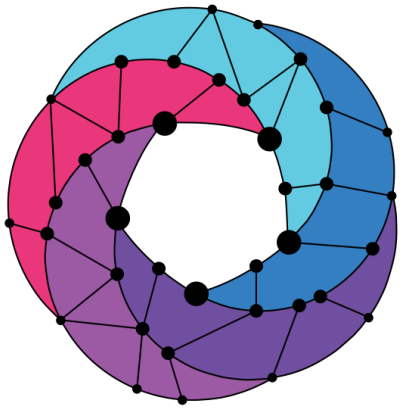


Open Source MANO

**OSM White Paper**

# OSM SCOPE, FUNCTIONALITY, OPERATION AND INTEGRATION GUIDELINES

**A White Paper prepared by the OSM End User Advisory Group**

**Issue 1**
**February 2019**

ETSI
06921 Sophia Antipolis CEDEX, France
Tel +33 4 92 94 42 00
info@etsi.org
www.etsi.org

# Authors


### EUAG contributors

| | |
|---|---|
| Andy Reid | BT (EUAG Chairman) |
| Andrés González | Telenor |
| Antonio Elizondo Armengol | Telefónica |
| Gerardo García de Blas | Telefónica |
| Min Xie | Telenor |
| Pål Grønsund | Telenor |
| Peter Willis | BT |
| Phil Eardley | BT |
| Francisco-Javier Ramón Salguero | Telefónica (Editor) |

### OSM TSC/MDL review

| | |
|---|---|
| Adam Israel | Canonical |
| Alfonso Tierno Sepúlveda | Telefónica |
| Francesco Lombardo | EveryUp |
| Gianpietro Lavado | Whitestack |
| Mark Shuttleworth | Canonical |
| Matt Harper | RIFT.io |
| Michael Marchetti | Sandvine |
| Vanessa Little | VMware |

### ETSI support and editorial review

| | |
|---|---|
| Claire Boyer | ETSI |
| Silvia Almagia | ETSI |

# Contents

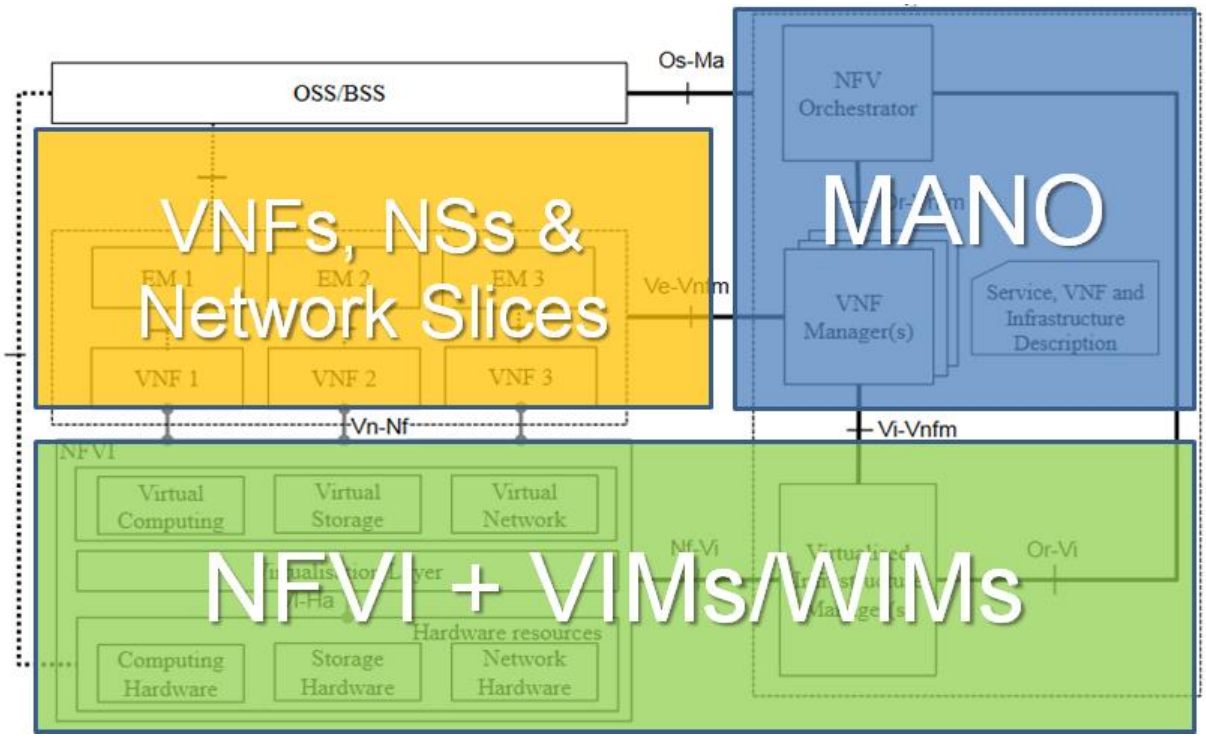
# Introduction

There are a number of new technologies which currently underpin the development in the telecoms industry, many of which are brought together under 5G mobile. Much of the attention of 5G has been focused on developments in the radio system, which promise a significant increase in the download rates available to mobile devices. However, much the more radical promise 5G is built on developments which are less visible. At the heart of these less visible developments is Network Functions Virtualisation (NFV), which enables a quantum change in the flexibility of mobile networks and their ability to support a much wider variety of services, many of which will exploit connected sensor devices which come in the broad banner of the Internet of Things (IoT) as well as new applications emerging in the automotive industry.

The power of NFV is that it enables the full automation of many processes that were previously manual and slow. These processes include the deployment of service-specific functionality into the network which is the basis of many of the plethora of anticipated IoT applications. In contrast to current manual processes, the deployment of service specific functionality to wherever it is most suitable in the network under NFV can be cheap and fast using full automation. It is therefore NFV which a key technology component the anticipated explosion in services with 5G. Of course, NFV is not restricted to mobile access and this process automation will have a similar impact on fixed networks.

NFV can be broadly split into three parts:

1. **The NFVI and VIMs/WIMs**. The first part is the NFV infrastructure (NFVI) which hosts virtual machines and/or containers and connects them together with virtual links (VLs). For the purposes on this white paper, the infrastructure management systems (VIMs and WIMs) which control the creation of virtual machines (VMs), containers and virtual links are also include with the infrastructure.
2. **VNFs, NSs, and Network Slices**. The second part is the collection of VNFs themselves including the interconnected composition of VNFs into network services (NSs) and the composition and sharing of NSs to form network slices. The VNFs are interconnected compositions of specific VMs and/or containers which are hosted on the NFVI.
3. **Management and Orchestration (MANO)**. The third part is the management and orchestration system which controls the life cycle of the VNFs, NSs, and network slices, controls and maintains their configuration, and monitors their in-life health and performance.




**NFV partitioning**

Open Source MANO is a solution to this third part of NFV and this gives OSM its overall scope. OSM aims to support the widest range of NFVI, VIMs, WIMs as well as the widest possible range of VNFs, NSs, and Network Slices. As well as covering the widest possible range of NFVI and hosted functions and services, OSM is an orchestration and management system which manages life-cycle, configuration, and in-life aspects of the hosted functions.

This white paper sets out in more detail the scope and functionality of OSM including how OSM interfaces to other systems. These other systems to which OSM interfaces, include the other parts of NFV as well as non-virtualised network functions and existing OSS/BSS systems.

**Chapter 2** describes the general architectural approach of OSM both to the way it is structured internally and the way in which it interfaces to other systems. OSM takes a layered approach to services and functions and reflects this in its own architecture as well as its interfacing to other systems. OSM offers a service management interface (the OSM northbound interface) to its clients in a higher service layer and constructs services as requested. OSM does this by consuming services from lower service layers, notably from the NFVI through service management interfaces provided VIMs and WIMs. In addition to this *service platform view*, this chapter also describes a *platform operation view* used by OSM to include interactions with other systems which support administration and monitoring with the service layer which OSM controls.

**Chapter 3** develops the *service platform view* and details the scope and functionality of the OSM northbound interface though which the client service layers can request and manage the network services, component VNFs, and network slices characteristic of the service layer managed by OSM. It

then describes the way OSM abstracts the services supplied by the heterogeneous mix of infrastructure services requested and managed through the variety of VIM and WIM interfaces.

**Chapter 4** develops the *platform operation view* detailing the scope and functionality of the essential housekeeping functions of security management including authentication and authorisation along with data logging and exportation of data to external systems.

Finally, **Chapter 5** describes how OSM integrates with other existing systems and shows how OSM can be introduced into a network operator's existing OSS/BSS environment without requiring any radical upheaval in OSS/BSS and the processes they support. The chapter details integration interfacing in both the service platform view as well as the platform operation view.

# Service Platform view and Layered Service Architectures

## Services and their lifecycles

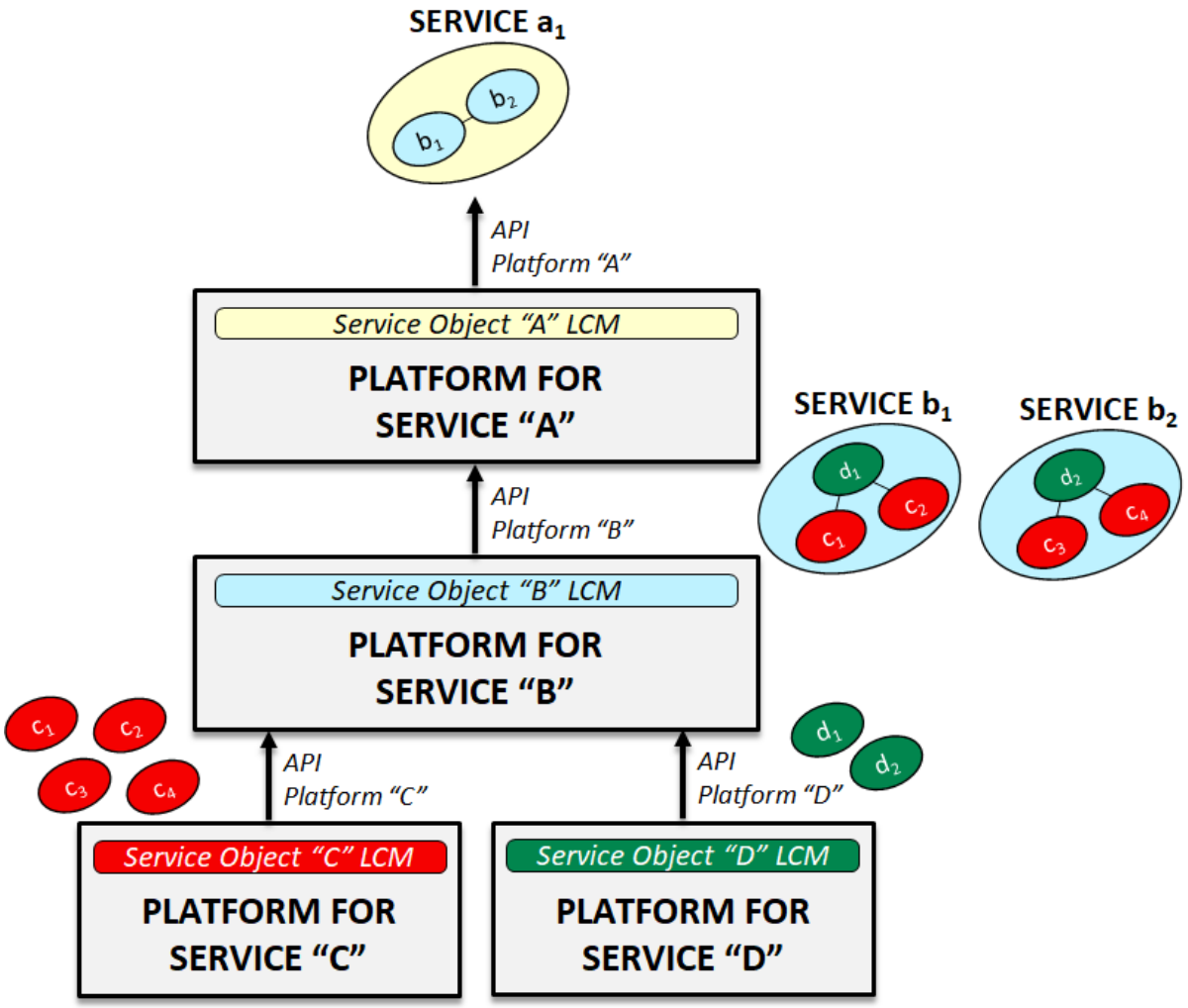
The provision of services of a growing sophistication and functionality usually requires a similar degree of complexity in the internals of those services. Likewise, the new constructs for the new wave of SW-defined networks that are intended to bring significant new functionalities and flexibility, and also come with an internal machinery that is far from being trivial, and a proper division of that intrinsic complexity is required in order to handle it in an efficient manner.

One of the most obvious examples of such a duality between functionality and inner complexity is the case of a Network Slice, one of the key concepts that come with 5G networks. Unsurprisingly, the complexity of a network slice is high, but the good news is that it is designed in a manner that favors a manageable split into nested components, which, in turn, can be subdivided (and, hence, managed) into simpler elements. Thus, a slice can be split in a set of Network Services, and such Network Services can be split, in turn, into a set of virtual machines, virtual networks, physical nodes and, potentially, transport connections.

Walking the same path but in reverse order —from the simplest components to the more sophisticate ones— is the key for a successful management of these environments. As usually happens in other fields in engineering, this complexity is intended to be more effectively handled in a layered fashion, based on stacked service platforms. Thus, service layers can be stacked to create composite services of growing complexity, up to building service objects of the required level of complexity in each case.

In order to make effective the handling of this variety of "service objects", they are intended to be created and controlled on demand. The creation process of each of these "service objects", along as the rest of the rest of their lifecycle, should be provided "as a service" by a platform specialized on providing a given type "service object". Thus all the "service object" of a given type can be controlled and monitored via the invocation of a well-known API from a well-known platform, which is responsible of that "service object" lifecycle.

