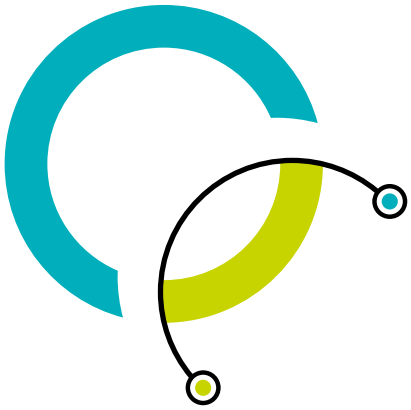


OpenSlice

by ETSI

OpenSlice White Paper

OSL 2024Q4
RELEASE WHITE PAPER

1st edition – March 2025

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

ETSI

Authors

OSL Leadership Group

OSL Technical Steering Committee

Contents

Introduction

With Release 2024Q4, OpenSlice (OSL) aims to remain at the forefront of addressing the evolving needs of the telecom industry, focusing on overcoming current challenges faced by telcos. This latest release brings optimized resource orchestration, improved integration with modern telecom tools, and more efficient deployment and monitoring processes. These advancements ensure that OSL continues to lead the way in supporting both research and industry sectors. The release also introduces replicable functionality, setting new standards for modern Operator Platforms, such as CAMARA and TM Forum Operate APIs. Additionally, it strengthens telco cloud scenarios through synergies with LF Sylva, leveraging the power of open, contemporary, and standardized APIs.

Useful links:

- Subscribe for Release 2024Q4 Webinar (3rd April 2025)
- OSL News Page Release 2024Q4 announcement
- ETSI announces OSL Release 2024Q4
- Software documentation

Release 2024Q4 Highlights

Among others, the latest Release 2024Q4 fosters the following key highlights:

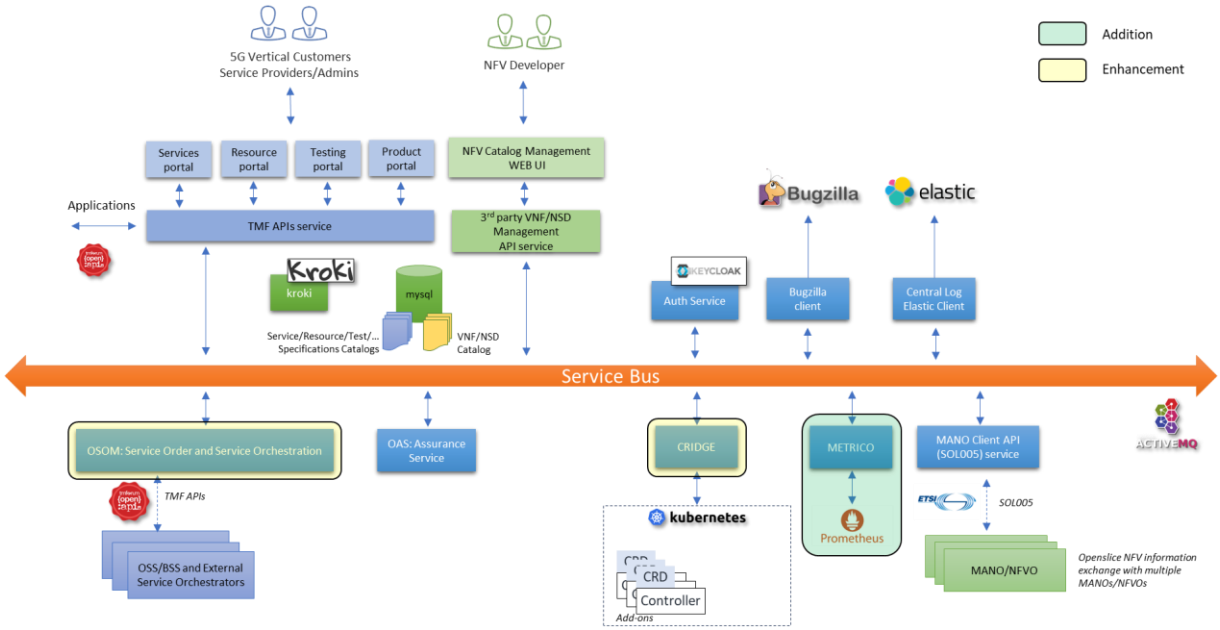
- **OSL Addons:** A newly launched and continuously updated Addons repository leverages Custom Resource (CR) orchestration support to enable modern telco features and use cases:
  - **CAMARA:** Allows the exposure and usage of Quality on Demand (QoD) API
  - **LF Sylva:** Provides a Kubernetes operator to manage Sylva workload clusters
- **Kubernetes support extension:** The enhanced CR orchestration logic, boosts stability, expands Kubernetes resource exposure, and introduces intuitive resource mapping and lifecycle management
- **TM Forum APIs extensions:** Incorporating Resource Activation and extending the Product Layer TM Forum APIs
- **Robustness and Performance enhancements:** Bringing significant enhancements to concurrent, multi-tenant usage while optimizing key components to minimize their footprint for sustainable long-term deployments.
- **End-to-end testing pipelines:** Enabling seamless, repeatable validations of deployment and orchestration examples with every codebase update.
- **METRICO:** A new (experimental) inbuilt mechanism to integrate monitoring solutions within the service lifecycle, facilitating efficient closed-loop management.

