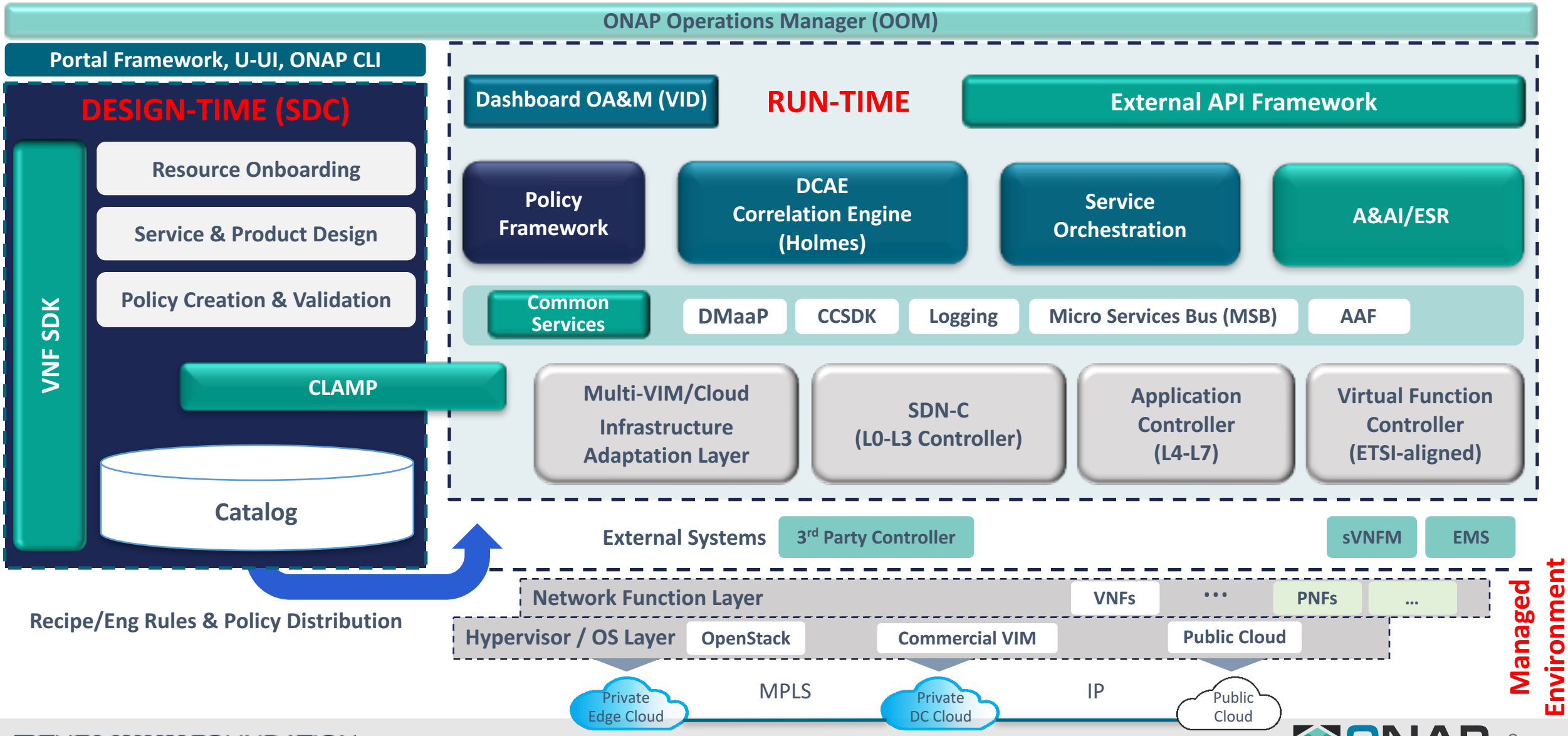
ONAP Beijing Architecture (v2.0.3)
For TSC Approval

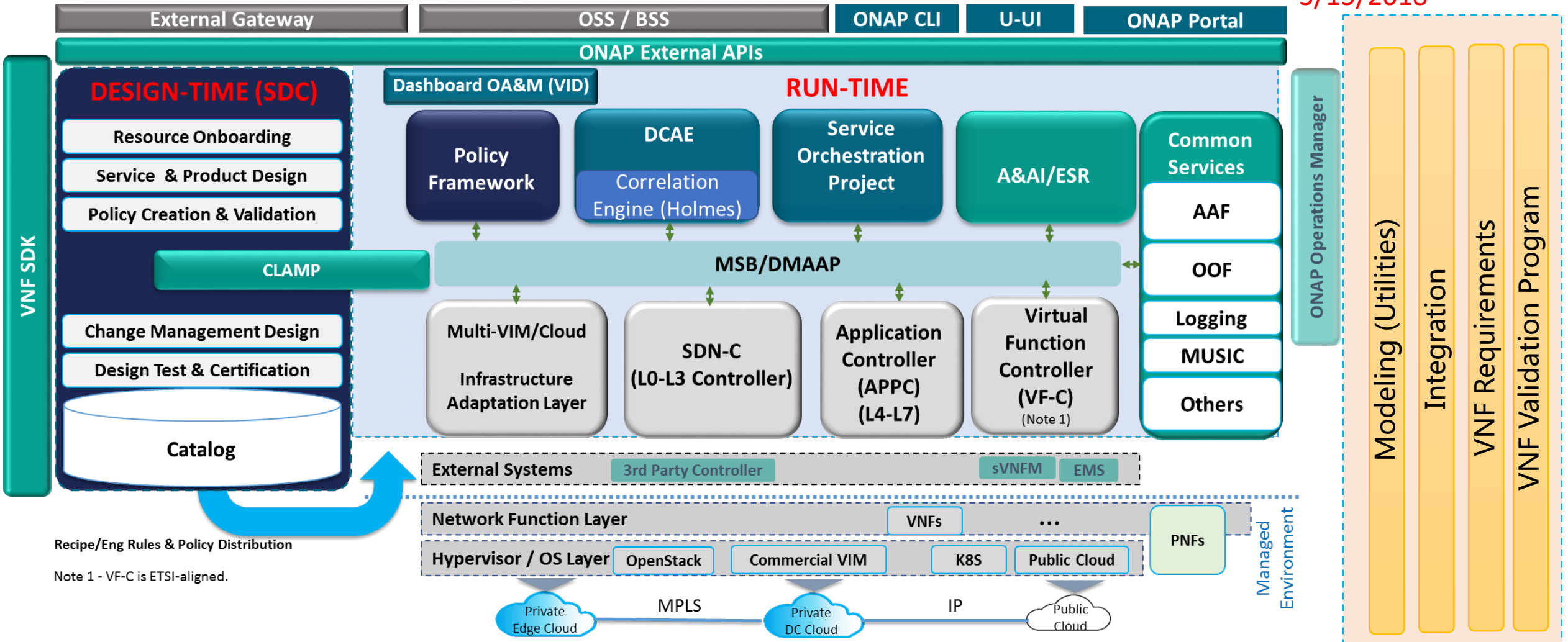
Chris Donley

ONAP Amsterdam Architecture (for reference)



ONAP Beijing Architecture
(High-Level View with Projects)

Version 2.0.3
3/15/2018



Changes in Beijing Architecture

- Since the Beijing release is focusing on Platform Maturity, the ARC has minimized changes to the functional architecture
  - New projects approved by the TSC
  - Formatting updates to provide clarity

- As a foundation for the long-term target architecture, ONAP will strive for model-driven, microservices-based, modular architecture, and enable integration with other systems via MSB/proxy
  - Enable secure communication between internal and external components
  - Improve APIs for better consistency between components
  - Align with models from Modeling subcommittee and external SDOs (as appropriate)
  - Use OOM for instantiation and register with Microservices Bus

ONAP Beijing Architecture Recommendations

Recommendations for Beijing Release

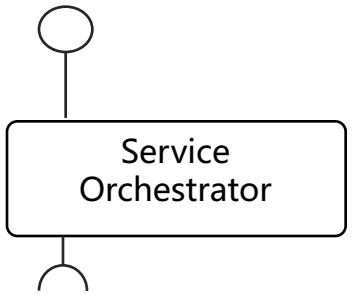
- Achievable **Recommendations** for Beijing to begin aligning with target architecture
  - Key Architecture Principles: Modular, Microservices, Model-driven
  - Modular:
    - Improve APIs between components. Document them using Swagger.
    - Align interfaces with industry standards, where available/appropriate
  - Microservices-based
    - Use OOM for onboarding, instantiation (support MSB natively), scalability
    - Register with MSB
    - Use microservices best practices (e.g., separate data from the component, use common KV stores/databases, for KV stores – prefer Consul, etc.)
  - Model-driven
    - Align with info/data models from the Modeling Subcommittee (to be published Feb 2018)
    - Explore ways to use model-driven principles inside each component as well as across various components *(SDC, SO, VFC, APPC, ..)*
- Release Theme is Platform Maturity/S3P
  - S3P (see Jason Hunt 1/4 TSC presentation for list of projects providing support for S3P)
    - Adhere to cloud-native principles (see https://martinfowler.com/, https://12factor.net/)
    - Use common libraries/SDKs/common services where possible to improve overall system resiliency/reliability
    - More focus on common services (where possible) also to help reduce testing
    - MUSIC for managing various ONAP components states, DB replication in a consistent manner
    - Enhanced AAF to support credential management for certificates
    - Support Image Management service to enhance security, virus checking, etc.

Functional Description of Beijing Project Components

Note – The detailed description of features/capabilities/interfaces for each component planned for the Beijing release will be provided by respective projects.