Moreover, since these platforms can be layered and invoke the APIs of lower level platforms, platforms in upper layers can easily create and offer "as a service" complex composite objects in a manner that scales in terms of operation in a simple and traceable manner, as depicted in the following figure:




**Service platform architecture**

Examples of these kinds of platforms are an NFV Infrastructure (NFVI+VIM, providing VMs and virtual networks), a SW-Defined Network (e.g. an SDTN, providing long-distance connections), or a Network Service Platform (providing Network Services on demand composed of the other types of elements).

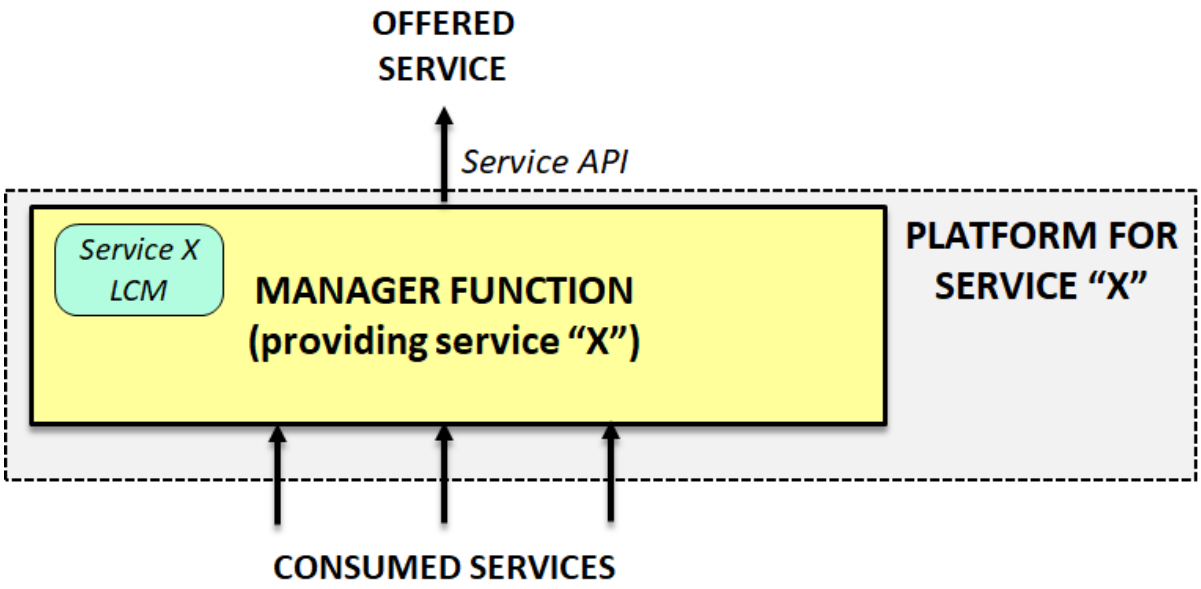
A key aspect of this type of architectures is that **a given platform is not intended to provide "service objects" exclusively to a single "upper" platform**, but it can provide those "service objects" to whichever platforms might request it via its API, bringing to the end to end architecture much higher flexibility. That versatility will be intensively leveraged in the following sections.

## Types of functions in a service platform

Two types of functions can be identified in a service platform: one (and only one) **manager function** and, optionally, some **managed functions**.

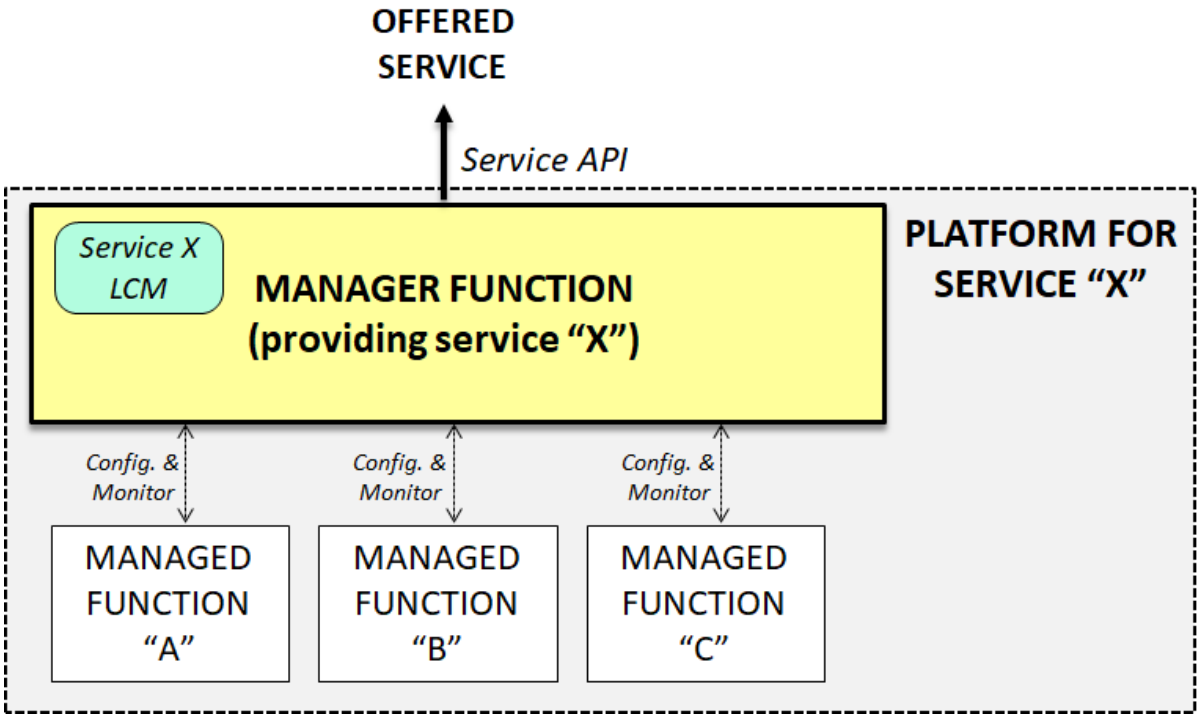
In order to set up a platform, at least, it is required the capability of consuming the APIs from other platforms available southbound, the ability to build a new type of "service object" and handle its lifecycle, and the capability of exposing on demand the new "service objects" via an appropriate API

northbound. Those capabilities are concentrated in the so-called "**Management Function**" of the platform, which owns the lifecycle of the offered "service objects", becoming the heart of the platform itself.



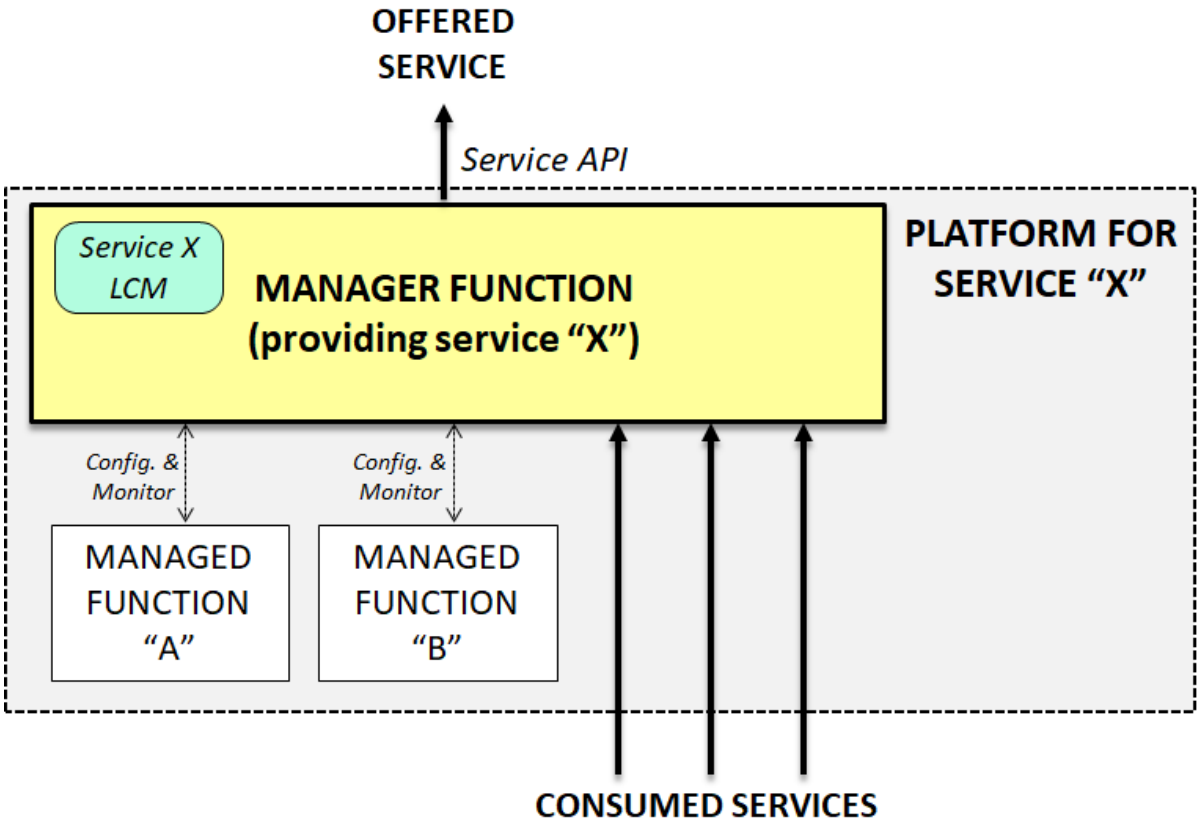

**Manager Function in a Service Platform consuming other services**

However, we may wonder, what happens at the bottom layer of the stack? At the lowest level, platforms do not have service platforms underneath to create services out of them, so they are required to create their own service using the most basic set of available functions (e.g. compute nodes, switches, storage backends, etc.). These other functions are not intended to offer "service objects" on demand, but to offer a standalone functionality, which can be part of a pool and be dynamically configured and monitored by the manager functions. Those final functions (or pools of them) can be referred as **managed functions**. The relation between both types of functions to build a platform placed at the bottom of the stack, is depicted in the following figure:




**Manager Function in a Service Platform at the bottom of the service stack**

In general, it is perfectly normal seeing both types of southbound interactions in many platforms, combining the consumption of some basic services with the control of a number of managed functions which belong to the platform itself, as depicted in the following picture:




**Manager Function in a Service Platform with both managed functions and consumed services**

In must be noted, however, that a managed function is not supposed to belong to more than one platform at the same time, so that conflicts of being controlled by more than one manager can be avoided.

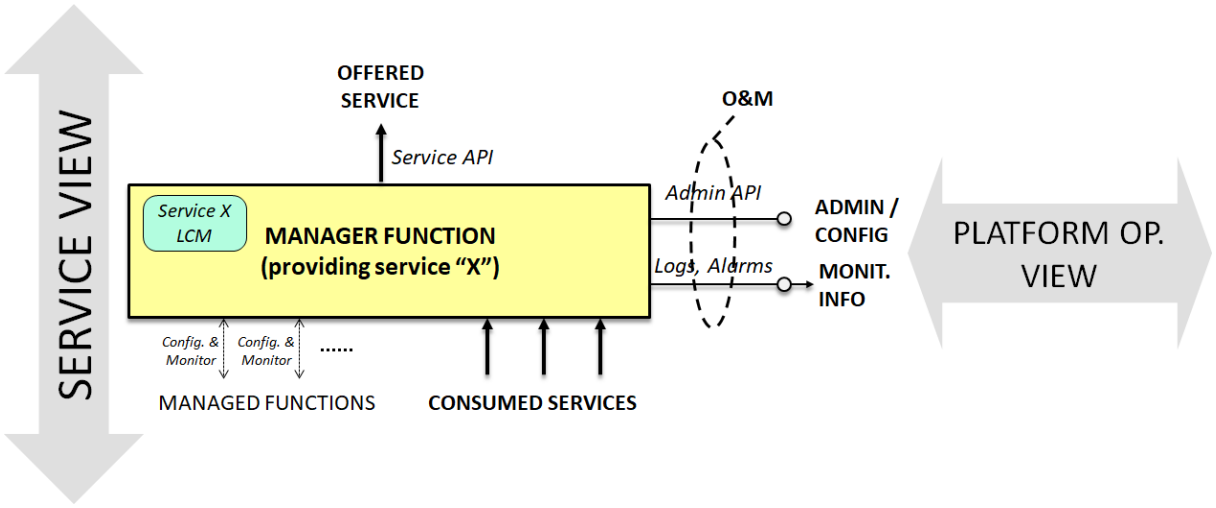
## Service Platform view vs. Platform Operation view

At this point of the discussion, we need to distinguish two complementary views that coexist in these architectures and that can be considered at the same time.

On the one hand, there is the "**Service Platform view**", thoroughly described above, focused on the core functionalities that the architecture intends to offer. In all cases, the working assumption is the existence of potentially frequent and dynamic requests related to those "service objects" that create the main functionality, and which can benefit of large economies of scale in terms of efficient operation given their large number and large dynamicity. These are intended to be highly automated and coordinated, due to the recurrent benefits, and the fact that they tend to create a myriad of new (virtual) components with their own lifecycle, really hard to control without a proper automation and coordination structure. Some of the most prominent examples of this kind are the dynamic provision of Network Services or Network Slices, which often imply a large cascade of dynamic operations across the different layers of the stack that can be triggered on demand that which define a potentially complex composite lifecycle.