The extensive release notes are available to probe further into the list of overall changes introduced during Release 2024Q4, along with the updated internal components' versions.

Architectural Progression

Release 2024Q4 builds on the already introduced service-based OSL architecture, also incorporating a new component, i.e. METRICO.

Furthermore, the architecture underwent several core enhancements in the OSOM component, to facilitate the support of METRICO within the orchestration logic, and also in the CRIDGE component, already introduced in Release 2024Q2, taking into account numerous suggestions and feedback from the userbase.



OSL Release 2024Q4 Architecture

OSL Addons

OSL Addons repository introduces an archive of reusable and replicable projects that leverage OSL architecture to enable telco cloud scenarios and expose capabilities of a modern Operator Platform.

Stay tuned as addons are continuously updated!

CAMARA

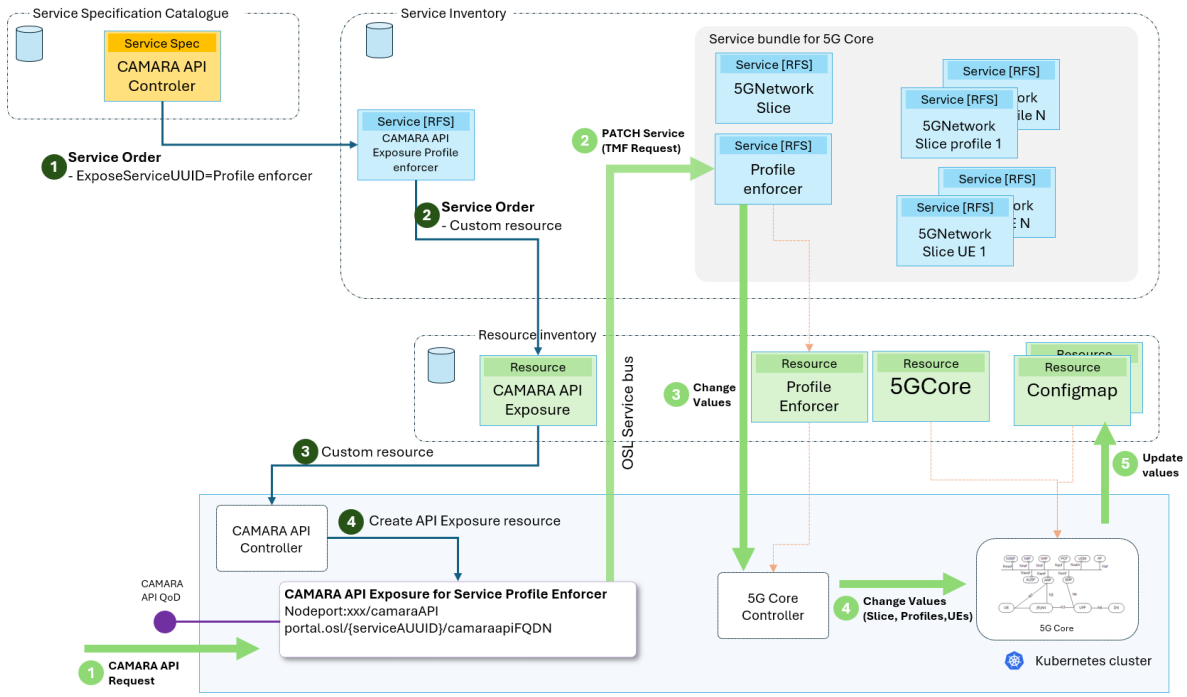
The CAMARA as-a-Service (CAMARAaaS) OSL Addon is a prototype service developed by OSL and allows users of OSL to expose CAMARA APIs for controlling their TMF-based services. By doing so, it enables runtime operations, such as enforcing Quality of Service (QoS) profiles on User Equipment (UEs) or updating 5G Network Slice characteristics, using standardized CAMARA API endpoints. The work is in progress for future enhancements (e.g. multi-tenancy, etc).