SO: Service Orchestrator (1/2)

- Service Management Interface
- Network Service LCM Interface
- Actuation Request Interface
- Service Definition Reception interface



Service Management Interface
VNF LCM Interface
NF Config Interface
Optimization request Interface
Inventory Service Interface
Transport Service Interface
Network Service LCM interface

Definition:

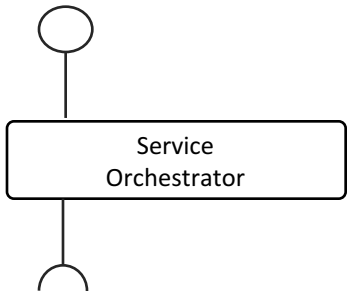
- Provides functionality for the execution of specified process and automated sequencing of activities, task, rules and policies needed for creation, modification, removal of network application, infrastructure services or resources.
- Supports different specializations with specific orchestration scopes.
  - Specialization scopes include, but are not limited to, PNF orchestration, Service and VF scaling, Homing and placement.

Provided Interfaces:

- Service Management Interface
  - Provides the interface to manage the services that orchestration provides.
- LCM interfaces
  - interfaces for service LCM.
  - interfaces for VNF LCM.
- Actuation Request Interface
  - Provides support of actuation requests towards the services.

SO: Service Orchestrator (2/2)

- Service Management Interface
- Network Service LCM Interface
- Actuation Request Interface
- Service Definition Reception Interface



Service Management Interface
VNF LCM Interface
NF Config Interface
Optimization request Interface
Inventory Service Interface
Transport Service Interface
Network Service LCM interface

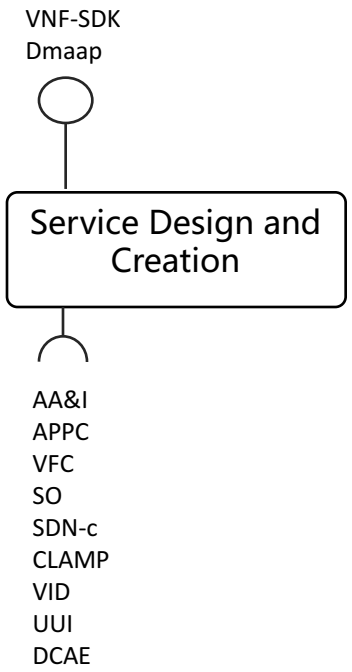
Consumed interface Interfaces:

- VNF LCM Interface, from: Application controller
- Optimization Request Interface, from: Optimization Framework
- NF Config Interface, from: Application controller
- Inventory Service Interface, from: Available and Active Inventory
- Transport Service Interface, from: SDN Controller.
- Network Service LCM interface, from: Virtual Function Controller
- Service Distribution interface, from: Service Design and Creation

Consumed Models:

- Network Service Descriptor (from SDC)

SDC: Service Design and Creation (1/2)



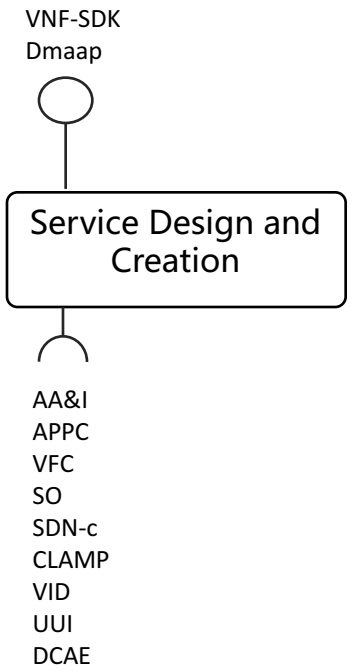
Definition:

- SDC is the Integrated development platform for modeling.
- SDC provides set of functionalities starting from the VNF onboarding and up to the service distribution.
- SDC provides a set of modeling capabilities including VNF modeling, Service modeling.
- As part of the work for Beijing SDC is adding the Monitoring template modeling capabilities and work flow modeling capabilities.

Provided Interfaces:

- External API'S
  - Retrieve artifacts.
  - Query catalog.
  - Upload artifacts.
- Distribution client
  - Java based client for subscribing and receiving notifications from Dmaap regarding Services.
  - The client provides a way to define what artifacts need to be retrieved from the service.
  - The client downloads the artifacts from SDC and publishes a notification regarding its success or failure.
- SDC parser
  - A java based parser for Tosca.
  - The parser allows components to parse the CSAR artifact provided by SDC.
- Generic Designer Frame work
  - Infrastructure to support easy integration of new designers into SDC.
  - SDC-UI a set of ui components and styles to allow designers the same look and feel as SDC.

SDC: Service Design and Creation (2/2)



Consumed interface Interfaces:

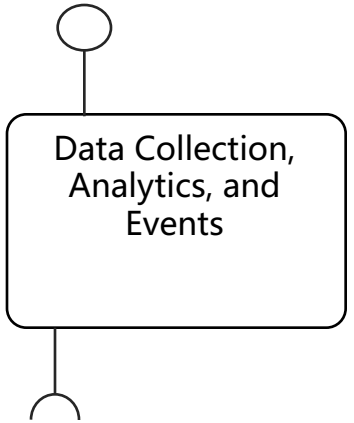
- VNF SDK – SDC can query VNF's from the marketplace and onboard them into SDC.
- Publish and subscribe to notifications using Dmaap.

Consumed Models:

- VNF packages Heat based and Tosca based

Data Collection, Analytics, and Events

- Data collection interface
- Deployment interface
- Config binding interface



- Data movement platform interface (DMaaP)
- Data enrichment interface (A&AI)
- Service model change interface (SDC)
- Policy interface (Policy)

Definition:

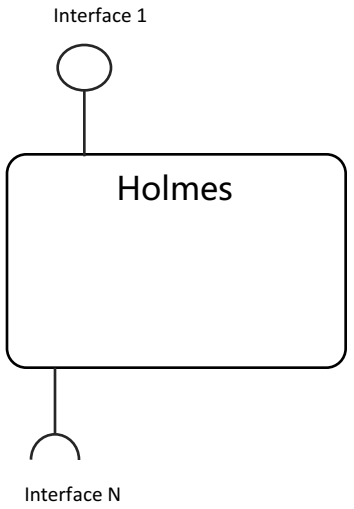
DCAE is the ONAP subsystem that supports closed loop control and higher-level correlation for business and operations activities. DCAE collects performance, usage, and configuration data; provides computation of analytics; aids in trouble-shooting and management; and publishes event, data, and analytics to the rest of the ONAP system for FCAPS functionality.

Provided Interfaces:
- Interface 1: Data collection interface (provided by DCAE collectors, consumed by VNFs and others)
  - Interface for various FCAPS data entering DCAE/ONAP.
- Interface 2: Deployment interface (provided by DCAE Deployment Handler, used by CLAMP and other northbound applications/services)
  - Interface for triggering the deployment and changes of a control loop
- Interface 3: Configuration Binding Service
  - Interface for querying the information of the services that are registered to DCAE Consul

Consumed Interfaces:
- Interface 1: Data movement platform interface (provided by DMaaP)
  - Interface for data transportation between DCAE subcomponents and between DCAE and other ONAP components
  - This interface can also be used for publishing events to other ONAP components.
- Interface 2: Data enrichment interface (provided by A&AI)
  - Interface used by DCAE collectors and analytics for querying A&AI for VNF information for the purpose of enriching collected raw data by adding information not contained in original data.
- Interface 2: Service model change interface (Provided by SDC)
  - Interface for DCAE Service Change Hander fetching control loop models and model updates.
- Interface 4: Policy interface (Provided by Policy)
  - Interface for DCAE Policy Hander fetching configuration and operation policies on control loop and control loop components from Policy.

Consumed Models: TOSCA models descripting control loop construction (e.g. collection and analytics apparatus)

Holmes



Definition:
- Maintains a series of rules inside to support different kinds of correlation analysis scenarios and control loops.
- It publishes the analysis result to the data bus to feed any components that need the data.

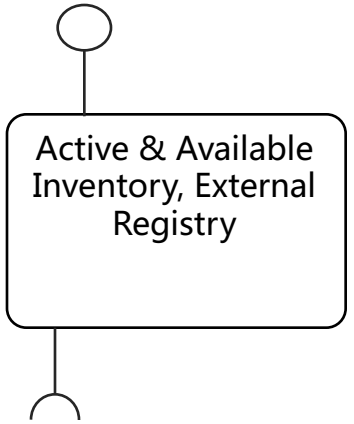
Provided Interfaces:
- Rule Management Interface
  - Provides the ability to create, updae, delete or query rules.

Consumed interface Interfaces:
- Data pub/sub interfaces, from: DMaaP
- Service info query interfaces, from: DCAE
- Service registration and locating interfaces, from: MSB
- Inventory info query interfaces, from: A&AI

Consumed Models
VES Model
Inventory Model

Active & Available Inventory, External Registry

- Inventory Service Interface
- External Register Interface



Definition:
- AAI maintains a live view of services and resources in the network, providing the state and relationships of the service components
- Maintains the view of the managed systems services and resources, as well as information of the external systems that ONAP will connect to.
- It provides real-time views of a managed systems resources, services and relationships with each other.
- AAI provides a GUI to provide users the ability to find and inspect inventory data. This includes a free-text search, inspection of specific entities and their relationships, and aggregated views of data.
- AAI also provides a GUI to manage the external system information, including the register/update/delete and query external system.