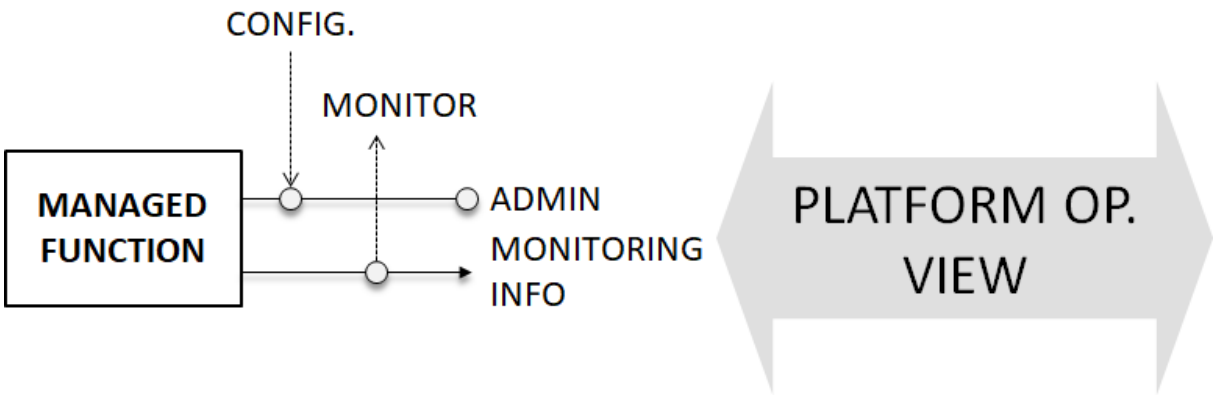
On the other hand, we cannot forget that the functions (manager function and managed functions) that compose the service platforms are also elements that need to be installed and bootstrapped, and may be configured, updated and supervised, having potentially their own lifecycle. These O&M actions over these elements (which constitute the "**Platform Operation view**") are intended to be sporadic and rather static, aiming to set boundary conditions for the "As a Service" aspect of the service platform (e.g. create a new VIM tenant and assign a quota). Usually, the gains associated to automation here are of lower scale, due to the lower recurrence, and may be selectively eased by process-oriented wizards and dashboards.

In the case of the manager functions, both views coexist naturally in practice, with both paths running orthogonally and separated by their purpose (either explicitly or implicitly by profiles), being these O&M tasks a sort of "management of the manager" of a service platform. In practice, the lifecycle of the manager may be synonym of the lifecycle of the service platform as a whole.




**Service Platform view vs. Platform Operation view in a Manager Function**

In the case of managed functions, these O&M interfaces (when required) sometimes are also accessed by the manager function (that performs configurations and monitors the element) and avoid conflicts by the split in the type of actions.




**Platform Operation view in a Managed Function**

Some tools to ease the O&M of the set of functions that constitute the architecture are often provided by the different commercial products present in the architecture, and can be grouped under an umbrella of common O&M tooling, so that they can benefit of supplementary common services (e.g. DNS, LDAP, CA, etc.), common backends (e.g. to collect monitoring info), or even wrap sequences of sparse O&M actions in a GUI wizard tools in order to ease them, either by making them more visual or by guiding the operator to minimize errors (e.g. by narrowing down the available options).

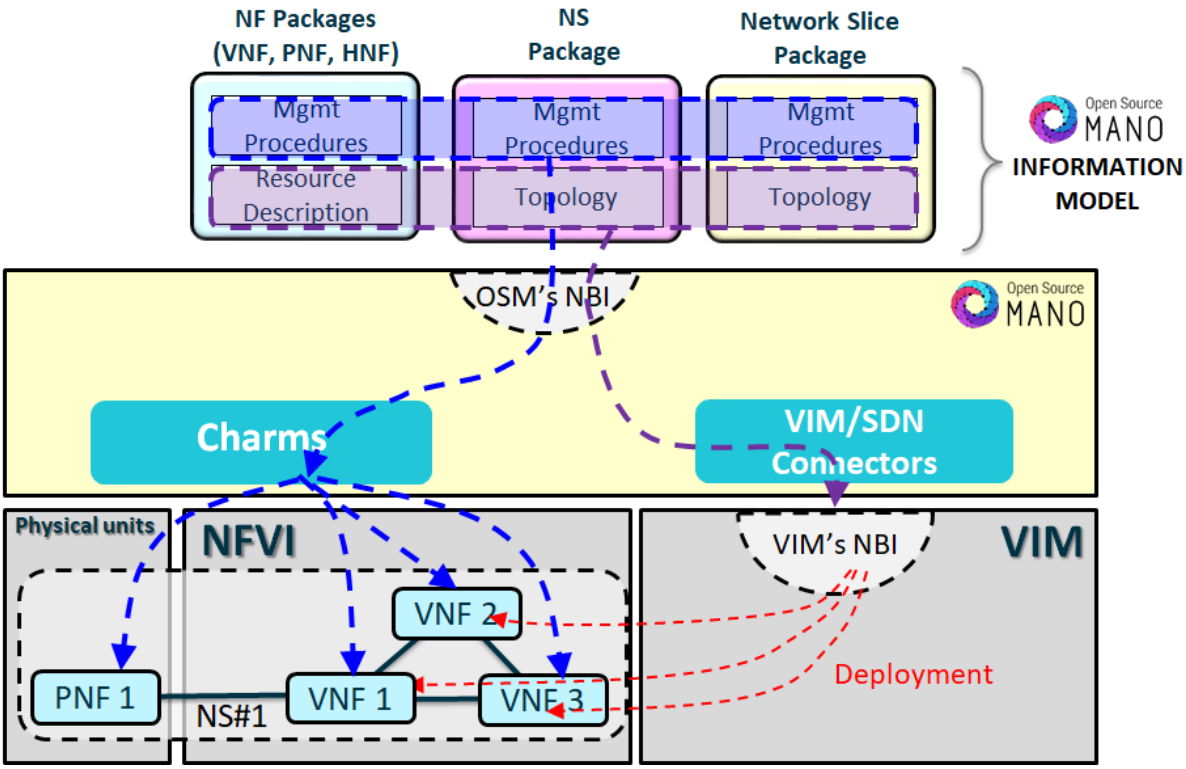
# OSM Scope and Functionality

## OSM Objetives and Scope

The goal of ETSI OSM (Open Source MANO) is the development of a community-driven production-quality **E2E Network Service Orchestrator (E2E NSO)** for telco services, capable of modelling and automating real telco-grade services, with all the intrinsic complexity of production environments. OSM provides a way to accelerate maturation of NFV technologies and standards, enable a broad ecosystem of VNF vendors, and test and validate the joint interaction of the orchestrator with the other components it has to interact with: commercial NFV infrastructures (NFVI+VIM) and Network Functions (either VNFs, PNFs or Hybrid ones).

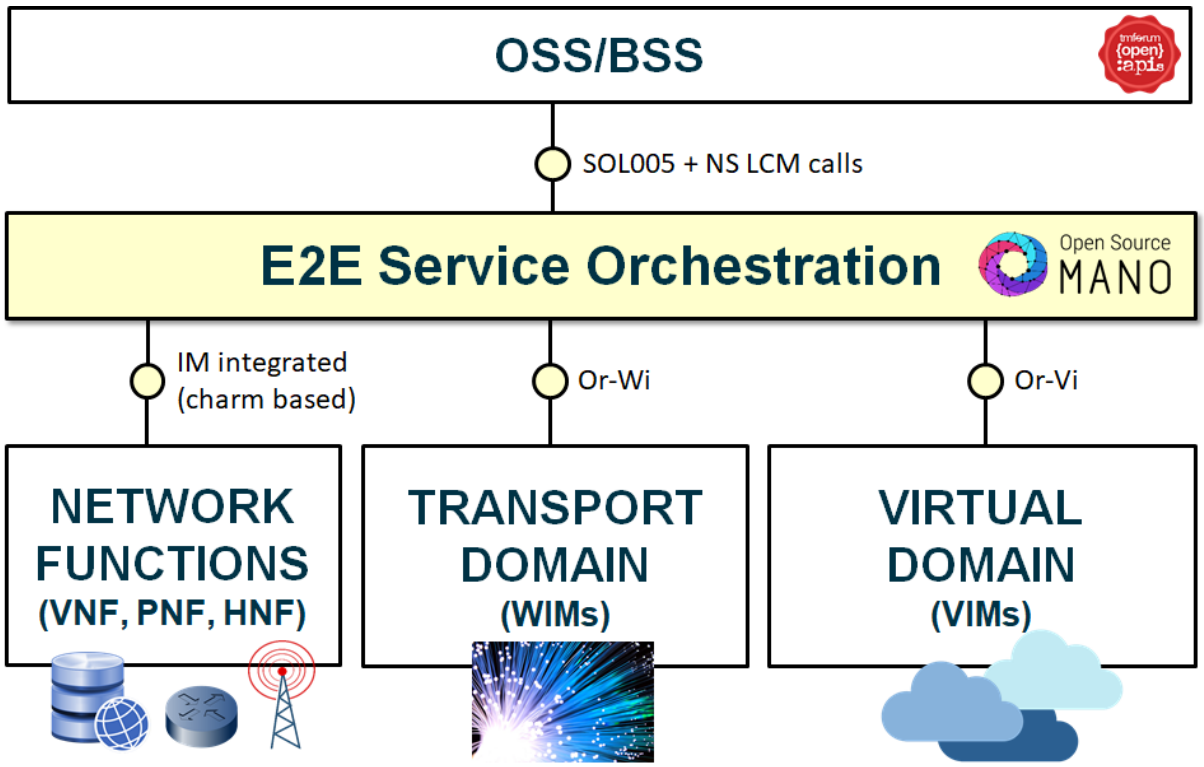
OSM's approach aims to minimize integration efforts thanks to four key aspects:

1. **A well-known Information Model (IM)**, aligned with ETSI NFV, that is capable of modelling and automating the full lifecycle of Network Functions (virtual, physical or hybrid), Network Services (NS), and Network Slices (NSI), from their initial deployment (instantiation, Day-0, and Day-1) to their daily operation and monitoring (Day-2).
   - Actually, OSM's IM is completely infrastructure-agnostic, so that the same model can be used to instantiate a given element (e.g. VNF) in a large variety of VIM types and transport technologies, enabling an ecosystem of VNF models ready for their deployment everywhere.
2. OSM provides a **unified northbound interface (NBI)**, based on NFV SOL005, which enables the full operation of system and the Network Services and Network Slices under its control. In fact, OSM's NBI offers the service of managing the lifecycle of Network Services (NS) and Network Slices Instances (NSI), providing as a service all the necessary abstractions to allow the complete control, operation and supervision of the NS/NSI lifecycle by client systems, avoiding the exposure of unnecessary details of its constituent elements.




**OSM's IM and NS operation via NBI**

3. The **extended concept of "Network Service" in OSM**, so that an NS can span across the different domains identified —virtual, physical and transport—, and therefore control the full lifecycle of an NS interacting with VNFs, PNFs and HNFs in an undistinguishable manner along with on demand transport connections among different sites.




**OSM interaction with different domains**

4. In addition, **OSM can also manage the lifecycle of Network Slices**, assuming if required the role of Slice Manager, extending it also to support an integrated operation.

## Service Platform view

OSM provides the capability of realising one of the main promises derived from NFV and the dynamic capabilities that it brings: creating networks on demand ("Network as a Service" or NaaS) for either their direct exploitation by the service provider or for their potential commercialization to third parties.

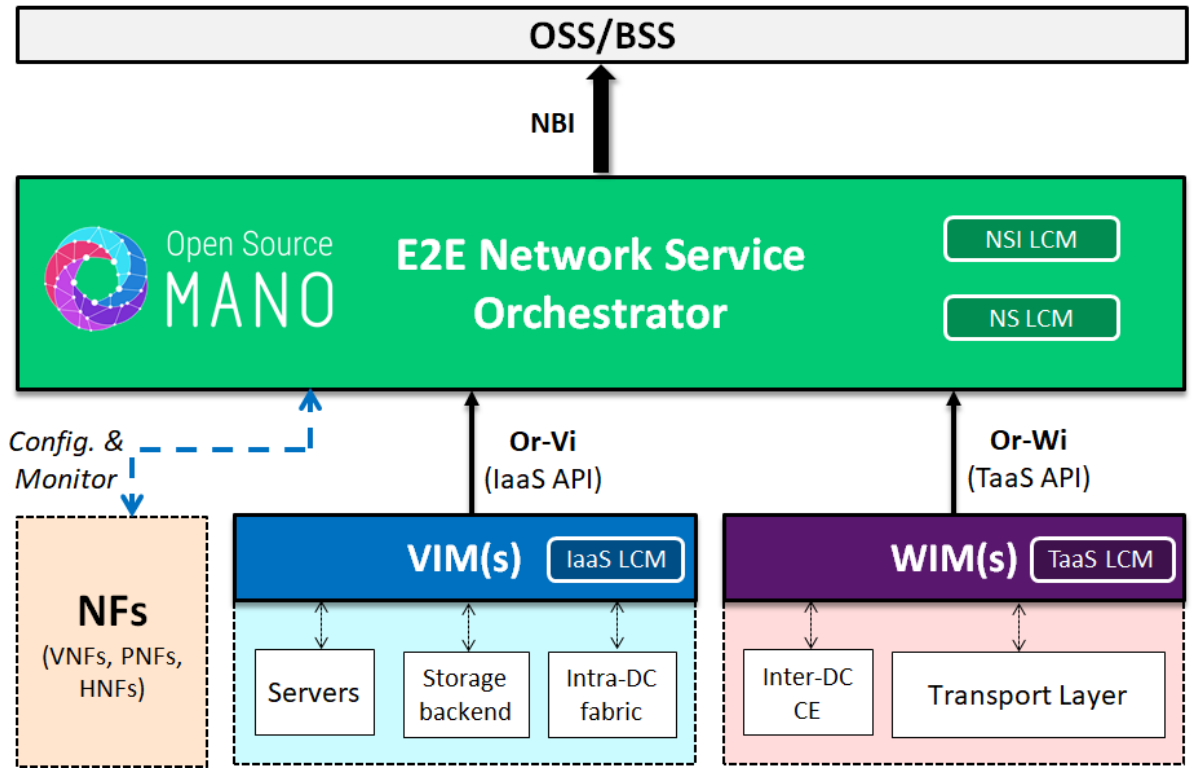
In that sense, OSM works as a **Network Service Orchestrator (NSO)**, *manager function* of a *Network Service Platform* (see ***Service Platform view and Layered Service Architectures*** for details), intended to provide the capability of creating network services on demand and returning a *service object* ID that can be used later as a handler to control the whole lifecycle and operations of the network service via subsequent calls to OSM's northbound API and monitor its global state in a convenient fashion.

