integrated into SDC Design Studio and Workflow

***CE-6 not needed - see External Controller materials

Generic NF Controller – External/Internal Interface Definitions



	Interface Definitions
CE-1	Distribution of artifacts from Service Design and Creation – artifacts distributed to Run Time Catalog, GNFC receives notification and pulls from Run Time Catalog *Note: Controller Design Plugin Extensions to be integrated into Service Design & Creation*
CE-2	Service requests from Orchestration ONAP Optimization Framework (OOF) queries for VNF state and available capacity
CE-3	Closed Loop action requests from Data Collection, Analytics & Events/Policy
CE-4	Inventory retrieval from Active & Available Inventory by Service Logic Processing engine Inventory updates to Active & Available Inventory by Assigned Resources Inv
CE-5	Lifecycle management requests to Multi-Cloud (e.g., stop/start VM)
CE-6	**LCM requests to an external VNF Manager that has responsibility of the target VNF**
CI-1	API Handler looks up or retrieves the corresponding Service Logic instance that maps to NB service request (service/network yang)
CI-2	API Handler calls Service Control Processing to perform the Service Logic on the target service or network
CI-3	Prior to CI-2, API Handler might query the (in-memory) Operational/Config Trees for the network or service details (if already existing)
CI-4	Service Control Processing retrieves the Service Logic, Config Templates, Engineering rules, and Policies as part of processing the requested action
CI-5	Service Control Processing queries and/or updates Operational/Config Trees as part of making changes to the network (VNFs/PNFs)
CI-6	Service Control Processing requests adapter layer to update/configure VNF/PNF update using the appropriate adapter for the VNF/PNF
CI-7	Service Control Processing queries and/or updates local Assigned Resources Store/Inventory as part of making changes to the network (VNFs/PNFs)

GNFC – External Interface Details

	Interface Definitions	Beijing Rel.	Casablanca Rel.	Protocol /Service	Comments
CE-1	Distribution of artifacts from Service Design and Creation	SDC➔[no GNFC]	SDC ➔ GNFC (trigger) GNFC ➔ Run Time Catalog (pull)	DMaaP	
CE-2	Service requests from Orchestration Queries from ONAP Optimization Framework (OOF) for VNF state and available capacity	SO, Portal ➔[no GNFC] OOF ➔ [no GNFC]	SO, Portal ➔ GNFC OOF queries – not in scope?	REST	Generic Request API. See next slide for orchestration requests for LCM actions.
CE-3	Closed Loop action requests from Data Collection, Analytics & Events & Policy	DCAE ➔ [no GNFC] Policy – not in scope	DCAE ➔ GNFC Policy – not in scope	DMaaP	
CE-4	Inventory retrieval from Active & Available Inventory by Service Logic Processing engine Inventory updates to Active & Available Inventory by Assigned Resources Inventory	A&AI ⇔ [no GNFC]	A&AI ⇔ GNFC	REST	
CE-5	Configuration requests for cloud infrastructure networking Lifecycle management requests to Multi-Cloud (e.g., stop/start VM)	Multi-Cloud – not in scope	GNFC ➔ M-Cloud	REST	
CE-6	**Lifecycle management requests to SOL003 VNFM (e.g., create/instantiate/terminate/scale/heal VNF)** **Lifecycle Notification events from SOL003 VNFM**		**GNFC ➔ SOL003 VNFM** **SOL003 VNFM ➔ GNFC**	**REST**	**E release??**

- Controllers are to be Model-Driven – APIs in Dev, Design, Run-Time catalogs
- Payloads: parameter values defined in the platform Data Dictionary (model/meta-data driven)
- Beijing Release does not have an implementation of GNFC
- For Casablanca it is recommended that APPC and SDNC begin to transition toward GNFC (for new use cases such as 5G)

Next Steps

- Further discussion in the Orchestration Scenarios Task force
  - Agree on standard approach for a VNF to provide it's application parameters
  - Agree on mechanism(s) to pass values of the application parameters to VNFs
- Design and develop a PoC of using a SOL003 Plugin to APP-C/GNF-C so that configuration and other LCM can follow the current path