Provided Interfaces:
- Inventory Service Interfaces:
  - AAI Resources REST API - Provides the ability to store, read, update inventory information
  - Gizmo: a low-level REST CRUD API that provides targeted and simple access to entities, relationships, and their properties. Gizmo also provides lightweight collections, and transactional bulk write support
- Complex Query Interface: AAI Traversal REST API – Provides the ability to perform complex traversals of the AAI Graph
- External Register interface
  - Provides the ability to register/update/delete and read external system information

Consumed interface Interfaces:
- DMaaP
- SDC
- Multivim

Consumed Models: TOSCA Service and Resource Models

AAF Functional Description

- The purpose of AAF (Application Authorization Framework) is to organize software authorizations so that applications, tools and services can match the access needed to perform job functions. This is a critical function for Cloud environments, as Services need to be able to be installed and running in a very short time, and should not be encumbered with local configurations of Users, Permissions and Passwords.
- To be effective during a computer transaction, Security must not only be secure, but very fast. Given that each transaction must be checked and validated for Authorization and Authentication, it is critical that all elements on this path perform optimally.
- AAF contains some elements of Role Based Authorization, but includes Attribute Based Authorization elements as well.

Essential Functional Components:

- The core component to deliver this Enterprise Access is a RESTful service, with runtime instances backed by a resilient Datastore (Cassandra as of release 1.3)
- The Data is managed by RESTful API, with Admin functions supplemented by Character Based User interface and certain GUI elements.
- The Service accessible by provided Caching Clients and by specialized plugins
- CADI (A Framework for providing Enterprise Class Authentication and Authorization with minimal configuration to Containers and Standalone Services)
- Cassandra (GRID Core)

Additional Functional Components:

- Secret Management service : This service provides a way to add/delete/retrieve secret domains and secrets within each domain
- Security of private keys using PKCS11 based HSMs: Secures the private keys of CA using PKCS11 based HSMs. Also show cases the HSM integration with softHSM2.0 and optionally provide a way to secure the softHSM using TPM 2.0 hardware entity.

AAF Functional Description

- CADI stands for Code, Access, Data and Identity, This Framework addresses the Runtime Elements of Access and Identity.
- Many other tools address elements of this vision. For instance, Sonar and Fortify evaluate code for maintainability and security problems, while Voltage provides encryption for Data at Rest.

However, CADI Framework contributes to these for runtime applications by:

- Code - CADI provides reusable Security Client code, and ties in with appropriate Security Interfaces (i.e. J2EE Standard Filters)
- Access - CADI provides links to Authorization tool(s) (AAF) for Fine-grained Authorization. Also, data from Identity servers is also made available to Coders easily. For instance, CADI provides a clean API to read the Course Grained Authorization information available within the CSP Cookie.
- Data - CADI provides simple setup for certificates needed for TLS connectivity, so that barriers to using HTTPS are greatly reduced. It also ensures that no one using CADI uses clear-text passwords in Configuration files.
- Identity - CADI Framework obtains the Identity of any callers by delegating to CSO approved Identity servers on behalf of the client, so that Applications can be assured of who is talking to them.

AAF Functional Description

Entities within AAF

Namespaces

- A Namespace, in AAF, is the ensemble of Roles, Permissions and Identities controllable within the domain assigned to a member of the Organization's chain-of-command.
- Namespaces are known by domain, in dot-delimited form. ex: com.att.aaf or com.att.wfa, and they are hierarchically managed.

People in Namespaces

- Owner (Responsible)
- A key feature of how AAF works for an Enterprise is by supporting federated responsibility. This responsibility is clearly delineated by the owner entry. Organizations (i.e. companies) may establish their own policies

Roles

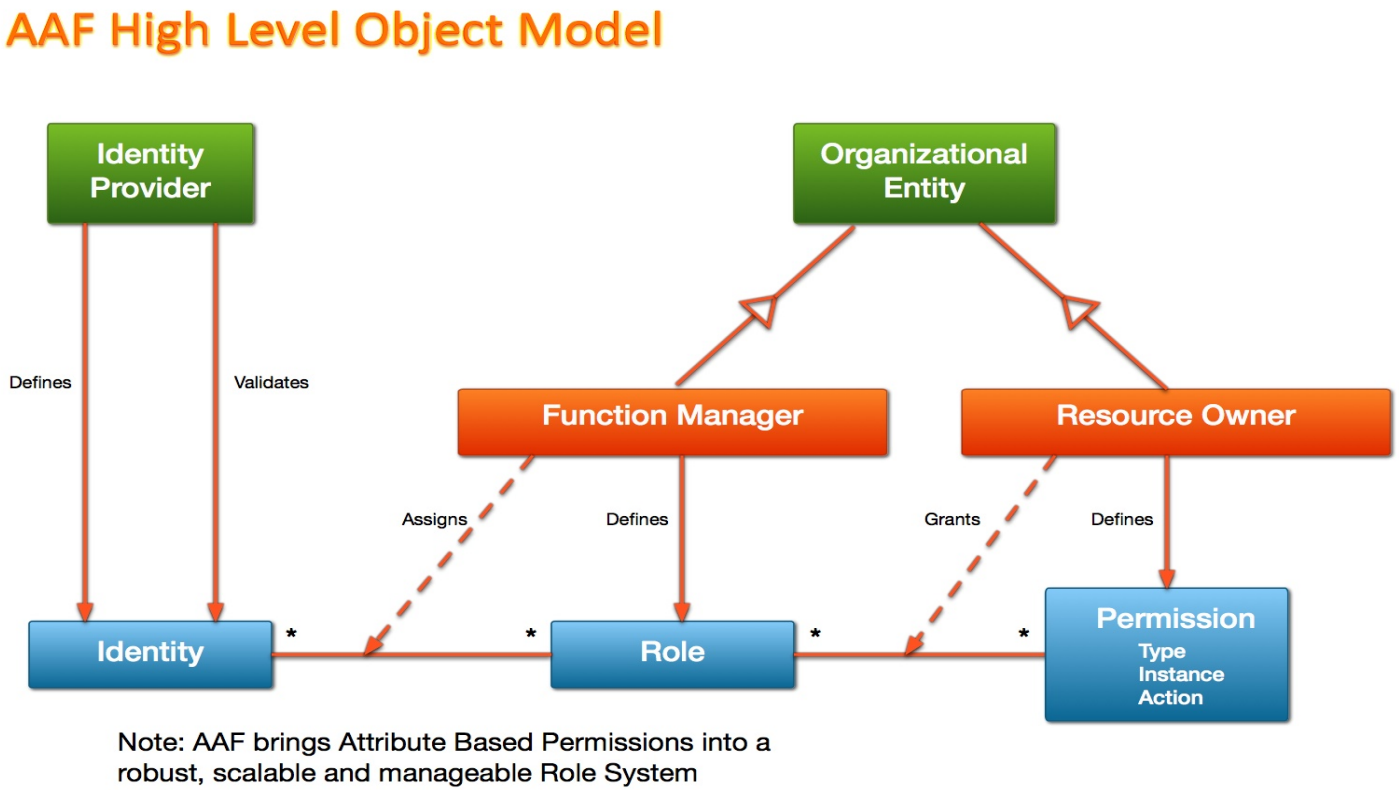
- I) "Roles" is a bit of an overloaded term in Security software. It has unfortunately been typically used as a flat, unscoped Group, known only to the Application.
- Typical examples such as "user" and "admin" have no meaning outside the immediate context of the application. "admin"??? "admin" of what? What are the behaviors allowed for "admin"? Is the "admin" of Application A, the same as Application B? (answer, no!)

Permissions

Permissions are the other side of the decoupled Authorization equation. Permissions represent some resource that needs to be protected by the Application in question. examples:

- A GUI Admin page should be available to only those who job function is to administer this app.
- SPI Data that this App accesses should only be accessible to those who are allowed to see SPI data.
- Full data reporting dumps should only be accessible to Audit teams.

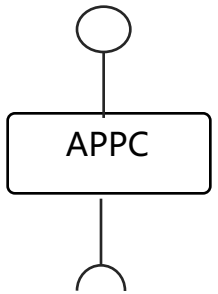
AAF Object Model



DMaaP Functional Description

- Data movement as a platform(DMaaP) is a premier platform for high performing and cost effective data movement services that transports and processes data from any source to any target with the format, quality, security, and concurrency required to serve the business and customer needs.
- DMaaP consists of three major functional areas:
  1. **Data Filtering** - the data preprocessing at the edge via data analytics and compression to reduce the data size needed to be processed.
  2. **Data Transport** - the transport of data intra & inter data centers. The transport will support both file based and event based data movement. The Data Transport process needs to provide the ability to move data from any system to any system with minimal latency, guaranteed delivery and highly available solution that supports a self-subscription model that lowers initial cost and improves time to market.
  3. **Data Processing** - the low latency and high throughput data transformation, aggregation, and analysis. The processing will be elastically scalable and fault-tolerant across data centers. The Data processing needs to provide the ability to process both batch and near real-time data.
- This service is build using Apache Kafka and Restful API is created for message publishing, subscribing and admin activities

APPC (1/2)



Definition:
Application Controller (APPC) performs the functions to manage the lifecycle of VNFs and their components.

- It provides a comprehensive set of controller actions such as Configure, Modify Configuration, Start, Stop, Migrate, Restart, Rebuild, and so on. Consult readthedocs documentation for the full set of APIs offered/supported.
- It supports a set of standard VNF interfaces (Netconf, Chef, Ansible.), and
- Is designed to be self-service using a model driven architecture that provides a layer of abstraction making APPC completely service, VNF, and site agnostic.

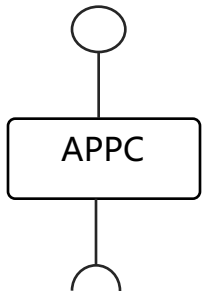
Information about the APPC architecture and provided APIs can be found in the APPC User Guide and LCM & OAM API Guides on readthedocs

.