In the case of OSM, there are two types of *NaaS service objects* that OSM is able to provide on demand to support the NaaS capability: the **Network Service (NS)** and the **Network Slice Instance (NSI)**, being the latter a composition of several Network Services that can be treated as a single entity (particularities of both types of *NaaS service objects* will be described in the next sections).

OSM, as *manager function* of a *service platform*, consumes services from other *service platforms* and controls a number of *managed functions* in order to create its own composite higher-level *service*

*objects*. Thus, OSM consumes services provided by the platform(s) in charge of the Virtual Infrastructure (to obtain VMs, etc.) and the platform(s) in charge of the SW-Defined Network (to obtain all the required kinds of inter-DC connections), and, once assembled, configures and monitors the constituent network functions (VNFs, PNFs, HNFs) in order to control the LCM of the entire NS/NSI to be offered on demand.

This view of OSM as part of a *service platform architecture* for NFV is summarized in the following picture:




**OSM in Service Platform view**

## Services offered Northbound

### *OSM as provider of Network Services (NS) on demand*

The Network Service (NS) is the minimal building block in OSM to manage networks provided *as a service*, which bundles in one single *service object* a set of interconnected network functions (VNFs, PNFs and HNFs) which can span across different underlying technologies (virtual or physical), locations (e.g. more or less centralized) and geographical areas (e.g. as part of the service of a large multi-national corporate customer).

In order to enable effectively a "service on demand", these newly created Network Services are not provisioned as the result of a handcrafted or ad hoc procedure, but as the outcome of a simple and well-known method based on API invocations (to OSM's NBI) and descriptors following OSM's Information Model. These descriptions should facilitate the creation of Network Services composed of different network appliances (VNFs, PNFs or HNFs) coming from different vendors, so that those

appliances (also called **Network Functions** or **NF**) can come pre-modelled by their provider and the service provider can focus on modelling the Network Service itself.

Once a Network Service is entirely modelled (in a **Network Service Package**), the model works effectively as a template that can be particularized ("parametrized") upon NS creation time to incorporate specific attributes for that NS instance, returning a unique NS instance ID, useful to drive LCM operations at a later stage. OSM puts also in place all the necessary abstractions to allow the complete control, operation and supervision of the NS lifecycle —in a normalized and replicable fashion— by the client system (usually, OSS/BSS platforms). This NS instance "handler" is not required to expose unnecessary details of its constituent elements, in order to minimize the impact over the final service of potential changes in the NFs or the NS topology that did not intend to mean a change on the actual service offer.

In order to achieve the desired level of flexibility and abstraction, OSM augments the concept of NS with respect to ETSI NFV to incorporate physical and transport domains to enable real E2E services that can be extended beyond virtual domains. Thus, it is possible in OSM:

- To combine in a single NS virtual network functions (VNFs), physical network functions (PNFs) and, even, network functions composed of both physical and virtual elements (Hybrid Network Functions, or HNFs), more typical of elements closer to the access network.
- To deploy such NS across a distributed network and even create inter-site transport connections on demand, leveraging on the APIs of SW-Defined Network Platforms.

Both **OSS and BSS platforms are expected to be consumers of the NS created on demand by OSM**, and sometimes may even keep the control of some constituent network functions of the NS if required (this is quite useful to reuse legacy network nodes without requiring major changes in the OSS). For that reason, OSM has also the capability to delegate selectively the control of specific constituent NFs of the NS to the OSS/BSS platform if explicitly specified in the NS model , giving full freedom to support legacy or hybrid scenarios as desired.

### Lifecycle and operation of a Network Service

In the following sections, the stages related to the lifecycle and operation of the NS in the E2E Network Orchestrator are thoroughly discussed and described, so that the API capabilities (and the companion IM) can be better understood in a context of operation:

0. Preparation phase: Modelling
1. Onboarding
2. NS creation (day-0 and day-1)
3. NS operation (day-2)
4. NS finalization

It must be noted that, although the initial phase of modelling is a mere pre-requirement, prior to any actual existence of the NS itself (and with no API interactions involved), it is required to understand the NS lifecycle and the API calls that are available at later phases.

### Phase #0: Modelling

OSM's IM provides mechanisms to include the complete blueprint of the NS behaviour, including both a full description of the NS topology, the lifecycle operations that are enabled, and the NS primitives that are available, along with their automation code. Since Network Services are composed, by definition, of one or several Network Functions (VNFs, PNFs or HNFs) of heterogeneous types and internal behaviours —and likely to come from different providers—, the IM provides a mean to let the provider describe the internal topology, required resources, procedures and lifecycle of the Network Functions. This information come bundled in the so-called NF Packages.

This two-layered modelling approach has several advantages:

- Prevents that the designer of the NS Package (i.e. a Service Provider such as Telefónica) is directly exposed to NF internals, and can focus on the composition of the NS itself, based exclusively on the external properties and procedures of the NF.
- Enables the consistent and replicable validation of the NFs and their companion NF Packages across all the supply chain, so that the NF vendor can guarantee that their elements are always used and operated in the appropriate manner.
- Obviously, the same NF Package can be used in more than one NS with no additional modelling work at NF level.

### Phase #1: Onboarding

Once the models are ready, they can be injected to the system, so that they can be used as templates for NS creation later on, in a process that is known as **onboarding**.

OSM's NBI offers API calls to support CRUD (Create, Read, Update, Delete) operations over the corresponding NS and NF Packages, in order to support the two-layered modelling approach previously described (that can become three-layered in the case of Network Slices), the API supports specific CRUD operations to handle the corresponding NS and NF Packages (and NST when applicable) as independent but related objects. In these operations, and particularly in the onboarding step, the necessary checks to validate in-model and cross-model consistency are performed.

### Phase #2: NS creation (day-0 and day-1)

Once the corresponding NS and NF Packages are successfully onboarded in OSM, there is all that is needed to use them as templates for the actual NS creation. Accordingly, OSM offers API calls to support CRUD (Create, Read, Update, Delete) operations related to NS instances.

In the case of the **NS creation** operation (also known as **NS instantiation**), OSM takes as input an NS Package and, optionally, a set of additional deployment constraints (e.g. target deployment locations for specific VNFs of the NS) and parameters to particularize in the NS, as explicitly allowed by the NS Package.

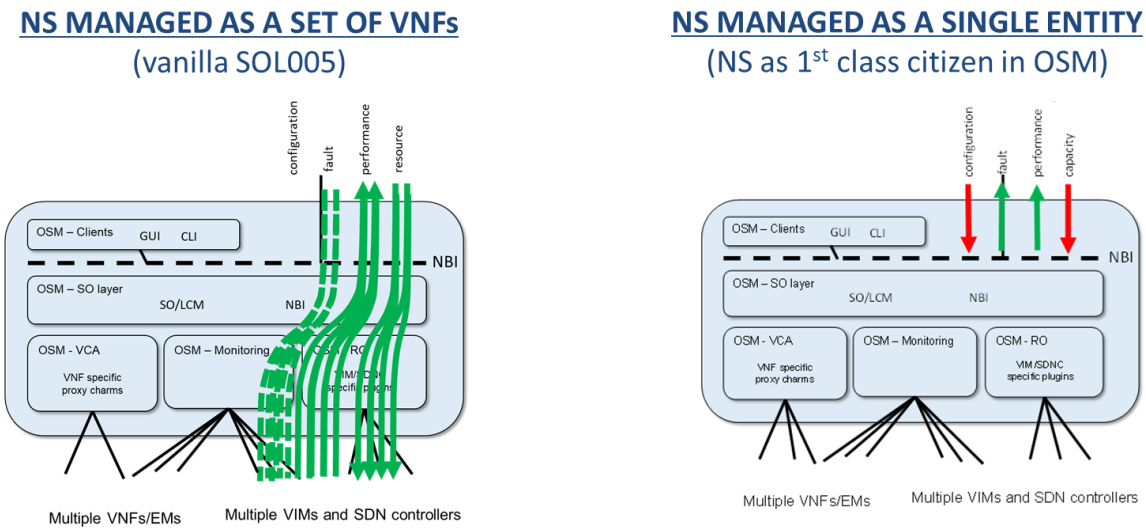
During the NS creation, OSM interacts with different service platforms southbound (VIMs and WIMs) and managed functions (NFs) to create the composite *service object* of the NS instance.

### Phase #3: NS operation (day-2)

Once the NS has been successfully created, the NS instance becomes the only relevant object for further operation, lifecycle and assurance actions.

Once an NS has been successfully created, the NS instance becomes the only relevant object for further operation, lifecycle and assurance actions. The NS/NSI instance can be subject to different types of API-driven operations, which fall into one of these categories:

- **Common Lifecycle operations.** There must be a number of API calls that allow to trigger well-known standard actions potentially applicable to any NS, such as scaling actions, pausing/resuming, on-demand monitoring requests, SW upgrades, etc.
- **Actions derived from NS primitives.** Besides operations potentially applicable to any NS/NSI, each NS/NSI can have a set of operations that are relevant only for the specific functionality that the NS/NSI offers, such as the addition of new subscribers, changes in internal routing, etc. Those actions are enumerated and codified in the corresponding NS Package (that leverage, in turn, on the atomic actions available in NF Packages) are exposed by the API as primary actions available in that given NS/NSI.




**NS managed as a single entity via NS primitives**

- Although not directly requested by the client system via API, it must be noted that other actions can be internally triggered in the NS as a result of a closed-loop policies defined in the NS or NF Packages. Usually, these actions involve the monitoring of some parameter of the NS or the NF and the triggering of one of the aforementioned actions if a given threshold is reached (e.g. automatic scale-out).

In fact, it is possible in OSM to work with metrics and alarms with great flexibility:

- Descriptor-defined alarms and metrics related to **VDU-PDU level**, but still explicitly co-related with NS/NF instances.
- Descriptor-defined alarms and metrics related to **application-specific** (NF or NS) KPIs.

- It also allows on-demand requests to export alarms, events and metrics via Kafka bus, and a smooth integration with the most popular frameworks, including ELK, Prometheus, and Grafana.

It must be noted that OSM also allows the management of **brownfield scenarios** where some elements had to be managed out-of-band by an external/legacy entity.

#### Phase #4: NS finalization

As any other on-demand service, it is possible to finalize a NS and release the resources that had been assigned, preserving those components that should not be removed (e.g. persistent volumes).

The "delete" call of the API (from the of the aforementioned CRUD operations related to a NS) is in charge of triggering that process and report on demand of its status of completion.

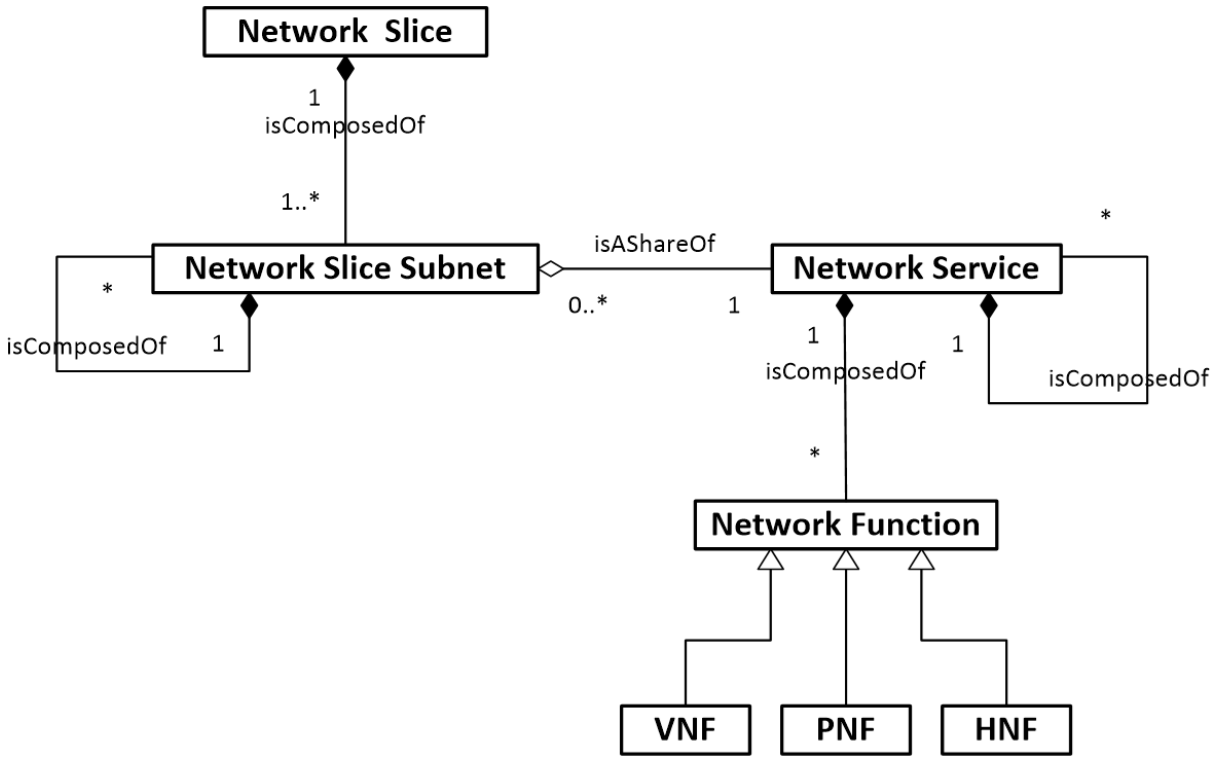
### *OSM as provider of Network Slices*

#### Network Slices and Network Services

ETSI OSM is also capable to to provide Network Slices *as a service*, assuming also the role of Slice Manager as per ETSI NFV EVE012 and 3GPP TR 28.801, extending it also to support an integrated operation of **Network Slice Instances (NSI)** along with Network Service instances (NS).