In a nutshell, CAMARAaaS Addon performs API transformations from CAMARA API model to TMF API model and vice-versa. The supporting use case is the following:

- An OSL Service Provider (e.g. an Operator) has a running 5G Core (e.g. from another service order in OSL).
- A running 5G controlling service exposes already some characteristics (i.e. via TMF Service Inventory) that can be configured. Thus, someone can reconfigure the latter during runtime (e.g. change the quality of a slice via a TMF API service request).
- On a subsequent step, the Service Provider makes a Service Order in OSL to expose this running 5G Core service via a CAMARA API endpoint.
- The CAMARAaaS Addon is a wrapper between the CAMARA requests and the TMF API Service Inventory models. These CAMARA APIs will then be used to control the lifecycle and the operations that shall take place in an already existing OSL Service.

The Addon introduces a generic CAMARA API Service, which acts as a wrapper for existing (running) services registered in TMF Service Inventory. With the assumption that there is a 5G controlling running service, the architecture ensures:

1. **API Exposure**: CAMARA APIs are orchestrated by OSL (offered as-a-service) and their endpoints are exposed to the end-users (clients).
2. **Service Mapping**: The CAMARA API Service references a running service (identified by a unique ID), enabling targeted operations. The invoking of CAMARA API endpoints will result in updates in the running service's characteristics.
3. **Operational Flow**: Updates triggered via CAMARA APIs are propagated to the operator's service through OSL's message queue (Active MQ), ensuring synchronization of service characteristics.



OSL CAMARAaaS Addon Architecture

The exposure of CAMARA APIs, along with the support of OSL on multi-domain scenarios makes it an ideal platform for supporting the GSMA OpenGateway initiative, spearheaded by TM Forum, GSMA, and CAMARA.

Read more about the topic at:

- Offering CAMARAaaS
- Quality on Demand (QoD) Provisioning API Proof of Concept (PoC)

LF SYLVA

Sylva, a project under the Linux Foundation, is designed to meet the unique demands of the telecom and edge cloud sectors. As telco networks increasingly transition toward edge computing environments, Sylva provides a tailored cloud software framework for these specialized needs. It does not only address technical challenges in this domain, but also provides a reference implementation of the framework, which can be validated against industry requirements.

The primary goal of Sylva is to ensure that telecom operators have a reliable, flexible, and scalable cloud infrastructure that supports the edge computing needs of modern networks. Additionally, Sylva

aims to create a validation program for these implementations, ensuring that they meet the evolving standards and demands of the industry. You can read more about Sylva here.

The synergy between OSL and Sylva offers a powerful combination that addresses the increasing complexity of managing telco and edge cloud infrastructures. In Release 2024Q4, we developed an Addon operator that enables the integration of OSL and Sylva, thus optimizing service orchestration and resource management for telecom operators. This is demonstrated through new resource operators that OSL developed as extensions. These operators allow OSL tenants to request and manage Sylva workload clusters directly through OSL's platform and TMF APIs. This capability means that tenants can now order Kubernetes clusters managed by Sylva in a self-service manner, simplifying resource allocation and management. The integration of Sylva with OSL leverages TMF APIs to expose and manage Sylva's telco-focused cloud software framework. By doing so, OSL provides an efficient means of ordering and orchestrating Sylva's resources within the broader context of network services.