Provided Interfaces: (refer to APPC User Guide for details)

Interfaces	Service	Purpose / Comments
South-bound Adapters	REST to OpenStack or MultiCloud	APPC interacts with southbound adapters for VNF Lifecycle Management Actions
NETCONF	NETCONF	NETCONF Adapter facilitates communication between APPC and a VNF that supports NETCONF protocol
Chef Server	HTTP/HTTPS	Used for communication between the APPC Chef Adaptor and the Chef Server.
Ansible Server	HTTP	Used for communication between the APPC Ansible Adaptor and the Ansible Server.
LCM API	REST/DMaaP	APPC receives LifeCycle Management (LCM) commands from SO and Policy and takes applicable actions on the VNF/VNFC/VM and responds back.
OAM API	REST/DMaaP	APPC receives operational commands: Start, Graceful Stop. APPC reports: status, KPIs. This set of API is for action on the APPC component itself and not the VNFs.

APPC (2/2)



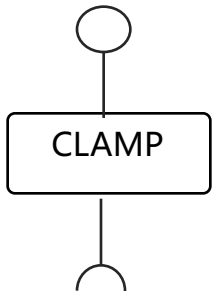
Consumed Interfaces

Component	Service	Purpose
A&AI	REST	APPC retrieves and updates the VNF data (orchestration status) in AAI.
DMaaP	HTTP	APPC sends the Asynchronous responses and Failure events to DMaaP Message Bus and receives action requests from various components of ONAP (SO, Policy, SDC)
SDC	DMaaP	APPC requests and receives notifications from SDC for VNF License Artifacts, TOSCA dependency models, along with others.

Consumed Models:

- TOSCA Dependency Model
- APP-C also support a variety of models, most of which are created by the APPC Configuration Design Tool (CDT). Examples are:
  - Reference Data Model (json)
  - VNF Capabilities Model (json)
  - Template Model (json or xml)
  - Parameter Definition Model (yaml)

CLAMP (1/2)



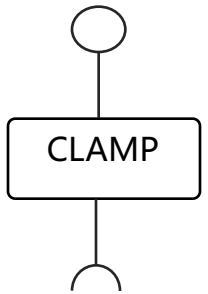
Definition:
CLAMP is a platform for designing and managing control loops.
It is used to design a control loop, configure it with specific parameters for a particular network service,
then It interacts with other systems to deploy/undeploy and execute the control loop.

Information about the CLAMP architecture and provided APIs can be found in the CLAMP User Guide and LCM & OAM API Guides on readthedocs

Provided Interfaces: (CLAMP doesn't expose functional relates API's except the one below, refer to CLAMP documentation for more details)

Interfaces	Service	Purpose/Comments
HealthCheck	REST	Health Check of CLAMP instance

CLAMP (2/2)



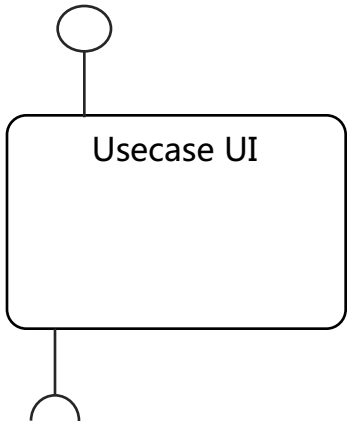
Consumed Interfaces:

Interfaces	Service	Purpose/Comments
SDC	REST	Distribution of service to DCAE
DCAE	REST	Common Controller Framework
Policy	REST(JSON data)	Create Configuration and Operational policy

Consumed Models: (No specific models were used in the above interface except for Policy(json))

Use-case UI

- None



- Catalog Synchronization Interface
- Service Orchestrator Interface
- Inventory Service Interface
- NS/VNF Onboard Interface
- Service Registration Interface
- Data Subscription Interface

Definition:

- Provides portal for service life cycle management
- Provides portal for VNF alarm and performance
- Provides portal for VM alarm and performance

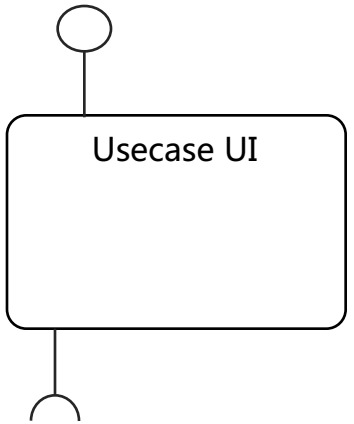
Provided Interfaces:

- None

Note: Can be more than one page

Use-case UI

- None



- Catalog Synchronization Interface
- Service Orchestrator Interface
- Inventory Service Interface
- NS/VNF Onboard Interface
- Service Registration Interface
- Data Subscription Interface

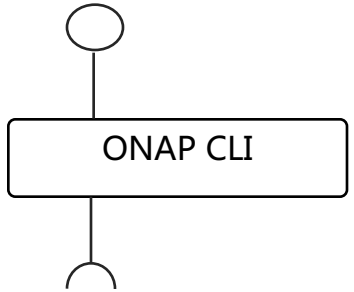
Consumed Interfaces:
- Catalog Synchronization Interface, from: SDC
- Service Orchestrator Interface, from: SO
- Inventory Service Interface, from: Available and Active Inventory
- NS/VNF Onboard Interface, from: VF-C
- Service Registration Interface, from: MSB
- Data Subscription Interface, from: DMaaP

Consumed Models:
- Network Service Descriptor
- VNF Descriptor

Note: Can be more than one page

ONAP CLI

micro-service discovery commands
external system and VNF cloud on-boarding commands
customer and subscription management commands
resource on-boarding commands
Product and service management commands
network service life-cycle management commands



micro-service discovery Interface (MSB)
external system and VNF cloud on-boarding interface (AAI)
customer and subscription management interface (AAI)
resource on-boarding interface (SDC)
Product and service management interface (SDC)
network service life-cycle management interface (SO)

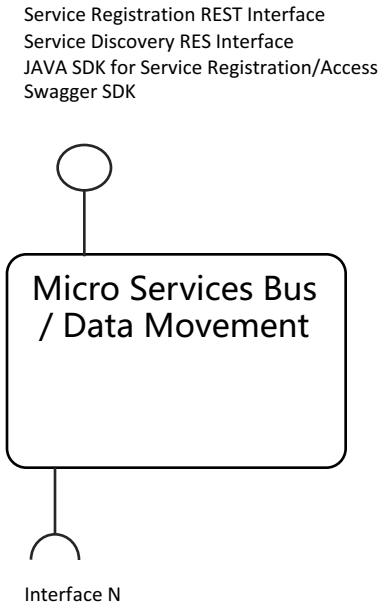
Definition:
- Provides Open CLI Platform (OCLIP) – Industry first Model-driven command line interface platform
- Provides commands for performing end-end service on-boarding and life cycle management
- Provides command console for Linux and web platforms.

Provided Interfaces:
- micro-service discovery
  - Provides commands for micro service discovery and registration
- external system and VNF cloud on-boarding
  - Provides commands for registering/unregistering VIM, VNFM, EMS and SDNC
- customer and subscription management
  - Provides commands for subscription and service-type life cycle management
- resource on-boarding
  - Provides commands for adding VNF modules
- Product and service management
  - Provides commands for design time creation and management of VF and NS
- network service life-cycle management
  - Provides commands for creating and managing Network services (NS)

Consumed interface Interfaces:
- REST API from SDC, SO, AAI, MSB.

Consumed Models:
- Open Command Specification (OCS) 1.0

Microservices Bus (1/2)

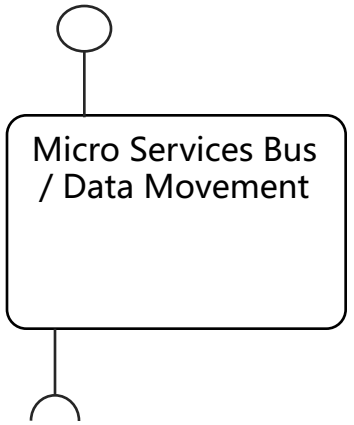


Definition:

- Provides a reliable, resilient and scalable communication and governance infrastructure to support ONAP Microservice Architecture(OMSA).
- Provides RESTFul API for service registration/discovery
- Provides JAVA SDK for service registration and access
- Provides a transparent service registration proxy with OOM-Kube2MSB
- Provides a transparent service communication proxy which handles service discovery, load balancing, routing, failure handling, and visibility-Internal API Gateway(Implemented) and Mesh Sidecar(WIP)
- Provides an External API Gateway to expose ONAP services to the outside world
- Swagger SDK for auto-generate the client SDK in different languages for every micro-services in ONAP. Also it deploys the generated SDK into the nexus.

Microservices Bus (2/2)

Service Registration REST Interface
Service Discovery RES Interface
JAVA SDK for Service Registration/Access
Swagger SDK



Provided Interfaces:
- Service Registration REST Interface
  - Provides the RESTFul interface to register ONAP services.
- Service Discovery REST Interface
  - Provides the RESTFul interface to discover ONAP services.
- JAVA SDK
  - Provides JAVA SDK to register ONAP services.
  - Provides JAVA SDK to access ONAP services.

  Note: ONAP components don' t need the MSB interfaces to leverage the transparent service registration proxy (kube2msb) and communication proxy (API Gateway).

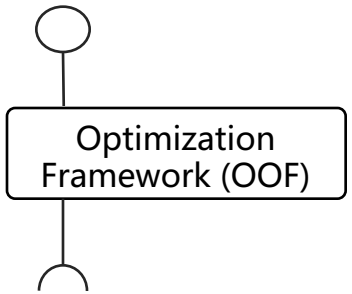
-Swagger SDK
  - provides Mvn plug-in settings for (JAVA)client sdk generation

Consumed interface Interfaces: (None)

Consumed Models:
- Swagger for RESTFul APIs definition
- Service definition in kubernetes config files for automatically service registration by kube2MSB

Optimization Framework - R2 (1/4)

- Service Orchestrator (SO)
- Change Management Portal



- Policy Interface
- Inventory Service Interface
- Service/Policy Models Interface
- Infrastructure Metrics Interface

Definition:
- ONAP Optimization Framework (OOF) provides functionality for creating and running optimization applications by leveraging a policy-driven, declarative approach.
- Supports different specializations (applications) with specific domain and optimization needs.
- Initial specialization scopes include, but are not limited to, VNF placement support via Homing and Allocation Service (HAS) with different use cases, and change management scheduling service.
- For R2, the OOF will support vCPE use case (as a MVP).
- For R2, the OOF will demonstrate an example of Change Management Scheduling Optimization using the OOF design framework with simple, representative constraints.