The intended use of a Network Slice can be described as a particularization of the NaaS case but more focused on the enablement of 5G use cases. This 3GPP spec defines a specific type of underlying construct, the Network Slice, which is intended to provide the illusion of separated specialized networks for different purposes. Unsurprisingly, Network Slices, in practice, operate as a particular kind of Network Service or, more generally, as a set of various Network Services that are treated as a single entity.

3GPP defines the relation between Network Slices and vanilla NS as per ETSI NFV in a very specific manner, where Network Services become the so-called *NS Subnets* of the Network Slice, while the Network Slice with its constituent *NS Subnets* can be deployed and operated as if they were a single entity. The following picture depicts the intended relation between both concepts:




**Relation between Network Slices and vanilla ETSI NFV Network Services**

Still consistent with the same modelling of a regular NS, these *NS Subnets* can be either exclusive to an upper-level Network Slice (*Dedicated NS Subnets*) or shared between several Slices (*Shared NS Subnets*), such as in the case of the RAN. Likewise, they would have their own lifecycle and operations as any other NS, so no disruption in the modelling is created.

The Network Slice therefore has two key characteristics compared to a Network Service

- The Network Slice is composed from a number of Network Subnets
- A Network Slice Subnet is a share of a Network Service.

First, as it can be seen, the Network Slice concept defined in 3GPP overlaps almost completely with the concept of "Nested NS" (an NS composed of various NS) as defined in ETSI NFV, with the only addition of including some PNFs and Transport connections explicitly, features that are already included in the extended concept of NS that OSM already provides. Therefore, the decision of incorporating the Network Slice as a particular case of NS in OSM was rather natural.

Second, we can see that a single *NS Subnet* instance is, by definition, a share of a single NS instance. However, as both *NS Subnets* and NS can be recursively composed (that is a *NS Subnet* can be composed of *NS Subnets* and an NS can be composed of NSs) there is still full flexibility in the model. This sharing relationship is the key new feature which slicing introduces and has an important consequence for orchestration. This consequence is represented in the UML figure above by the fact that the 'isAShareOf' relationship is an aggregation (open diamond) and not a composition (solid diamond).

Most significantly, the *NS Subnet* cannot control the existence of the NS of which it is a share. As the NS (for example a RAN NS) may be providing shares to other *NS Subnets*, when a network slice instance which is sharing this NS is no longer needed and deleted, the composed *NS Subnets* may be deleted, but the NS itself must not be deleted as it may well be still providing shares to other network slices.

This means that the lifecycles of network slices with their constituent *NS Subnets* can be managed as a composition hierarchy in same way the lifecycles of NSs and NFs can be managed as a composition hierarchy, the two composition hierarchies must not be mixed together.

### Lifecycle and operations of a Network Slice

As previously described, Network Slices operate as a grouping of a set of potentially shared Network Services, which would become the so-called *NS Subnets* of the Network Slice. The Network Slice with its constituent *NS Subnets* can be deployed and operated as if they were a single entity.

Alike Network Services, 3GPP TR 28.801 describes the lifecycle of the global Network Slice, which is comprised of the four following phases:

5. **Preparation.** In the *Preparation* phase, the Network Slice (or Network Slice Instance, NSI) does not exist yet. The preparation phase includes the creation and verification of Network Slice Templates (NST), the onboarding of these, preparing the necessary network environment to be used to support the lifecycle of NSIs, and any other preparations that are needed in the network.

6. **Instantiation, Configuration and Activation.** During *Instantiation/Configuration*, all resources shared/dedicated to the NSI have been created and are configured to a state where the NSI is ready for operation. The *Activation* step includes any actions that make the NSI active (if dedicated to the network slice, otherwise this takes place in the preparation phase). Network slice instantiation, configuration and activation can include instantiation, configuration and activation of other shared and/or non-shared network function(s).

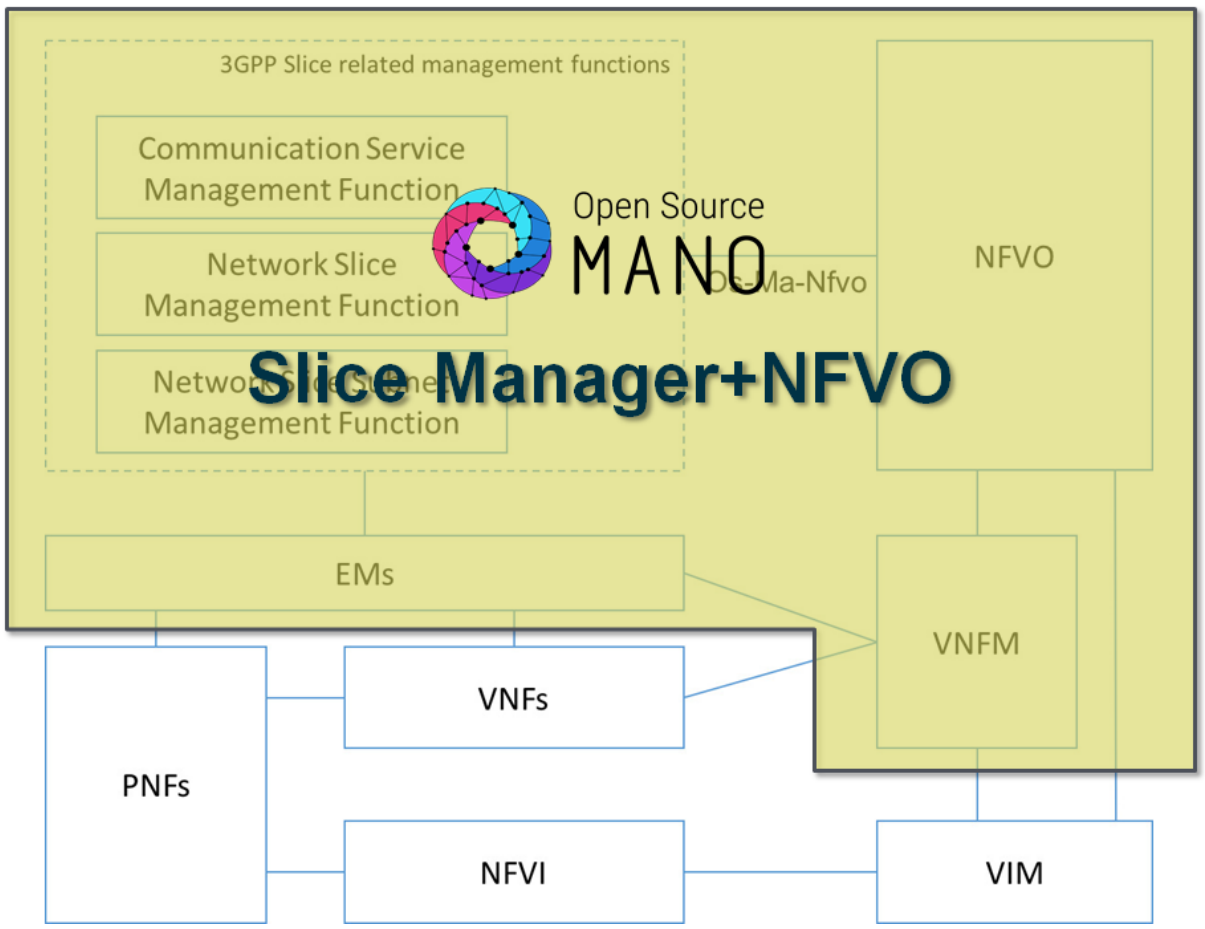
7. **Run-time.** In the *Run-time* phase, the NSI is capable of handling traffic to support communication services. The Run-time phase includes supervision/reporting (e.g. for KPI monitoring), as well as activities related to modification: upgrade, reconfiguration, NSI scaling, changes of NSI capacity, etc.

8. **Decommissioning.** The *Decommissioning* phase includes deactivation (taking the NSI out of active duty) as well as the reclamation of dedicated resources (e.g. termination or re-use of network functions) and configuration of shared/dependent resources. After decommissioning the NSI does not exist anymore

Two non-mutually exclusive modes of deployment and management to support this lifecycle are feasible in OSM: **Full E2E Management** (Integrated Modelling) and **Standalone Management** (Vanilla NFV/3GPP).

### Full E2E Management (Integrated Modelling)

In this mode of operation, the Network Slice can be treated as kind of meta-Network Service, and can be modelled as per the augmented NS lifecycle model described previously, so that OSM works also as Slice Manager (Slice-M). For convenience, the NSI becomes as a first-class object of OSM.




**Full E2E Management of Network Slices**

In this mode, there is a natural match between the different phases of the lifecycle, where the Network Slice Template (NST) and the Network Service Instance (NSI) play, respectively, the same roles as the NS Package (the template defining a NS) and the NS instance in the general lifecycle of an NS in OSM:

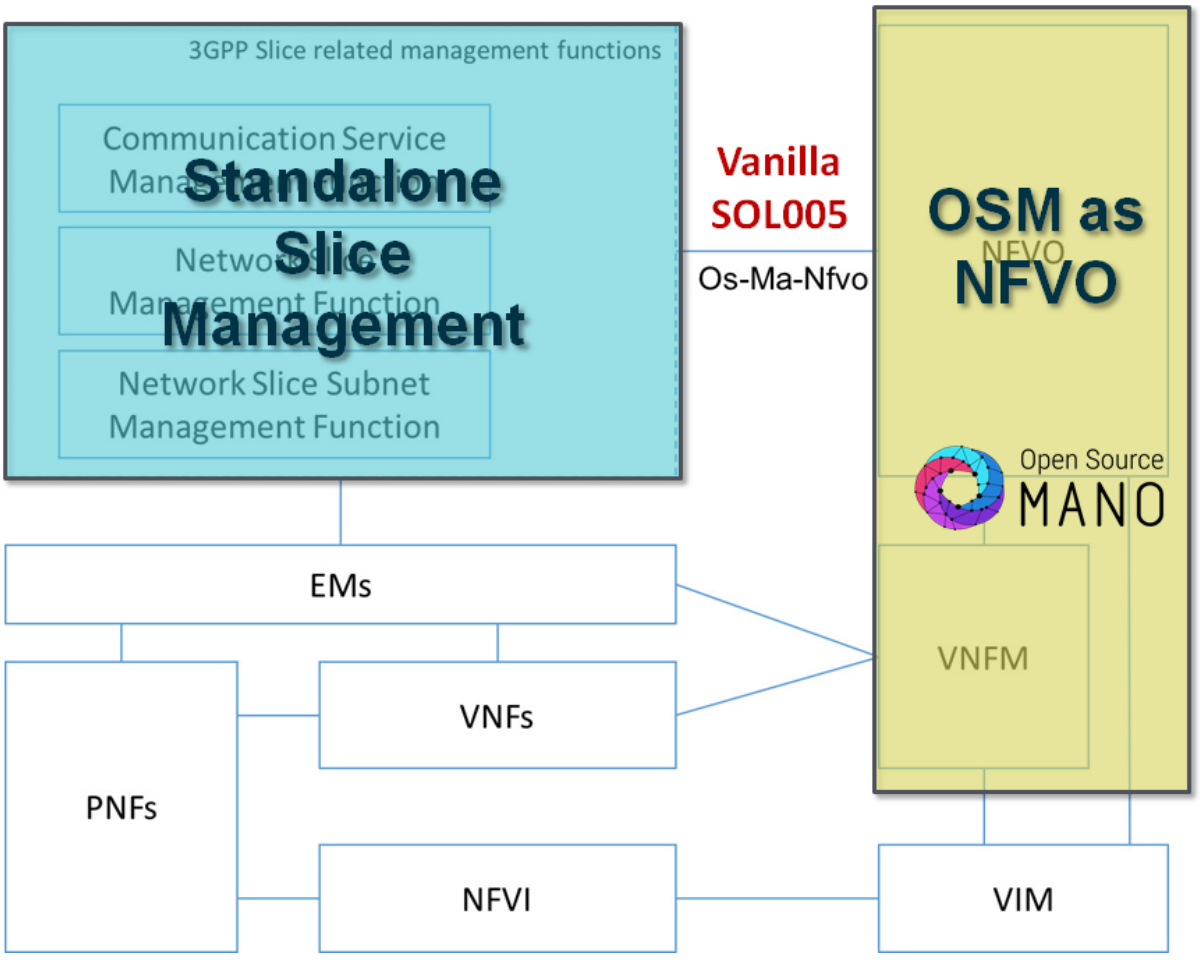
- **Preparation** is comprised of ***Phase #0 (Modelling)*** and ***Phase #1 (Onboarding)***.
- **Instantiation, Configuration and Activation** is equivalent to ***Phase #2 (NS Creation)***.
- **Run-time** provides a standardized subset of the operations available at ***Phase #3 (NS Operation)***.
- **Decommissioning** is equivalent to ***Phase #4 (NS finalization)***.

There are some obvious advantages of this approach:

- The Preparation phase is largely simplified, as there is no split in the information models between the 3GPP Slice Manager and the NFV Orchestrator (a "reduced" NSO specialized in virtual components, as defined by ETSI NFV).
- Day-2 operations are integrated in a single platform and a single northbound interface.
- Possibility to add custom primitives to a given slice, alike the general NS constructs allow.
- Packages are generated, by definition, with a multi-vendor scenario in mind.
- The slices can include non-3GPP related network functions with no need of special integration.

### Standalone Management (Vanilla NFV/3GPP)

Alike the case of legacy OSS/BSS described in the NS operations, it is also possible to allow an external standalone system to manage the lifecycle of slices (as standalone Slice Manager) and leverage on OSM simply as if it were a vanilla NFV Orchestrator (NFVO), using the regular (non-augmented) SOL005 interface.




**Standalone (Vanilla) Management of Network Slices**

While this scenario is way less convenient in terms of operation than the one of integrated management, it may be useful for small or vertical deployments of slicing and can also be supported as fallback option.

In this mode of operation, the management of the slices happens entirely outside of OSM in a separate management element, which would only leverage on vanilla NFVO capabilities of OSM. On the other hand, from the perspective of OSM, the standalone slice management would look like just any other OSS.