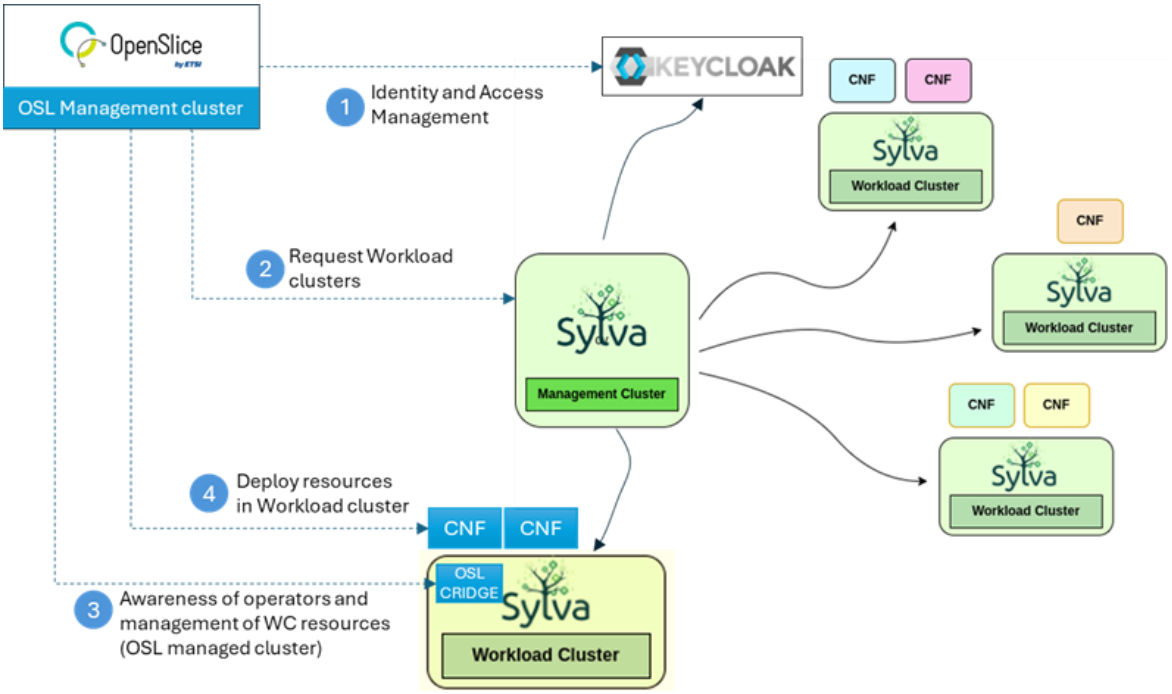
The collaboration between OSL and Sylva highlights several key advantages for telecom operators, including:

- **Self-Service Resource Management:** OSL tenants can seamlessly request and manage Sylva workloads, providing greater flexibility and speed in resource provisioning.
- **Streamlined Orchestration:** By utilizing TMF APIs, OSL simplifies the process of requesting and managing Sylva clusters, reducing the complexity of managing edge cloud infrastructure.
- **Scalability and Flexibility:** Sylva's cloud framework is built specifically for the telco industry, offering scalability and flexibility for telecom operators looking to expand their edge computing capabilities.
- **Efficiency and Automation:** The integration enables automated resource provisioning and lifecycle management, ensuring that telecom operators can focus on innovation rather than manual resource management.

We explored the following aspects, which are also represented visually in the following figure:

1. **Identity and Access Management:** This setup uses Keycloak for Identity and Access Management (IAM). Both OSL (installed in an OSL management cluster) and Sylva's services (Sylva services installed in management cluster) interact with Keycloak to authenticate and authorize users or services, ensuring secure access to resources. This step is crucial for managing user permissions, access rights, and the security of operations.
2. **Request Workload Clusters:** The **OSL Management Cluster** interfaces with the **Sylva Management Cluster** to request workload clusters. This is a crucial aspect of the integration, as OSL provides a self-service capability for tenants (e.g., telecom operators) to request Sylva's Kubernetes-based workload clusters easily.
3. **Awareness of Operators and Management of Workload Cluster Resources:** After the workload cluster is created and managed by Sylva, OSL becomes aware of the workload cluster operators and the resources within the workload clusters. In this process, OSL is managing the workload cluster resources as if they were its own. OSL's CRIDGE, is utilized to ensure that OSL is fully integrated and aware of the Sylva-managed resources.

4. **Deploy Resources in Workload Cluster:** Once the workload clusters are set up and managed, Cloud-Native Functions (CNF) are deployed in the clusters. These CNFs represent the telco workloads and applications that the operator needs to manage. The deployment of CNFs takes place in the workload clusters, utilizing Sylva's capabilities to handle these resources efficiently.



OSL – Sylva integration

More details of the approach can be found in the following three-part news blogposts:

1. Details in Identity and Access Management
2. Sylva Workload Cluster as a Service
3. Awareness of resources in a Sylva Workload Cluster

The Addon's source code is in the respective OSL GitLab repository.

The feature was part of the demonstration that was performed during the SNS4SNS event at ETSI, Sophia Antipolis, France, 12-14 Nov. 2024. More information about the demos can be found here.

Also, make sure to check out the related video demonstration.

METRICO

Services and Resources created through OSL potentially provide a vast amount of data and metrics during their runtime. These data could be leveraged to perform actions upon the said services. Before the introduction of the METRICO component, OSL would need to rely exclusively on external tools' implementation to report metrics to it.