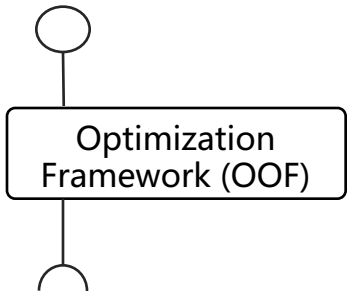
Provided Interfaces:
- Placement Optimization Interface
  - Provides functionality for the Service Orchestrator to specify placement services [vCPE use case]
- Change Management Portal Interface
  - Provides functionality for calculating schedules that satisfy time constraints and conflicts [R2 focus on a demonstration of the design framework; alignment with Change Management scheduling use case]

Consumed Interfaces:
- Policy Interface, from: Policy
- Inventory Service Interface, from: Available and Active Inventory
- Service/Policy Models Interface, from: SDC
- Infrastructure Metrics Interface, from: Multi Cloud

Consumed Models:
- Policy Models (constraints) and License Models from: SDC
- Infrastructure Metrics Models, from: Multi Cloud

Optimization Framework - R3+ (2/4)

- Service Orchestrator (SO)
- Change Management Portal



- Policy Interface
- Inventory Service Interface
- Service/Policy Models Interface
- Infrastructure Metrics Interface

R3 and Beyond

- ***Application Metrics Interface***

Note on Functionality for R3 and beyond:
While the items listed below are not part of MVP for R2, the OOF team is initiating discussions and collaborations on these during the R2 time frame, so that the OOF is aligned with the use cases that will mature in R3 and beyond.

Additional Definitions for R3 and beyond:
- For R3 and beyond, the OOF will focus on supporting multiple R3 and beyond use case (along with R2 use cases) [supporting VoLTE, VNF/Service scale-in/out (higher order control loop), and RAN-5G].
- Synchronization with Change Management Scheduling use case.
- Providing additional sample/example optimization applications as documentation.
- Proving initial Knowledge Base on OOF (building blocks and recipes for creating complex applications).
- Further alignment with S3P objectives.

Additional Provided Interfaces for R3 and beyond:
- Placement Optimization Interface
  - For R2, the focus is on supporting vCPE use case;
  - for R3 and beyond, the OOF will also support VoLTE, VNF/Service scale-in/out (higher order control loop), and RAN-5G.
- Change Management Portal Interface
  - For R2, the focus is on a demonstration of OOF with simple, representative constraints; this effort will be aligned with the Change Management scheduling use case (for R3 and beyond)

Additional Consumed Interfaces for R3 and Beyond:
- Application Metrics Interface, from: DCAE [R3 and Beyond]
- These are in addition to Policy, Inventory, SDC Models, and Infrastructure metrics from R2

Additional Consumed Models for R3 and beyond:
- Application Metrics Models, from: DCAE

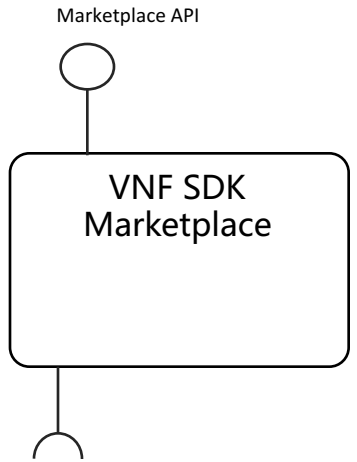
Optimization Framework - Interface Definitions (3/4)

Optimization Request Interface	
Provided Service	Homing Service API (HAS-API)
Supplementary Information	The OOF will provide a list of possible candidates for placement, which the service orchestrator can use to instantiate the service
Interface Provider(s)	Optimization Framework
Provided Capabilities	- Optimization Information Request

Optimization Framework - Interface Definitions (4/4)

Scheduling Optimization Request Interface	
Provided Service	Change Management Scheduling API (CMSO API)
Supplementary Information	The OOF will provide a possible schedule of VNF actions (e.g. upgrades) that satisfy the constraints on availability periods and constraints related to conflicts
Interface Provider(s)	Optimization Framework
Provided Capabilities	- Optimization Information Request

VNF SDK (VNF package validation) (1/2)



Definition:
- Provides functionality to upload, verify, and download VNFs
- VNF Vendors upload VNFs through the Marketplace portal
- Marketplace runs package validation tests
- VNFs are made available for Onboarding

Marketplace API:
- Upload/Re-upload VNF
  - Upload VNFs to the Marketplace
- Download VNF
  - Download VNFs
- Query VNF
  - Query VNFs by CSAR ID or conditions (name, version, type, provider)
- Delete VNF
  - Delete VNF from the Marketplace

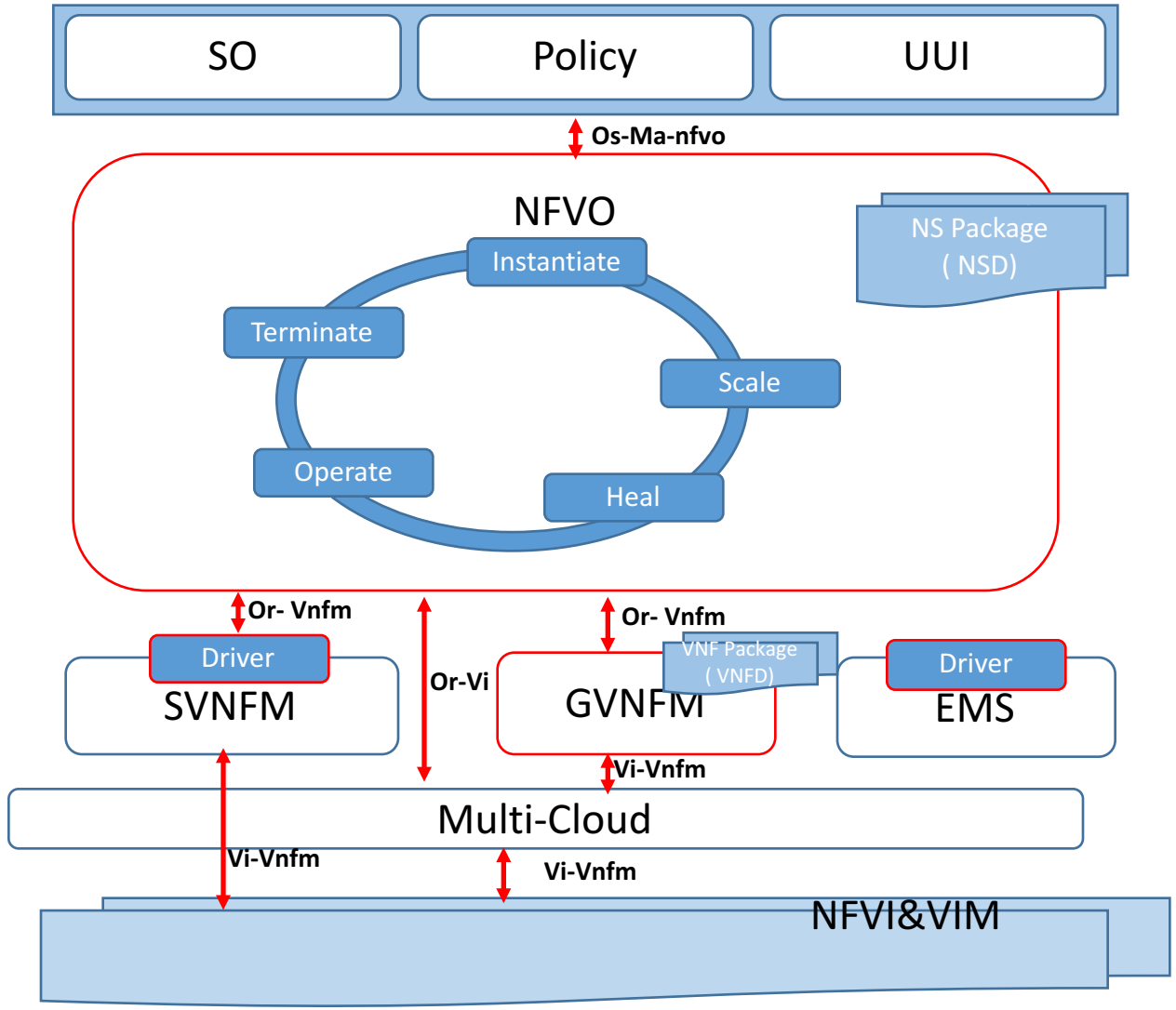
Consumed Models: SOL-004 (Beijing)

Note: Can be more than one page

VNF SDK - Interface Definitions (2/2)

Marketplace API	
Provided Service	Upload, Download, Query, Delete
Supplementary Information	The VNFSDK APIs are documented on the ONAP Wiki: https://wiki.onap.org/display/DW/API+Documentation
Interface Provider(s)	VNF SDK Marketplace
Provided Capabilities	VNF Pre-onboarding upload/download/validation/query/delete