In consequence, the lifecycle operations of the Slice may require all the additional preparatory and intermediate steps to guarantee an appropriate slice-NS mapping as defined by 3GPP:

- In this case, during the preparation phase, the resource requirement for an NST is realized by one or more existing Network Service Descriptors that have been previously on-boarded on the E2E Orchestrator (working as NFVO). The creation of a new NST can lead to requiring the update of an existing NSD or generation of a new NSD followed by on-boarding the new NSD if the slice requirements do not map to an already on-boarded NSD (i.e. available in the NSD catalogue). Indeed, the Network Slice for the multiple Network Slice Instances may be instantiated with the same NSD, in order to deliver exactly the same optimizations and features but dedicated to different enterprise customers. On the other hand, a network slice intended to support totally new customer facing services is likely to require a new NS and thus the generation of a new NSD.
- The network slice instantiation step in the second phase needs to trigger the instantiation of the underlying NSs. Vanilla NFV-MANO functions would only be involved in the network slice configuration phase if the configuration of virtualisation-related parameters is required on one or more of the constituent VNF instances.

#### *OSM's IM and NBI specifications*

OSM provides a well-known, complete and thoroughly tested **Information Model** to facilitate an accurate and sufficient description of the internal topology, procedures and lifecycle of Network Services and Network Slices. ETSI OSM's IM is openly (and freely) available for every industry player, continuously evolved by a large Community of industrial players, and being pre-validated in its intended E2E behaviour at the own OSM upstream community, so that new cloud and application technologies can be taken into account as they emerge and mature. The latest official version of ETSI OSM's IM is always available as an up-to-date spec at OSM's documentation and git repos.

On the other hand, **OSM's NBI** provides a superset of ETSI NFV SOL005 API calls with the addition of E2E NS operation capabilities and the ability to handle Network Slices. Alike the IM, the latest official version of OSM's NBI is openly available in OpenAPI format, and can be used as the authoritative reference for interoperability northbound, even facilitating the automated generation of code for client applications.

### Services consumed Southbound

OSM is oriented to consume the services of two kinds service platforms commonly available in industry:

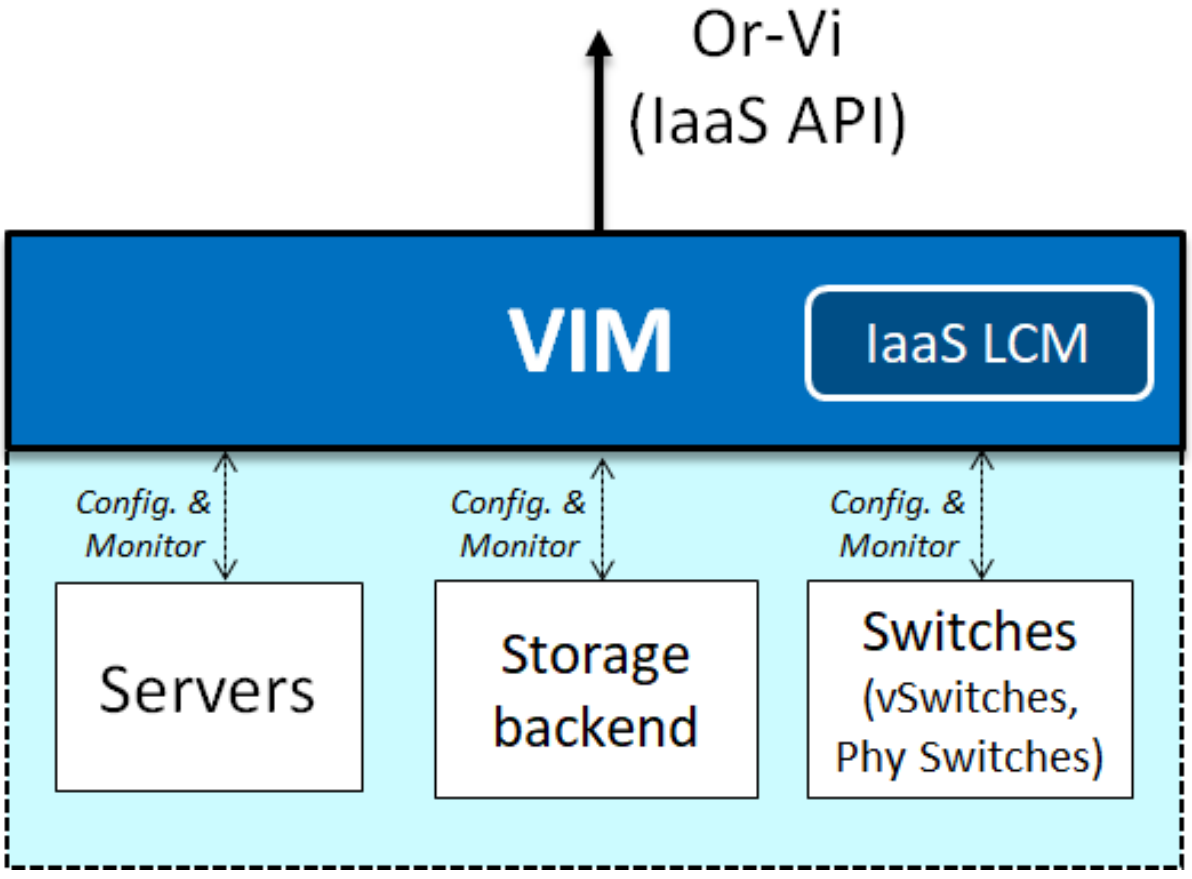
- **Virtual Infrastructure Platforms**, each managed by a ***Virtual Infrastructure Manager (VIM)***, which exposes an *Or-Vi* reference point northbound.
- **Software-Defined Network Platforms**, each managed by a ***WAN Infrastructure Manager (WIM)*** (often a kind of ***SDN Controller***), which exposes an *Or-Wi* reference point northbound.

In order to support the variety of alternative industry APIs that implement these reference points, OSM has plugin models for both VIMs and WIMs, so that all the variety of commercial southbound APIs are supported via their corresponding connectors.

### *VIMs as managers of Virtual Infrastructure Platforms*

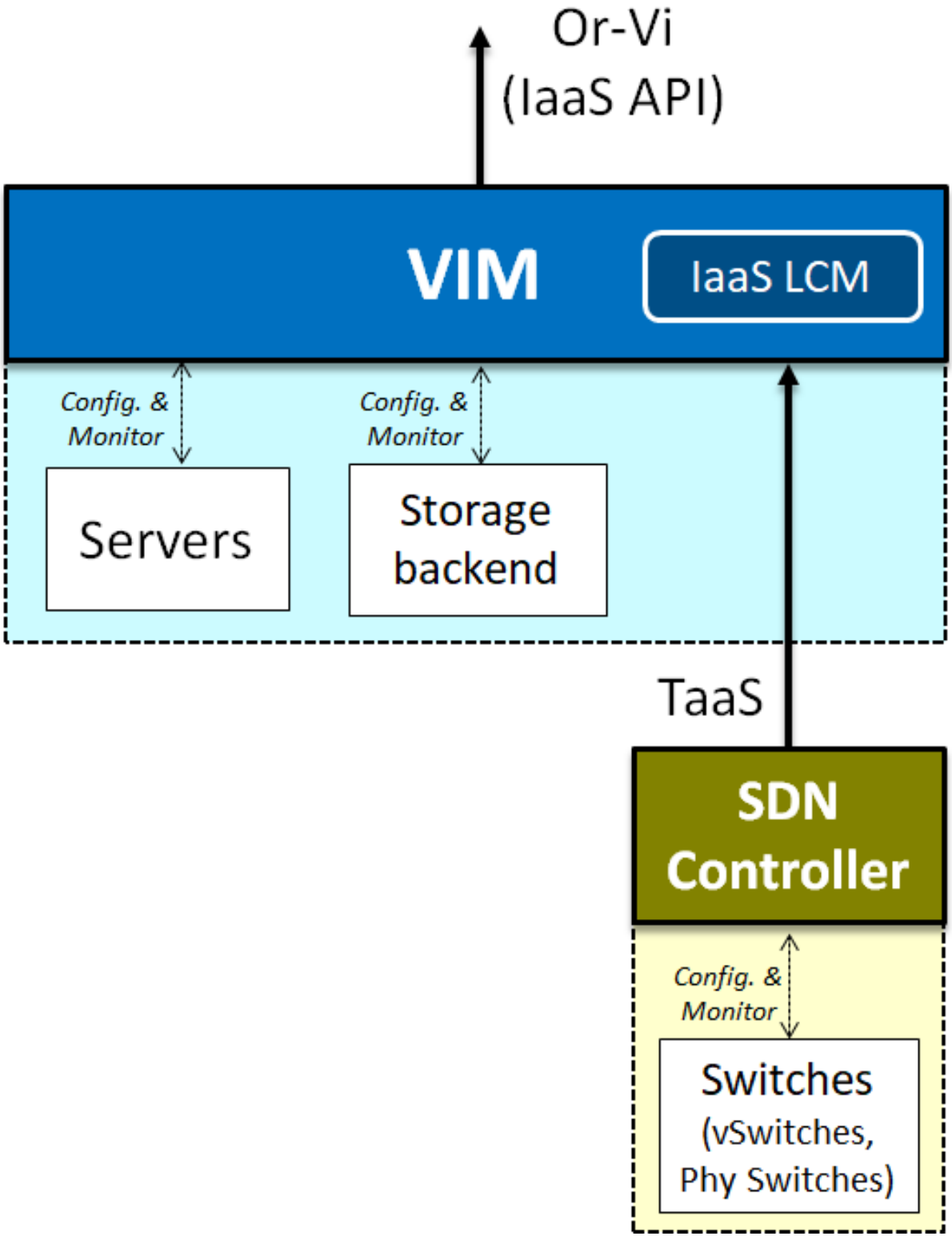
The functionality of providing chunks of the underlying physical resources dynamically is known in industry as "**Infrastructure as a Service**" or **IaaS**, and it is the most basic building block that a cloud can offer on demand. In our *service platform view*, the pool of resources that offers IaaS becomes a **Virtual Infrastructure Platform**.

The manager function of this platform is the **Virtual Infrastructure Manager** (**VIM**), and is one of the most popular platform managers in exploitation in industry. In order to perform this task, the VIM is in charge of controlling a set of compute, memory, storage and network resources, and returning them sliced as VMs. Thus, the VIM is in charge of controlling pools of compute nodes (i.e. servers, including their hypervisor SW), virtual networking (vSwitches and, sometimes, physical switches), and storage backends.



### VIM as manager of Virtual Infrastructure Platform

In some advanced cases, the VIM can be deployed in conjunction of an SDN Platform, which provides connectivity on demand adapted to the needs of the VIM:




**VIM consuming services from an SDN Platform**

It must be noted that in both configuration **the service exposed by the VIM is undistinguishable from the perspective of the northbound API, and hence to OSM**.

Due to the variety of VIM APIs available and the requirement to be open to future types of technologies, OSM provides a plugin model for the calls needed for the IaaS services required by OSM to instantiate and manage a NS/NSI. As of today, ETSI OSM already supports out-of-the-box (i.e. with no need of integration):

- Openstack-based VIMs (e.g. Canonical OpenStack, Red Hat OpenStack Platform, WindRiver Titanium Cloud (4 and above), WhiteCloud, SuSE OpenStack, Mirantis OpenStack, ECEE, FusionSphere, etc.)
- VMware VIO 4 and above.
- VMware vCloud Director
- Amazon Web Services (AWS)
- OpenVIM

### SDN Assist

Most VIMs provide the automatic creation of network connectivity for management and signaling interfaces but not for those that are dataplane intensive (use of PF passthrough or SR-IOV). In those cases, the VIM is able to create virtual resources with *Enhanced Platform Awareness (EPA)* requirements but cannot take care of providing the required underlay (physical) connectivity between them.

In those cases, where the VIM does not support natively the management of underlay networking, OSM is able to supply the missing functionality of handling the underlay connectivity with the help of an SDN controller, which manages a fabric where the compute nodes of the VIM are connected. This unique functionality of OSM, which is called **SDN Assist**, enables OSM to:

- Provide the dataplane connectivity that the VIM is unable to manage.
- Treat the VIM+SDN Assist combo as if they there were a single *augmented* VIM, so that, from user's perspective **they will behave like a regular unique manager function** of a given Virtual Infrastructure Platform.

In order to work properly, it is a pre-requirement to have a clear delineation between the *knowledge* and *responsibility* of the VIM and the SDN controller:

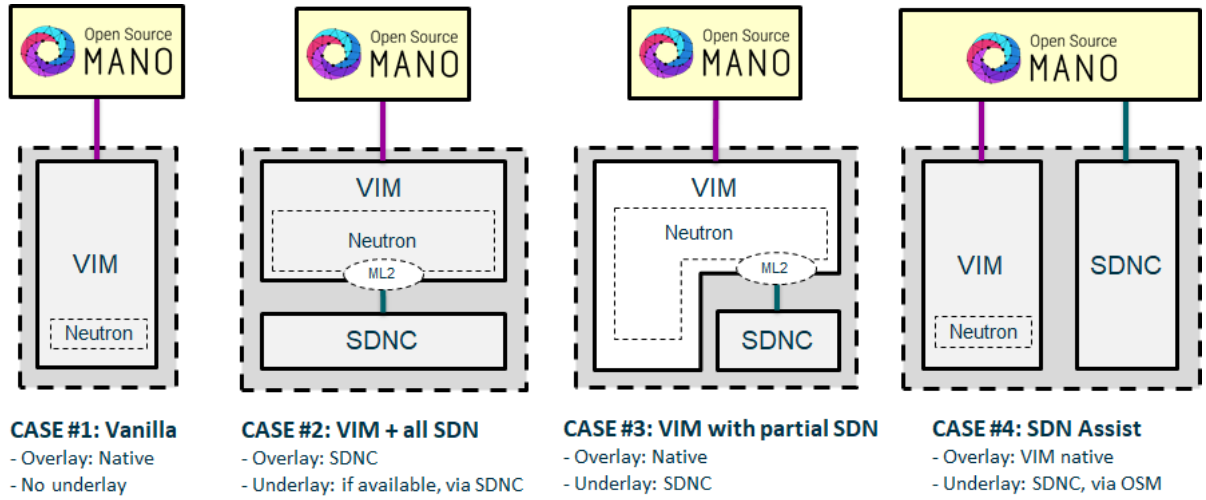
- The **VIM** will be in charge of deploying the VMs and the overlay networks, and providing to OSM the information about the compute nodes and interfaces assigned to the VMs.
- The **SDN controller** will be responsible of creating the underlay connectivity taking as boundary conditions the switches and ports to be connected to the same network. The internal switching topology of the datacentre will be known by the SDN controller, fed as part of the provisioning activities (i.e. prior to any instantiation process).