The newly introduced METRICO is designed to create an inherent mechanism that initiates customizable metrics-retrieval jobs and associates the data with services under scope. To address the challenge of data retrieval across multiple and heterogeneous environments, OSL is not coupling tightly with a monitoring solution, but is able to integrate with any monitoring stack that provides a programmable API, e.g. Prometheus. Prometheus is considered a widely utilized monitoring solution and many alternative monitoring tools converge towards its data model via implementation of Prometheus Exporters. On the other hand, OSL remains technology agnostic by capturing the monitoring requirements (e.g., data source, queries, interval, etc) through the TMF628 Performance Management model and API.

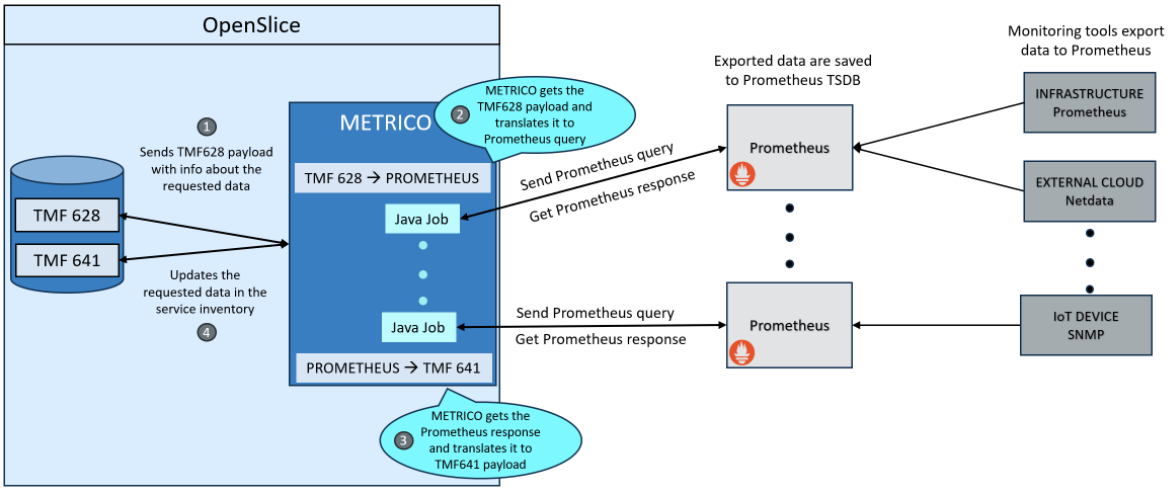
In a nutshell, METRICO allows OSL users to:

- integrate already owned monitoring systems with OSL (also supporting several parallel integrations)
- retrieve important metrics inherently through OSL Service Inventory
- natively empower the service lifecycle with external data through LCM rules
- perform closed-loop control management

OSL comprises the offered Service Specifications which, when ordered, create the respective Service instances, accessible through the Service Inventory (based on TMF638). The rationale behind the introduced component is that it exposes a defined METRICO Resource Specification which is used to design monitoring-related Service Specifications. The latter comprises meaningful characteristics (e.g. monitoring data source, query, interval, affected service, etc) that are filled in by the requester.

When the user orders a monitoring-related Service Specification (with populated characteristics of its preference), a TMF628 Measurement Collection Job entity is created which is responsible for gathering the requested metrics. Respecting the desired interval, this job returns the monitoring results to the appointed service in the Service Inventory, patching a designated characteristic.

The overall approach is summarized in the following figure:



METRICO Approach

Read more about the topic at:

- METRICO Introduction
- Design Monitoring Service
- Example: Integrate Prometheus monitoring solution as-a-Service

End-to-End Testing

The release also brings automation into testing OSL deployment, critical components, and functionality. Therefore, we present a new CI/CD repository[1] that ensures the stability of the overall software throughout code changes during the release cycle.

Specifically, the introduced repository automates the deployment, testing, and cleanup of an OSL instance and related services using Python and Helm. It provides a robust framework for validating OSL's functionality in Kubernetes environments. Namely, it:

- Automates the deployment of Argo CD and OSL within a Kubernetes environment
- Performs end-to-end testing for service catalog management and service orders
- Ensures that the expected characteristics are being exposed after a successful Kubernetes-based deployment
- Supports cleanup operations to uninstall resources after tests

The above pipeline is scheduled to run on a nightly basis on the develop branch and weekly on the main branch, respectively.