VF-C Components



NFVO Functions

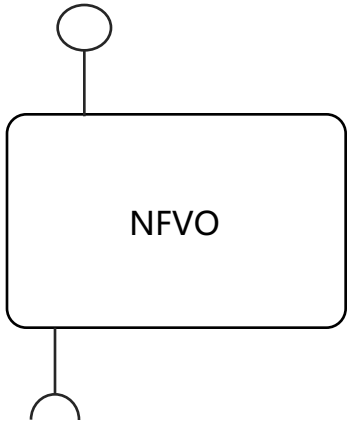
- Support NS Lifecycle Management including NS instantiate, scale, heal, operate (query /update/…)and terminate. Most of NSLCM interfaces align with SOL005 Os-Ma-nfvo reference point
- Support integration with multi VNFMs via drivers which include vendors VNFM and generic VNFM. The interfaces between NFVO and driver comply with Or-Vnfm reference point.
- Support integration with multi VIM via Multi-Cloud
- NFVO also supports integration with vendor EMS via driver

GVNFM Functions

- Support VNF Lifecycle Management, including VNF deploy, scale, heal, operate (start/stop/restart/…), update and terminate, etc
- Support multiple VNFs and multi-type VNFs from different vendors
- Support multiple VIM environments and multi-type VM environments based on VM or Docker

VF-C – NFVO (1/2)

- Network Service LCM Interface
- VNF Operation Granting Interface
- NS package management Interface
- VNF package management interface



VNF LCM Interface
Inventory Service Interface
Catalog Synchronization interface
Generic VIM Interface
Date report interface
Tosca parser interface
Optimization Request Interface

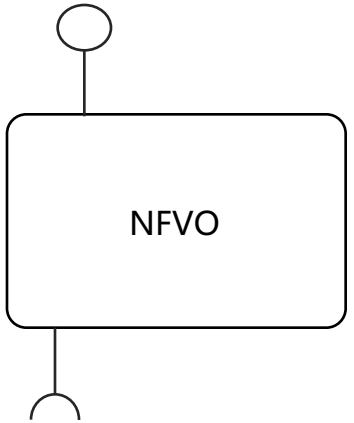
Definition:
- Provides reference implementation of NFVO in ETSI MANO architecture
- Provides Network service life cycle management
- Provides NS/VNF layer's FCAPS management

Provided Interfaces:
- Network Service LCM interface
  - Provides Network Service LCM interface
    (NS instantiate/scale/heal/terminate/query/…)
- VNF Operation Granting interface
  - Provides VNF Operation Granting interface and make granting decision
- NS package management interface
  - Provides runtime NS package management interface
- VNF package management interface
  - Provides runtime VNF package management interface

Note: Can be more than one page

VF-C – NFVO (2/2)

- Network Service LCM Interface
- VNF Operation Granting Interface
- NS package management interface
- VNF package management interface



VNF LCM Interface
Inventory Service Interface
Catalog Synchronization interface
Generic VIM Interface
Data report interface
Tosca parser interface
Optimization Request Interface

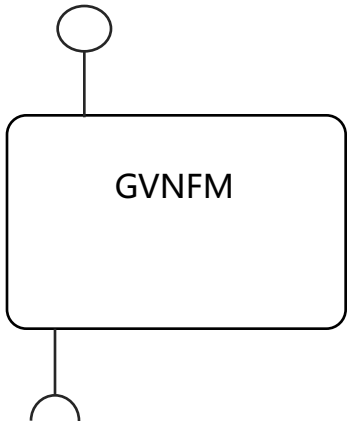
Consumed Interfaces:
- VNF LCM Interface, from: Generic VNFM controller, SVNFM
- Inventory Service Interface, from: Available and Active Inventory
- Catalog Synchronization Interface, from: SDC
- Generic VIM Interface, From: Multi-cloud
- Data report Interface, From: DCAE
- Tosca parser Interface, From: Modeling
- Service registration and discovery, From: MSB
- Optimization Request Interface, from: Optimization Framework -TBD

Consumed Models:
- Network Service Descriptor
- VNF Descriptor
- VES data format
- Bpmn Workflow

Note: Can be more than one page

VF-C - GVNFM

- VNF LCM Interface



- Catalog and notification Interface
- Inventory Interface
- Tosca parser Interface
- Generic VIM Interface
- VNF config interface

Definition:
- Provides the Generic VNFM
- Provides the VNF life cycle management
- Supports NFVO to implement Network service LCM management

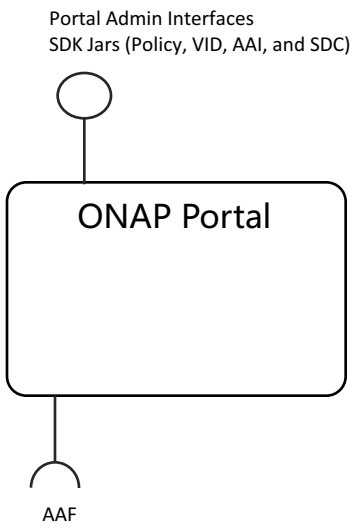
Provided Interfaces:
- VNF LCM interface
  - Provides the VNF LCM interface (VNF instantiate/terminate/query/…)

Consumed interface Interfaces:
- Catalog and notification Interface, from: NFVO
- Inventory Interface, from: A&AI
- Tosca parser Interface, From: Modeling
- Generic VIM Interface, From: Multi-cloud
- Service registration and discovery, From: MSB
- VNF Config interface: from VNF -TBD

Consumed Models
- VNFD

Note: Can be more than one page

ONAP Portal



Definition:

- Portal is a platform that provides the ability to integrate different ONAP applications into a centralized Portal Core.
- It allows decentralized applications to run within their own infrastructure while providing common management services and connectivity.
- It provides capabilities including application onboarding & user management, and hosted application widgets.
- Using the provided SDK, application developers can leverage the built-in capabilities (Services / API / UI controls) along with bundled tools and technologies.

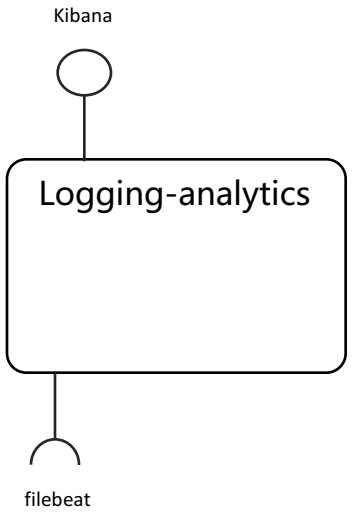
Provided Interfaces:

- Portal Admin Interfaces
  - Onboarding and User management
- SDK Module
  - SDK will be used as a base for any ONAP application being built.
  - Components using SDK jars: Policy, VID, AAI, and SDC
  - SDK bundles together a host of different tools, technologies, and standards that will assist the teams during the development phase.
  - SDK includes reusable UI components, authentication & authorization components, visualization & reporting engine, collaborative services, workflow manager, GIS / Map, web component, and widget development framework.

Consumed interface Interfaces:

- Restful APIs for fetching available roles to portal's onboarded application users. from: AAF

Logging-analytics (1/2)



Definition:
- Provides ELK stack where logs are streamed from filebeat sidecar containers per component

Provided Interfaces:
- Kibana UI:
  - dashboard for indexed logs
- Logstash:
  - Consumes logs pushed by filebeat containers into elasticsearch indexer

Consumed interface Interfaces:
- Filebeat container per microservice exposing logs through a Kubernetes PV:

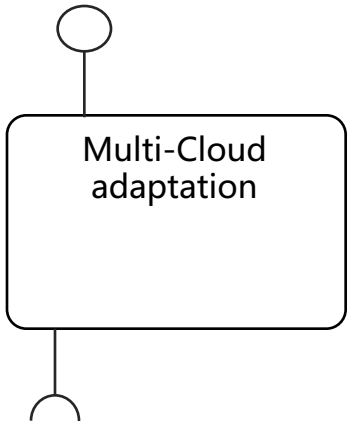
Consumed Models: none

Logging-analytics (2/2)

Logging-analytics Interface	
Provided Service	Kibana dashboard, search service on top of elaticsearch index of ONAP logs – via filebeat
Supplementary Information	
Interface Provider(s)	AA&I, APPC, SO, Policy, Portal, SDC, SDNC, VID
Provided Capabilities	- Search logs using lucene search format - Dashboard of configurable search views of logs

Multi-Cloud Adaptation (1/2)

- Resource LCM Interface
- VIM registry/un-registry Interface
- VIM FCAPS management Interface
- VIM capacity/capability query



ONAP Interface
- Inventory Service Interface
- DMAAP interface
- Logging interface

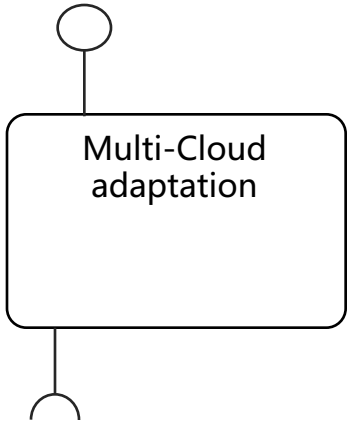
The third party Cloud interface
- OpenStack Interface
- VMware VIO interface
- Wind River Titanium Interface

Definition:
- enable ONAP to deploy and run on multiple infrastructure environments, including OpenStack and its different distributions, public and private clouds, and micro services containers, etc.
- provide a Cloud Mediation Layer supporting multiple infrastructures and network backends so as to effectively prevents vendor lock-in.
- decouple the evolution of ONAP platform from the evolution of underlying cloud infrastructure, and minimizes the impact on the deployed ONAP while upgrading the underlying cloud infrastructures independently.