In that scenario, OSM keeps the mapping between compute nodes and interfaces at VIM level and the switch ports at SDN controller level.

Due to the variety of SDN controllers and the requirement to be open to future types of technologies, OSM provides a plugin model for the calls needed for SDN Assist, namely, the ones required to provide the intra-VIM underlay part (i.e. to *assist* effectively). As of today, OSM provides SDN Assist plugins for for the following families of SDN Controllers:

- OpenDayLight (ODL)
- ONOS
- FloodLight

The following image depicts all the possible Or-Vi schemes, including SDN Assist:




**Or-Vi schemes supported by OSM**

## OpenVIM

OSM also provides a reference VIM with EPA capabilities and underlay support, called **OpenVIM**, which is shipped as part of OSM's releases since Release ONE.

As a reference VIM, **OpenVIM** is particularly useful in the context of deployment that require high I/O performance and efficiency (leveraging advanced EPA capabilities), deployments in the Edge, where the footprint requirements for a VIM can be very low, and setups where software in the compute nodes cannot be upgraded as often as the software of the control of the VIM.

## Metrics and alarms

In the different VIMs, OSM supports the interaction with different existing frameworks to gather metrics and alarms for different Virtual Domain technologies:

- AODH/Gnocchi for OpenStack
- VMware vRealise Ops Update
- AWS CloudWatch

Following the same approach as in other cases of API diversity, OSM uses a plugin model for VIM metric frameworks, which normalizes the types of metrics a alarms for OSM, and which is easily extensible to support additional frameworks and technologies.

### *SDN Controllers/WIMs as managers of Software-Defined Network Platforms*

Another type of *manager functions* widely available are the **SDN Controllers (SDNC)** and their specialized version for transport connections, the **Controllers for Software-Defined Transport Network (SDTN)**. Whenever these elements are invocated from an NSO such as OSM (as a superset of an NFVO), can be also referred as **WIMs (WAN Infrastructure Managers)**. In our *service platform view*, the pool of connectivity resources that is offered here on demand becomes a **Software-Defined Network Platform**.

Similarly to the role of the manager in other platforms, the key function of these WIMs is providing connections on demand and offering an API to manage their lifecycle and monitor them consistently. One of the key advantages of this approach is that WIM's API aims to be largely independent of the specific underlying elements, the network topology underneath and/or the switching technology itself, so the use of these connections on-demand becomes highly convenient for the client platform and leaves a lot of freedom to design and evolve the physical network underneath.

In order to provide that service, the WIMs, as managers of the platform, are in charge of controlling the switching/routing elements underneath and/or invoking other SDN Controllers in lower levels of a hierarchy. Quite often, these switching elements are designed specifically to support SDN operations with some well-known protocols (e.g. OpenFlow, OVSDB, TAPI…), although some traditional means, such as Netconf/YANG, are commonly used as well.

Thus, via SDN/SDTN Controllers it is possible setting up many different types of connections, involving different types of technologies:

- Virtual networks for a VIM
- MPLS connections
- VPN connections (overlay or with interaction with physical equipment)
- Inter-DC connections (various types)
- MAN connections
- Etc.

Due to the variety of WIM APIs available and the requirement to be open to future types of technologies, OSM provides a plugin model for the calls needed for the inter-VIM connections required by OSM to instantiate and manage a NS/NSI. As of today, OSM provides plugins to support:

- TAPI
- ONOS
- Dynpac

Following the same approach as in other cases of API diversity, this plugin model is easily extensible to support additional WIM APIs and technologies.

*Configuration and monitoring of Network Functions*

Regarding **Network Functions** (VNFs, PNFs, HNFs), OSM incorporates them in a manner that provides model-driven interaction with NFs through the use of Juju Charms, which allows NF vendors to encapsulate their configuration mechanisms (NETCONF+YANG, Expect, SSH+scripts, Ansible, etc.). This makes PNF (and HNF) management indistinguishable from VNF management in OSM.

Two different kinds of Juju charms are supported:

- **Native charms**, when NFs are able to run charms inside. This is particularly interesting for new VNFs or cloud-like VNFs/Apps that already support charms. Interaction with those charms happens directly from the orchestrator.
- **Proxy charms**, when NFs do not support running charms inside, which is always true for PNF. In that case, the proxy charm will use the appropriate configuration protocol to interact with the NF and run the desired actions for the primitive.

# Platform Operation view: management of OSM as Manager Platform

## Interaction with Common Services for platform operation

### Authentication

Following the recommendation of the standards (GS NFV-SOL005 – V2.4.1 Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point), the APIs offered by OSM can only be accessed by authorized consumers. Therefore, OSM has the ability to authenticate users against a user database and to authorize them to do a set of operations in the context of a project.

Authentication/authorization schemes in OSM can be standalone or integrated with external common services (company's user databases and systems). Specifically, in the case of an integrated/brownfield deployment with other components, OSM can interact with authentication/authorization servers using the protocols/mechanisms exposed by them; this interaction could vary, depending on the capabilities offered by the external systems from a simple user authentication to a user and project-based authentication and authorization.

It must be noted that the workflow for user and project management might differ depending on the case. In a standalone/greenfield case, users and projects can be created and managed directly via API. In the integrated/brownfield case, the management of users and projects might be stored in the organization's user databases and systems.

### Logging

OSM can export its logs to external systems using the protocols/mechanisms exposed by them. Logs can include:

- System information, relevant to know the status of the servers hosting OSM and the relevant processes
- Operational information. Each API call should be properly logged by the system

### Data exportation

OSM provides the appropriate mechanisms to expose metrics and relevant data to be consumed by external systems (e.g. data lakes). The simplest mechanism for this purpose is a messaging bus where multiple consumers can be connected, read the information from the bus and export it to any external system.

The minimal information expected to reach the previous bus would be:

- NF metrics (either coming from the NFVI/VIM or from the VNF itself)
- NF and NS alarms (either coming from the NFVI/VIM or from the VNF itself)

Nothing prevents that API calls and LCM events might also reach that bus and feed external systems.

## Management of OSM

### Authentication and Authorization Management

From the perspective of the real operation, OSM allows the partitioning of OSM resources into different **projects** (also known as **tenants**), providing a separate space for:

- A given set of accessible NF packages
- A given set of accessible NS packages
- A given set of accessible NST
- The set of running NS
- The set of running Network Slices (NSI)
- Authorized VIM accounts.
- Authorized WIM accounts.

Some of the previous resources (e.g. NF packages, NS packages and Network Slices) might be declared public (available for all projects) or available for a list of them. However, instances of Network Servers and Network Slices will only belong to one project.

In addition, not all operations are expected to be allowed to all users. OSM allows the definition of **roles**, with the set of privileges or access rights allowed from the whole set of operations:

- CRUD operations over a NF package
- CRUD operations over a NS package
- CRUD operations over NST – see next section
- CRUD operations over Network Services (instances)

- CRUD operations over Network Slices (instances) – see next section
- Advanced LCM of Network Services (e.g. scale out/in)
- Day-2 operations over Network Services
- Day-2 operations over Network Slices
- CRUD operations over VIM accounts
- CRUD operations over WIM accounts
- System operations (user, project and role mgmt. in a standalone/greenfield configuration)

**Users** of the system belong at least to one project, and each user might have potentially different roles depending on the project. This is what is commonly known as a **role-based access control (RBAC)** on a per-project and per-user basis, i.e. for each project a user belongs to, the user will have a role with specific access rights. It must be noted that RBAC mechanisms typically involve the definition of admin/root roles (with all privileges) and admin users, authorized in all projects with the admin/root role.

The way to enforce that the actual privileges are preserved is typically through the NBI. For REST APIs, this is typically achieved through token-based mechanisms, which means that:

- The client platforms of OSM are authenticated and authorized on a project basis, and get a token based on the role in the project. That token identifies the set privileges for subsequent operations.
- Client platforms will make API calls using that token. Based on the authorization rights identified by the token, some operations will be allowed and others forbidden.

*Bootstrap and addition of boundary conditions*

The creation of Network Services and Network Slice Instances requires OSM to be previously fed with enough information to deploy and operate them. The life cycle management of some of these inputs (NF Packages, NS Packages and NS Templates) is fully managed by OSM. However, there are some inputs whose lifecycle is managed by other service platforms:

- **VIM accounts**. In order to be able to deploy in a specific datacenter managed by a VIM, OSM needs to know and store the access information and credentials of the VIM account to be used. Those credentials are created by a VIM administrator, and should grant OSM sufficient rights to perform CRUD operations on flavors, networks, images, volumes and instances. Those VIM accounts need to be added to the NSO, previously to any deployment in that datacenter.
- **Management networks visible from the VIM account**. Each VNF deployed by OSM needs to be accessible for configuration and operation, and this means that at least one of the VNF interfaces will have to be connected to a management network with an IP address that is reachable from OSM. This requires not only the creation of the network in the VIM, but also guaranteeing that the network is usable from the VIM account that will be added to OSM, and is reachable via IP from OSM.

- **SDN controller accounts and relevant port mapping info for SDN Assist**. In those cases where OSM has to manage underlay networks on a datacenter through an SDN controller (*SDN Assist*), OSM needs to know the access information and credentials to reach the SDN controller and perform CRUD operations related to underlay networks. The setup of the SDN controller and its accounts, and its connection to the datacenter infrastructure needs to happen in a provisioning phase, typically by an administrator of the datacenter. Then, the SDN controller access information and credentials need to be added to OSM and associated to a specific VIM prior to any deployment in the datacenter controlled by that VIM. In addition, OSM will need to know the mapping between compute node interfaces, e.g. the tuple (compute node, physical interface), and SDN ports, e.g. the tuple (switch, port). That mapping will guarantee the coherence of the operations from OSM to the VIMs and SDN controllers.
- **WIM accounts and relevant port mapping info**. In order to be able to deploy inter-datacenter or inter-VIM NS/NSI, OSM will have to contact potentially one or several WIMs, and it will need to know the access information and credentials of the WIM account to be used. Those credentials are created by a WIM administrator, and should provide OSM enough rights to perform CRUD operations on inter-DC/inter-VIM networks. Those WIM accounts need to be added to OSM, prior to any multi-site deployment that may require the creation of inter-DC/inter-VIM networks. In addition, OSM will need to know the mapping between external DC/VIM ports, e.g. the tuple (VIM, swich, port) and the transport service point, e.g. the tuple (transport switch, port), which will guarantee the coherence of operations from OSM to the VIMs and WIMs.
- **PDUs**. In those cases where we want to define NS packages or NST consisting of PNF packages or HNF packages, OSM is instructed about the available PDUs, i.e. the network appliances that will be the building blocks of those PNF and HNF packages. We will need to provide the management IP address of those PDUs, their type (so that they can be managed as a pool if appropriate), their location (VIM) and the physical connections to the site (e.g. switch and port of the infrastructure where the PDU is connected).
- **PNFs and external networks**. In those cases where the Network Service (or Network Slice) needs to be connected to existing networks or PNFs that are not part of the Network Service itself, we need to feed OSM with appropriate information to connect those entities to our Network Service. One possibility among others would be adding those entities in OSM as external networks of a VIM so that they could be mapped to a Virtual Link in the Network Service at instantiation time. In addition, if OSM controls the creation of underlay networks, it will be necessary to provide the datacenter switch and port where the PNF or external network is attached.

### Catalogs and shared databases

In a standalone/greenfield configuration, without the possibility of relying on an external catalogue service, OSM is able to manage the catalogue of the different resources used to build, create and operate Network Service and Slices:

- Physical Deployment Units

- Network Function packages (VNF, PNF, HNF)
- Network Service packages
- Network Slice Templates
- Network Functions: instances of running VNFs, HNFs or PNFs
- Network Services: instances of running NS
- Network Slice Instances
- VIM accounts
- SDN controller accounts
- WIM accounts

The catalogue of Network Services and Network Slice Instances also stores the history of operations over that NS/NSI.

Regarding the technologies for storing those catalogues, OSM may support different types back-end options for brownfield/integrated setups.

### Platform logs and alams

In addition to the capability to export the logs to external systems, OSM can store internally the following logs:

- System information, relevant to know the status of the servers hosting OSM and the relevant processes
- Operational information. Each API call should be properly logged by the system

Log rotation policies are expected to be configurable and they should follow the log rotation company's policies.