[1] Requires login

Service Specification Exporting / Importing

Since Release 2024Q4, the OSL team decided to create a new Utilities repository to provide replicable functionalities that facilitate the usage of OSL instances. The initial utilities we brought to the userbase are the Service Specification Exporting and Importing.

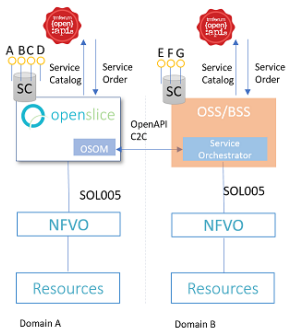
Specifically, a user can leverage the Service Specification Exporting utility to export designed service specifications from an OSL instance as packages. These packages cover all the relationships within a Service Specification, i.e. Resource and Service Specification relationships, Lifecycle Rules, Characteristics, etc, following an upside-down tree approach, starting from the lower-level, non-dependent Service Specification and moving upwards the bundle.

Similarly, the Service Specification Importing utility uses the exported packages and follows the exact opposite procedure to build the Service Specification bundle and inject it at the designated OSL instance.

You may probe further into the topic at:

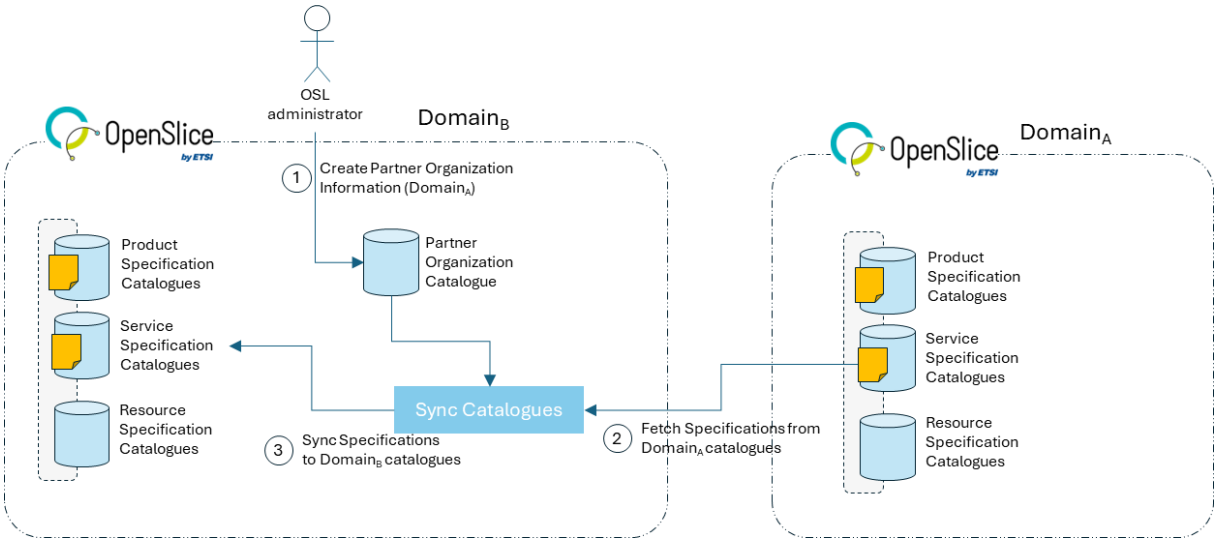
- Service Specification Exporting repository
- Service Specification Importing repository

Multidomain and Federation scenarios



Ever since 2018[2] OSL has been used in various cases supporting multi-domain exchange of services and resource catalogs. It has also been used for cross-domain service orchestration[3] and common federation scenarios, such as user federation through a shared authority like Keycloak. This was demonstrated in the LF Sylva Addon section. Additionally, OSL supports service federation through TMF-related APIs, not only for exposing catalogs and accepting service orders but also for implementing East-West interfaces between domains.

The first presented scenario is related to catalog synchronization, where one domain (B) (e.g., an operator) retrieves related service specifications from a partner domain (A). This enables Domain (B) to offer new services or bundles, involving services from partner domain (A).