Provided Interfaces:
- Resource LCM interface
  - Support VNF instantiation/termination (Resource create/delete/update/query)
- VIM registry/un-registry Interface
  - Support registry/un-registry of multiple clouds (VIM create/delete/update)
- VIM FCAPS management Interface
  - Support FCAPS required data collection, event/alert/metrics federation functions (message pub/sub)
- VIM capacity/capability query
  - Support placement and scheduling purpose (capacity/capability query/report)

Multi-Cloud Adaptation (2/2)

- Resource LCM Interface
- VIM registry/un-registry Interface
- VIM FCAPS management Interface
- VIM capacity/capability query



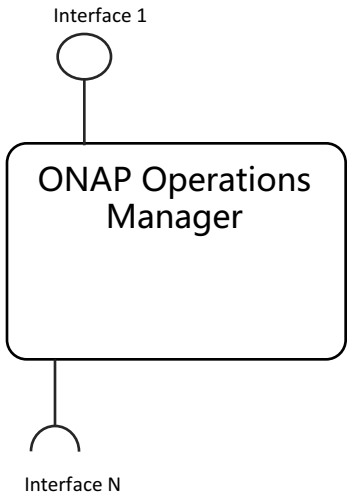
ONAP Interface
- Inventory Service Interface
- DMAAP interface
- Logging interface

The third party Cloud interface
- OpenStack Interface
- VMware VIO interface
- Wind River Titanium Interface

Consumed Interfaces:
- ONAP
  - Inventory Service Interface from Available and Active Inventory
  - Message pub/sub interface from DMAAP
  - Logging interface from Logging project
- The third party Cloud API
  - OpenStack Interface
    - Ocata
    - Mitaka
  - VMware VIO interface (Ocata)
  - Wind River Titanium Interface (Mitaka)

Consumed Models:
- Existing
  - VES data format
- Underworking
  - Hierarchical Infrastructure Resource Information Model
  - FCAPS Data Model

ONAP Operations Manager (OOM)



Definition: OOM is the life cycle manager of the ONAP platform (i.e. SDC, SO, SDNC, etc.). OOM uses the Kubernetes container management system and Consul to provide the following functionality:

- **Deployment** - with built-in component dependency management (including multiple clusters, federated deployments across sites, and anti-affinity rules)
- **Configuration -** unified configuration across all ONAP components
- **Monitoring** - real-time health monitoring feeding to a Consul GUI and Kubernetes
- **Restart** - failed ONAP components are restarted automatically
- **Clustering and Scaling** - cluster ONAP services to enable seamless scaling
- **Upgrade** - change-out containers or configuration with little or no service impact
- **Deletion** - cleanup individual containers or entire deployments

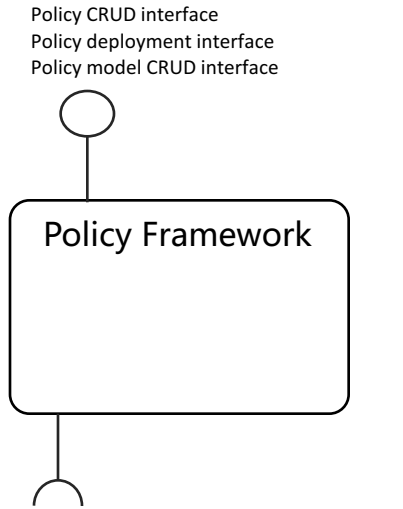
OOM supports a wide variety of cloud infrastructures to suit your individual requirements.

Provided Interfaces: Consul GUI for health monitoring, Kubernetes CLI for platform management

Consumed interface Interfaces: None

Consumed Models: ONAP Deployment Specifications, ONAP Component Configuration Artifacts

Policy Framework (1/2)



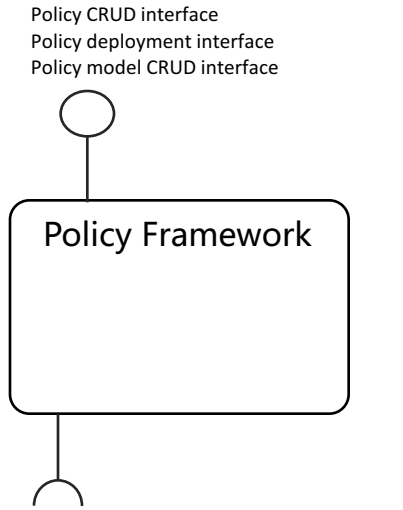
Definition:

- Provides a logically centralized environment for the creation and management of modifiable configurations, rules, assertions and/or conditions to provide real-time decision making on conditions and events that underlie ONAP's control, orchestration, and management functions
- Supports specification, decomposition, distribution and enforcement for various types of policies such as microservice configuration policy, operational policy, decision policy, guard policy, etc.
- Policy scopes include, but are not limited to, infrastructure/network management, products and services, operation automation, and security.

Provided Interfaces:

- Policy CRUD interface:
  - Provide UI and API options to create/query/update/delete configuration, decision and operational policies
- Policy deployment interface:
  - Provide UI and API options to deploy configuration, decision and operational policies
- Policy model CRUD interface:
  - Provide UI and API options to create, query, update and delete policy models

Policy Framework (2/2)



Consumed interface Interfaces:

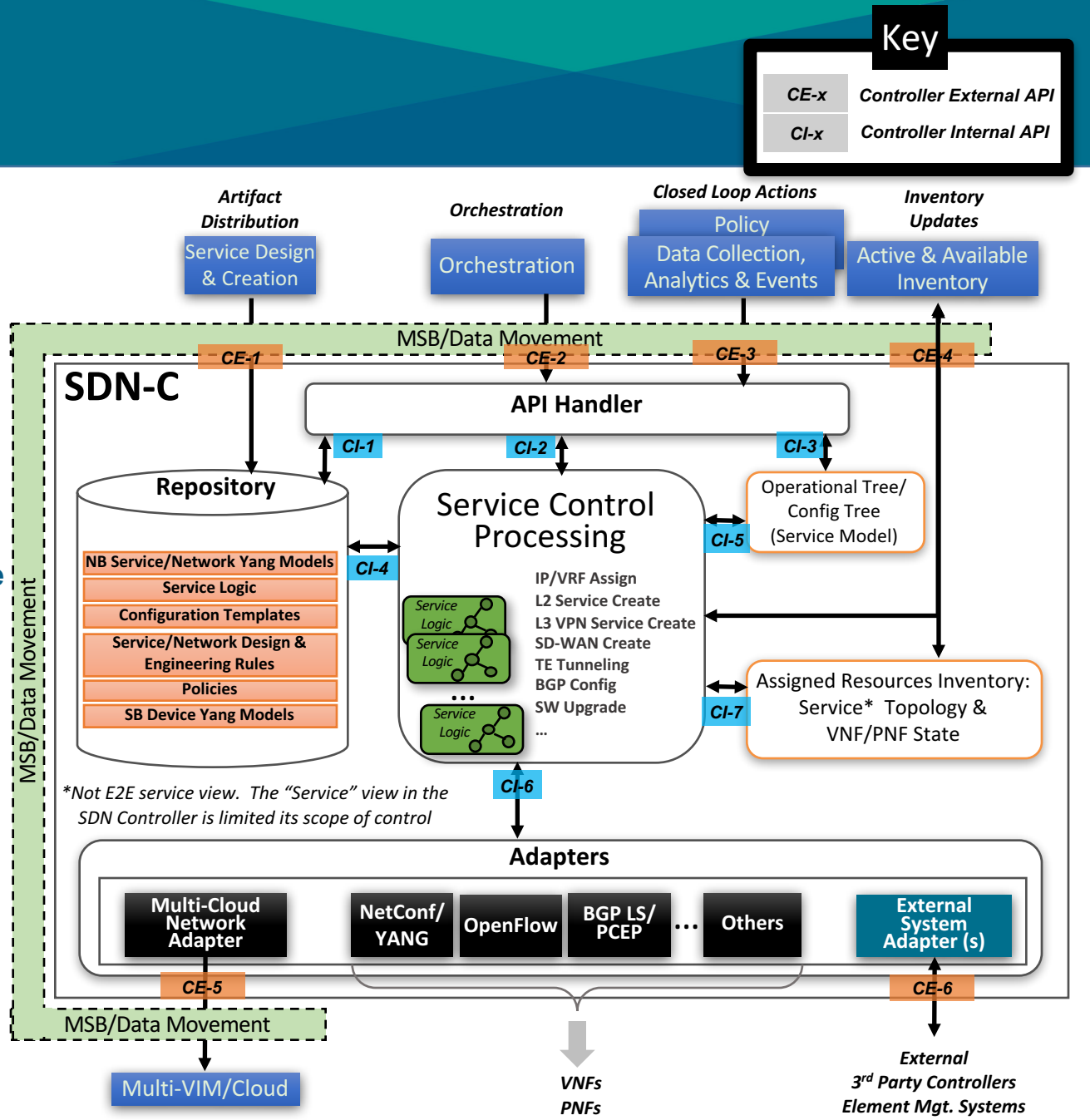
- Message Router Pub/Sub Service Interface, from: Data Movement as a Platform
- Service Registration/Discovery Interface, from: Microservices Bus
- VNF/VM LCM Interface, from: Generic NF controller
- NF Config Interface, from: Generic NF controller
- Inventory Service Interface, from: Available and Active Inventory
- Network Service LCM Interface, from: Service Orchestrator
- Network Service LCM Interface, from: Virtual Controllers (APPC and VFC)
- Service Distribution Interface, from: Service Design & Creation