### Security

OSM has to provide the following security-related functionalities with respect to its northbound interface:

- Authentication and authorized access from the clients.
- Proper isolation of catalogues per project. In some cases, as mentioned before, some of the catalogue resources could be shared between projects.
- Secured NBI based on TLS

In addition, OSM provides mechanisms to hinder malicious access that could compromise both the system and the data stored by the system. As an example, the internal architecture of OSM is prepared to follow common security practices in commercial distributions and installers, so that there are:

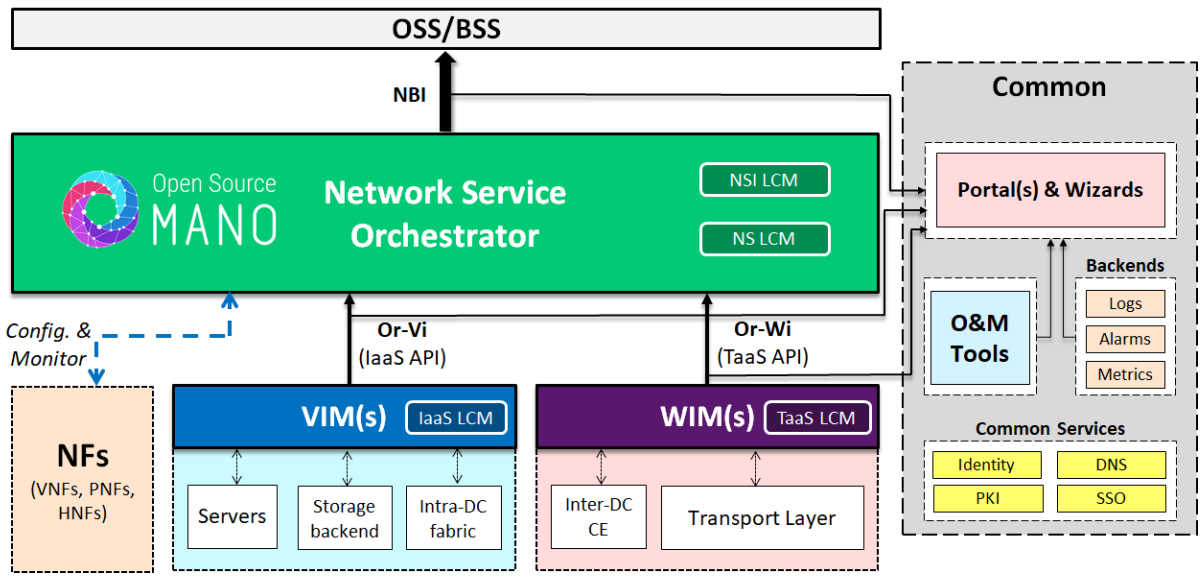
- No passwords stored in text files
- No default users and passwords in internal databases
- Proper encryption of the account information fed to the system (e.g. passwords from VIM accounts)
- No processes running as root user directly in the system host.
- Secured and authenticated communication between internal modules only if they run in different servers.

# OSM Integration Guidelines

Once OSM functionality and management have been throughly described, it is worth a recap of the strategies tham may be followed in order to integrate OSM into a global NFV/Network architecture based on a service platform approach and understand the expected interaction with other platform's services and a reference set of pre-existing common management and auxiliary services.

## Reference Architecture of Service Platforms and Common Management

In order to conduct the exercise, in our example we will take as starting point the service platform presented during the discussion of OSM Scope and Functionality and we will add a tentative set of auxiliary services and tools for joint management of the service platforms which will be taken as reference during the rest of the chapter:




**OSM in Service Platform architecture with Common Management**

Based on the Service Platform view thoroughly described in the discussion about OSM's scope and functionality:

- **OSM** as **Network Service Orchestrator** is in charge of controlling the lifecycle of Network Services and Network Slices offered on demand as a service northbound.

- There is a number of **VIMs** that work as managers of *Virtual Infrastructure Platforms*. They would expose the reference point Or-Vi northbound to provide IaaS on demand. This reference point would be accessible for OSM to obtain any IaaS needed to compose and manage the NS/NSI. In cases where a given VIM where unable to provide underlay connectivity, it might be possible to follow the *SDN Assist* approach, where OSM will work with an *augmented VIM* API, composed of a vanilla VIM API and an SDNC API.
- Likewise, there would be a set of **WIM(s)** that would work as managers of **SW-Defined Networks**. The WIMs would expose the reference point Or-Wi northbound to provide connectivity as a service among the VIMs (even if they required inter-datacenter connections). This reference point would be accessible for OSM to serve to the NS/NSI creation and lifecycle.

It must be noted that **a proper connection with these *service interfaces* is, strictly speaking, the only integration required to allow OSM to be completely functional and work normally**.

In addition to these well-known interfaces, the sample architecture incorporates a set of well-known **auxiliary services and tools for joint management of the service platforms,** which we can classify into different categories based on their nature:

- **Common services.** These functions offer basic functionalities, available for all the Service Platforms that may require them, which are intended to offer a common context easily accessible to all of them:
  - **Identity.** This service eases the management of user identities that can be shared across the different platforms with a well-known protocol (e.g. LDAP). In the context of API calls, this service typically works as a common service callable by the RBAC (role based access control) subsystem of each platform prior to any operation that requires issuing a token.
  - **Single Sign On (SSO).** This service is intended to provide a context after logging that is common across different web portals available in a portal collection, so that authentication in one of them can be trusted by the others.
  - **PKI.** A PKI infrastructure would facilitate the management of the different certificates as well as providing a common root source of trust for the local environment. This role is essential, among other purposes, to give client platforms means to recognize and trust the available serving platforms in a dynamic fashion.
  - **DNS.** A local name resolution service facilitates the use of common fully qualified domain names (FQDN) across the platforms, so that components and API endpoints can be easily referred by a name understandable by all the platforms.
- **O&M.** Collection of tools coming from the different platforms to ease their respective O&M operations and control their lifecycle, including the own platform setup process. One typical example are installers and lifecycle environments that come with different OpenStack distros.
- **Backends.** These functions are intended to ease the treatment of different types of data (permanent or volatile) that are generated by the platforms, so that it can be conveniently

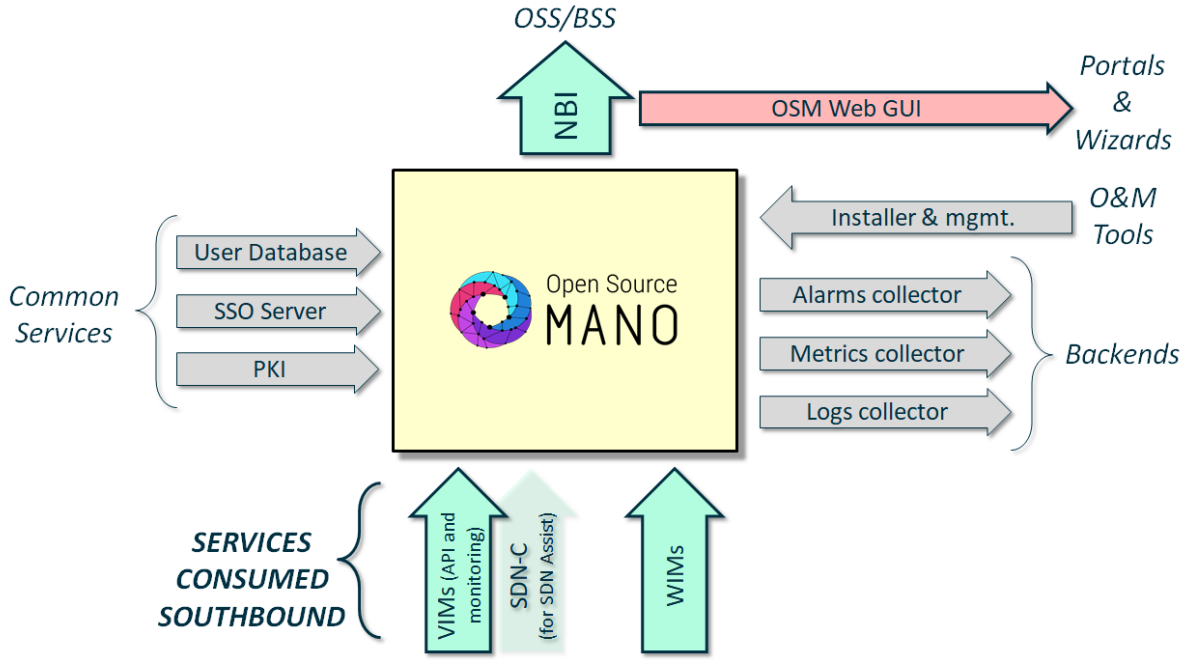
aggregated, filtered and correlated in a single point and also ease recurrent housekeeping tasks (e.g. replication), which can be conveniently centralized. Typically, these types of backends include a **logs backend**, an **alarms backend** and a **metrics backend**. Alarms and metrics backends can be used by the platforms to inject alarms and metrics related both to the platform itself or to the service objects that it offers on demand, so that they can be referred afterwads. On the other hand, logs and alarms backends might share the same type of backend technology in practice if the ingestion mechanisms can be normalized. On the contrary, the metrics backend might require a faster type of technology, such as a *Time Series Database (TSBD)*.

- **Web portals and wizards.** In order to ease some common operation tasks or gain a human-friendly/real-time view of the state of the platform and its components, there might be a set of tools of web-based portals that will facilitate such common interactions and might access both the service APIs and/or their respective O&M components. Into this category, we may typically find:
  - **Dashboards.** Intended to provide a global view of the state, either per platform or per global states that can be aggregated and/or filtered (e.g. alarms). Graphical visualization of topologies, templates and objects is a common feature, as well as the listing of the different catalogs and backend data, or various CRUD operations over them. These dashboards are usually capable of working as means to invoke on the demand the services offered by the respective platforms, requiring access to the corresponding service northbound APIs.
  - **Wizards.** In some cases, it is convenient to coordinate complex O&M operations that might involve more than one platform or which might benefit from a guided step-by-step process to minimize potential errors. Sometimes the roles of dashboard and wizard can be found bundled into the same tool (e.g Horizon).

It must also be noted that, although these auxiliary and common services are not strictly required to set up and operate the platform, are highly convenient to ease integration and troubleshooting across platforms, so it is a good practice to leverage on them for production setups, and hence we will consider them in these integration guidelines.

## Integration points

Taking as starting point the sample architecture described above, the following picture summarizes which integration points from the ones described above are relevant from OSM's perspective and might require a careful consideration and proper identification in the target environment:




**OSM integration points**

*(DNS is ommitted, as it is self-explanatory)*.

### Service view (northbound and southbound)

#### *VIM(s)*

To proceed with this integration, the following checks are recommended:

- Check if the VIM(s) to be used belong to VIM families already supported by OSM (otherwise a new VIM plugin might be required).
- Obtain the respective service endopoints of the VIM(s).
- Obtain accounts and tenants per VIM with the required permissions and privileges to perform all NSO operations.
- Check if there is a *management network* (sometimes called *OAM network*) in the VIM to attach the management interfaces of the VNFs and which is reachable from outside by OSM. Othewise, create it.
- If some VIMs were unable to create underlay connections intra-VIM, consider for those VIMs the use of and *SDN Assist* configuration. In those cases:
  - Check if the SDN Controller(s) are connected to a fabric that is properly configured to perform those intra-VIM underlay connections. If so, obtain the proper port mapping.
  - Check if the SDN Controller(s) to be used belong to an SDNC family already supported by OSM (otherwise a new SDNC plugin might be required).
  - Obtain the respective service endopoints of the SDNC(s).

– Obtain accounts and tenants per SDNC with the required permissions and privileges.

Once these checks are made and the proper information is gathered, the integration would be completed by the mere addition of these *boundary conditions* via one of the regular procedures in OSM (GUI, CLI, API). In this case, this action would mean the addition of VIM accounts (*'targets'*), including credentials, API endpoint, name of management networks, etc. Whenever required, the VIM account would also include the rest of the information required for *SDN Assist*.

In case there were some PDUs (or PNFs) which needed to be considered in an initial setup, they would be added to OSM via this same procedure in OSM, providing their management IP addresses, their type (if they should be pooled), their VIM of reference and the physical connections to the site (e.g. switch and port of the infrastructure where the PDU is connected).

#### *WIM(s)*

To proceed with this integration, the following checks are recommended:

- Check if the WIM(s) to be used belong to WIM families already supported by OSM (otherwise a new WIM plugin might be required).
- Obtain the respective service endopoints of the WIM(s).
- Obtain accounts and tenants per WIM with the required privileges to perform all NSO operations.
- Learn the mapping between external DC/VIM ports, e.g. the tuple (VIM, swich, port) and the transport service point, e.g. the tuple (transport switch, port).

Once these checks are made and the proper information is gathered, the integration would be completed by the mere addition of these *boundary conditions* via one of the regular procedures in OSM (GUI, CLI, API). In this case, this action would mean the addition of WIM accounts and set of mappings.

#### *OSS/BSS*

OSM's NBI should be accessible from the OSS and BSS platforms. In addition, at least an user and tenant in OSM with sufficient privileges would need to be created to be used by the respective OSS/BSS platforms.

### Common auxiliary services and tools

#### *Common services consumed*

If properly instructed, OSM can use existing available common services in order to minimize the provisioning time and improve the coordination with the rest of platforms in the environment. These kinds of platforms can be successfully leveraged by OSM:

- **User database.** OSM's RBAC can be leverage on an external user database via well-known protocols (e.g. LDAP) to support the user authentication during the RBAC phase.
- **Single Sign On Server (SSO).** OSM's web GUI can work in coordination with other pre-existing web portals sharing authentication via SSO, so that login/logout info is common across them.

- **Public Key Infrastructure (PKI).** OSM by default validates the authenticity of the endpoints of any platform it interacts with (VIMs, WIMs). In order to ease the setup for thus purpose, OSM may leverage on a pre-existing PKI to validate the autenticity of the endopoints.

### *Backends for OSM's outputs*

OSM produces and manages a large set of relevant information related to:

- Events and states in the Network Services and Network Slices and their components.
- Information related to its internal events as *manager function*.

OSM feed this information regularly to three types of events:

- *Log entries*, which can be fed to a central **log collector**.
- Alarms and other types of events, which can be fed to a central **alarm collector**.
- Likewise, there are a number of metrics related to the NS/NSI and OSM as *manager function* which can be fed to a **metrics collector.**

### *O&M tools*

**OSM's installer** (community-based or commercial) and companion **OSM's O&M tools** should be deployed in the same architectural region (i.e. with the same restrictions, security and access rules) as the rest of O&M tools of the architecture.

### *GUI and dashboard for alarms, logs and metrics*

OSM's **web-based GUI** can be added to a pre-existing collection of web portals (i.e. with the same restrictions, security and access rules), only by granting access to OSM's NBI and the common SSO service.

Likewise, it is also possible to include OSM's dashboard for alarms, logs and metrics in the collection of web portals provided it has proper access granted to the approrpiate backends as well as the SSO service.


ETSI
06921 Sophia Antipolis CEDEX, France
Tel +33 4 92 94 42 00
info@etsi.org
www.etsi.org