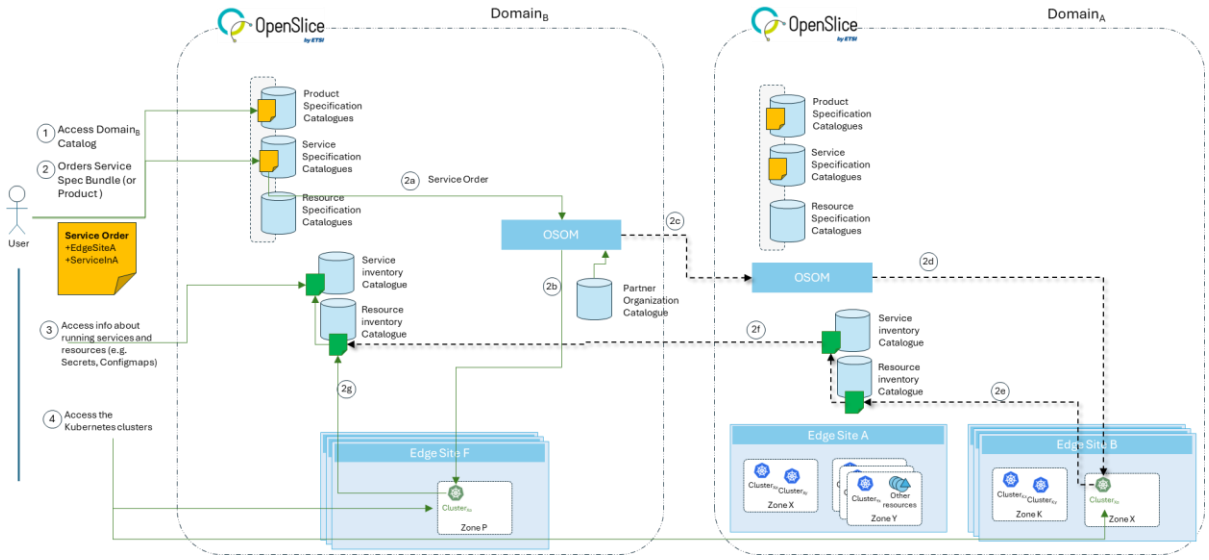
OSL Synchronize Catalogs via TMF Service Catalog APIs

As soon as Domain (B) exposes available services, that involve other services from partner domain (A), users of the domain can access the catalog and place service orders, as depicted in the next figure. Then, the Service Orchestrator from Domain (B) instruments the Service Orchestrator from Domain (A) to deliver the service (2c step) and expose it to its inventory. OSL then synchronizes the Domain (B)

[2] https://osl.etsi.org/documentation/2024Q4/deployment_examples/

[3] https://zsmwiki.etsi.org/index.php?title=PoC_2_Automated_Network_Slice_Scaling_in_Multi-Site_Environments

resource and service inventories so that the user can get information (step 3) and access (step 4) the requested resources of the partner Domain (A).



Users of Domain$_B$ can Order services involving services from Domain$_A$

Further details on how federation scenarios can be realized are available in the respective documentation section.

Generic OSL Controller

Since Release 2024Q2, OSL introduces extensive support for Kubernetes operators exposed as services within OSL catalogs. However, inspired by the operator pattern, developers can now write their own resource controllers and attach them to the OSL service bus. OSL's Service orchestrator (OSOM) has the capability of contacting external controllers, given a specific resource category that this controller can manage.

The intended goal is to write a controller that can handle resources of a specific type, e.g., resource Specifications for managing resources of a specific category. Therefore, a new resource controller can be registered into OSL, in the form of a Resource Specification with a designated name, category and version, and the implementation of the controller shall listen for messages in queues as specified by the name, category and version of the registered Resource Specification. Specifically, it shall listen for CREATE/UPDATE/DELETE actions, with the following scheme:

- CREATE / category_name / version
- UPDATE / category_name / version
- DELETE / category_name / version