Consumed Models

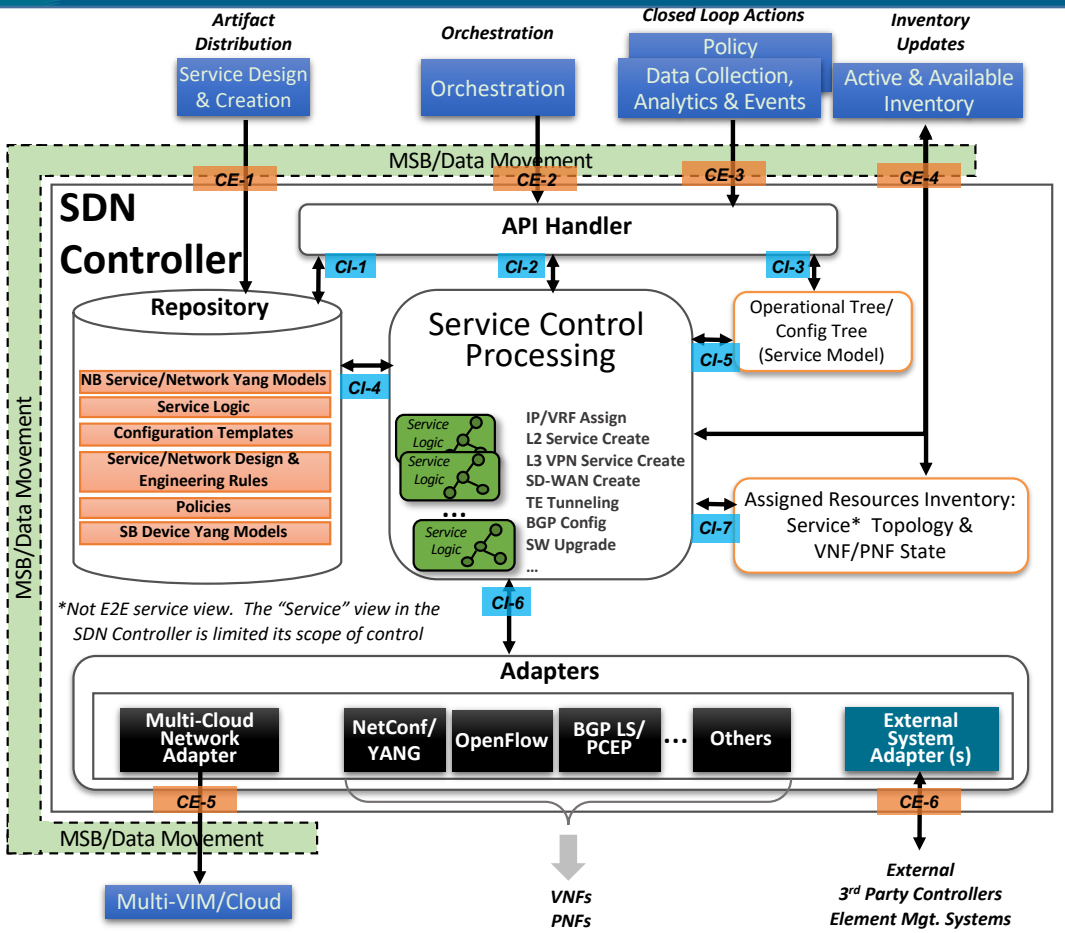
- Operational policy model
- Configuration policy model
- Resource model in the catalog
- Service model in the catalog

SDN-C Functions

- **SDN-C configures and maintains the health of L1-3 VNFs/PNFs and network services throughout their lifecycle**
- **Programmable network application management platform**
  - **Behavior patterns programmed via models and policies**
  - **Standards based models & protocols for multi-vendor implementation**
  - **Extensible SB adapter set supporting various network config protocols, including 3rd party controllers**
  - **Operational control, coordinated state changes across devices, source of telemetry/events, etc.**
- **Manages the health of network services & VNFs/PNFs in its scope**
  - **Policy-based optimization to meet SLAs**
  - **Event-based control loop automation to solve local issues near real-time**
  - **Action executor for outer control loop automation**
- **Local source of truth**
  - **Manages inventory within its scope**
  - **All stages/states of lifecycle**
  - **Configuration audits**
- **Key Attributes of Controllers**
  - **Intimate with network protocols**
  - **Manages the state of services**
  - **Single service/network domain scope per instance**



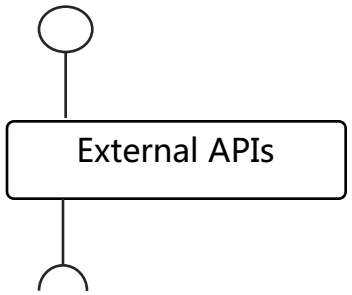
SDN-C External/Internal Interface Definitions



	Interface Definitions	Beijing Release
CE-1	Distribution of artifacts from Service Design and Creation (SDC)	Yes - TOSCA only
CE-2	Service requests from Orchestration (SO)	Yes
CE-3	Closed Loop action requests from Data Collection, Analytics & Events (DCAE)/Policy	Not in scope
CE-4	Inventory retrieval from Active & Available Inventory (A&AI) by Service Logic Inventory updates to Active & Available Inventory by Assigned Resources Inv	Yes – push/pull (no DMaaP topic subscription)
CE-5	Configuration requests for cloud infrastructure networking Lifecycle management requests to Multi-Cloud (e.g., stop/start VM)	Not in scope
CE-6	Lifecycle management or configuration requests to an external controller or system that has responsibility of the target VNF	Yes
CI-1	API Handler looks up or retrieves the corresponding Service Logic instance that maps to NB service request (service/network yang)	Yes
CI-2	API Handler calls Service Control Processing to perform the Service Logic on the target service or network	Yes
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)	Yes
CI-4	Service Control Processing retrieves the Service Logic, Config Templates, Engineering rules, and Policies as part of processing the requested action	Yes
CI-5	Service Control Processing queries and/or updates Operational/Config Trees as part of making changes to the network (VNFs/PNFs)	Yes
CI-6	Service Control Processing requests adapter layer to update/configure VNF/PNF update using the appropriate adapter for the VNF/PNF	Yes
CI-7	Service Control Processing updates the local Assigned Resources Store/Inventory once network updates are made successfully	Yes

External API Definition & Interfaces (1/2)

- Service Catalog
- Service Ordering
- Service Inventory



SDC: Catalog
SO: Service Instantiation
A&AI: Inventory Service Interface

Definition:

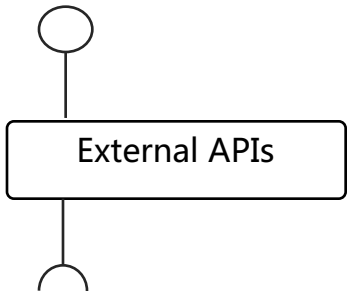
- ONAP External APIs expose the capabilities of ONAP. They allow ONAP to be viewed as a “black box” by providing an abstracted view of the ONAP capabilities.
- External APIs support that an external consumer of ONAP capabilities can be authenticated and authorized. These APIs can also be used for connecting to systems where ONAP uses the capabilities of other systems.
- Provides a clear and unambiguous ONAP service abstraction so that the BSS/OSS can exchange service requirements and service capabilities in a common and consistent fashion.

Provided Interfaces:

- Service Catalog Interface
  - Provides an external view of requestable ONAP Services and the retrieval of the associated Service template (model)
- Service Ordering Interface
  - Provides a mechanism for placing a service order with all of the necessary order parameters represented on the Service template. It allows users to create, update & retrieve Service Orders and manages related notifications.
- Service Inventory Interface
  - Provides a consistent mechanism to query the Service inventory from an external perspective.

External API – Consumed Interfaces/Models (2/2)

- Service Catalog
- Service Ordering
- Service Inventory



SDC: Catalog
SO: Service Instantiation
A&AI: Inventory Service Interface

Consumed Interfaces:
- SDC: Catalog
- SO: Service Instantiation
- A&AI: Inventory Service Interface

Consumed Models:
- Service Descriptor (TOSCA YAML Service Descriptor)

External API – Service Catalog (1/3)

Service Catalog	
Provided Service	Provides an external view of requestable ONAP Services and the retrieval of the associated Service template (model)
Supplementary Information	Service Models are TOSCA representations of the Services that may be requested.
Interface Provider(s)	External API
Provided Capabilities	• Query Service Catalog • Retrieve Service Model

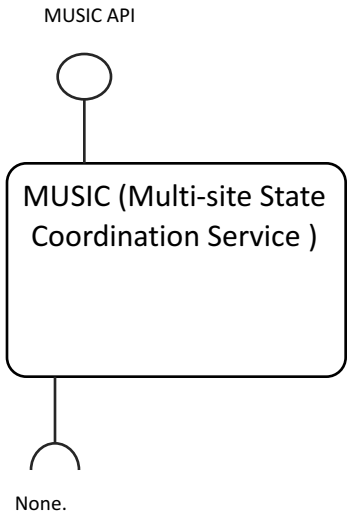
External API – Service Ordering (2/3)

Service Ordering	
Provided Service	Provides a mechanism for placing a service order with all of the necessary order parameters represented on the Service template. It allows users to create, update & retrieve Service Orders and manages related notifications.
Supplementary Information	Service Models are TOSCA representations of the Services that may be requested.
Interface Provider(s)	External API
Provided Capabilities	• Place Service Order • Query Service Order • Retrieve Service Order • Delete Service Order • Retrieve Service Order State • Subscribe to Service Order Notifications • Receive Service Order Notifications

External API – Service Inventory (3/3)

Service Inventory	
Provided Service	Provides a consistent mechanism to query the Service inventory from an external perspective.
Supplementary Information	
Interface Provider(s)	External API
Provided Capabilities	• Query Service Inventory • Retrieve Service • Retrieve Service State

MUSIC- Multi-Site State Coordination Service



Definition: MUSIC is a multi-site state coordination/management service with a rich suite of recipes that ONAP components/micro-services can simply configure and use for their state-management needs both within and across sites. At its core, MUSIC provides a scalable sharded eventually-consistent data-store (Cassandra) wherein the access to the keys can be protected using a locking service (built on Zookeeper) that is tightly coupled with the data-store.

MUSIC also supports Consul based distributed KV Store and its features include ability to store default configuration settings, dynamic changes to the configuration and propagating the changes to running service instances

Provided Interfaces:

- The MUSIC API is a REST API used to read and write state and manage access to it through a locking service.
- API to manage (Add/Modify/Delete) configuration/settings.

Consumed Interfaces - None.

Consumed Models - None.