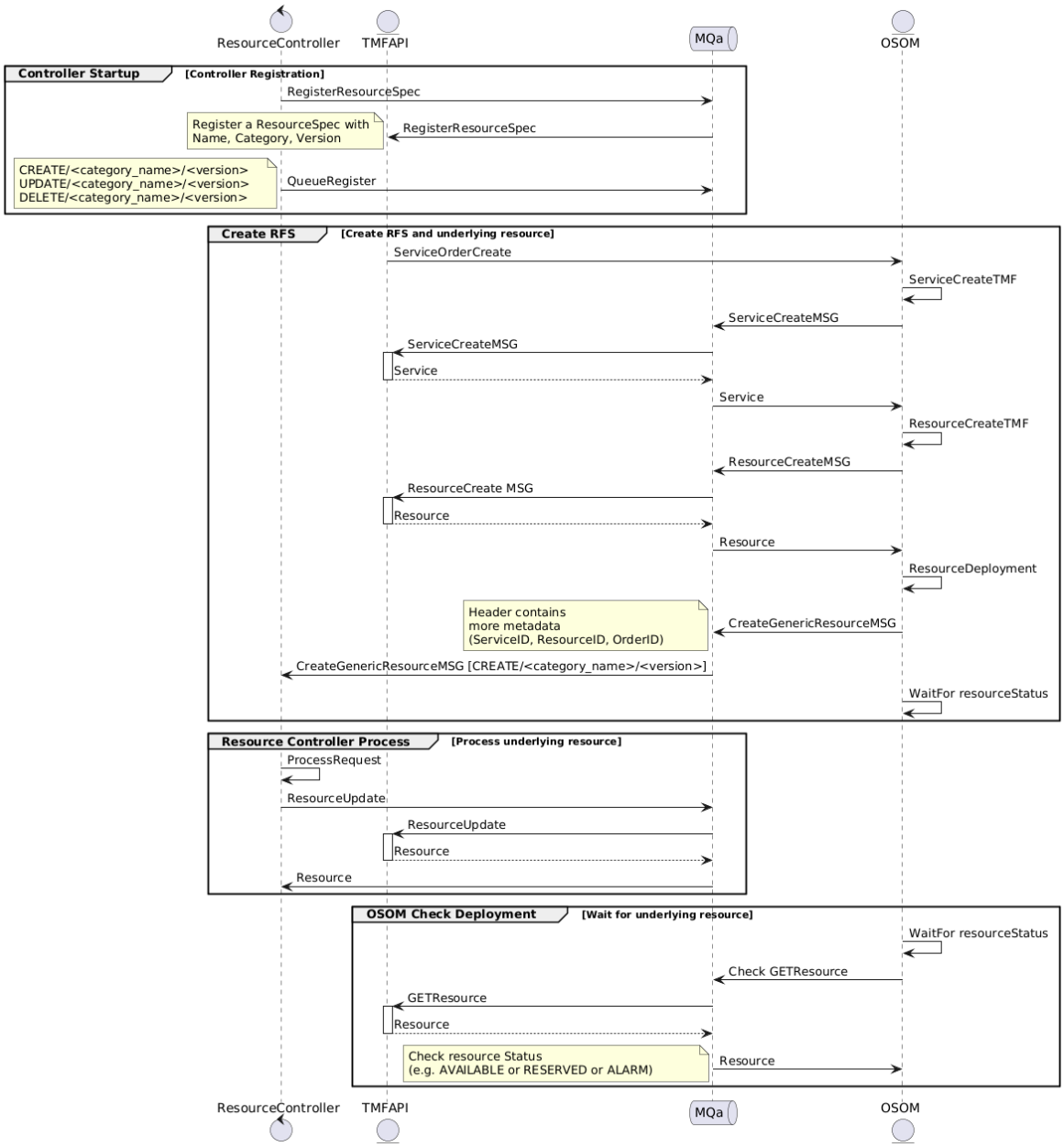
In a nutshell, the Resource Definition/Specification is registered at startup. During Service Orders of related-to-the-specification services, the controller is invoked (via message queue) and a new service and its underlying resource are created, with messages passing between the service and resource layers. The resource controller processes any updates or status changes. OSOM checks the final status of the deployed resource to confirm it is ready or identify any potential errors. The following diagram describes what the resource controller needs to perform, showcasing how the "Resource Controller" component interacts with various services (TMF API, Message Queue MQa, and OSOM) to register the needed resource types, create new resources (Resource Facing Services - RFSs and underlying resources), process updates, and check the resource's status.

1. **Controller Registration (top swimlane) – This happens when the controller bootstraps**
    - **Register a Resource Specification**: On startup, the Resource Controller posts a ResourceSpec (containing a name, category, and version) to register it in the OSL Resource Specification Catalog.
    - **QueueRegister**: The controller is registered in these three queues and listens on messages with ResourceCreate or ResourceUpdate payloads.
2. **Create RFS (middle swimlane) – This happens when there is a Service Order**
    - **ServiceOrderCreate**: When a new RFS needs to be created, a "ServiceOrderCreate" operation via the TMFAPI arrives to OSOM.

- **ServiceCreateMSG → ResourceCreateMSG**: Internally, a "ServiceCreateMSG" as well as a "ResourceCreateMSG" are sent to the TMFAPI component to create the related entities in the inventory.
- **Resource Deployment**: OSOM sends a message to the queue under the specific queue name to CREATE the resource.
- **CreateGenericResourceMSG**: A generic creation message is sent (e.g., CREATE/<category_name>/<version>), containing metadata such as ServiceID, ResourceID, and OrderID in the message headers.
    1. *org.etsi.osl.serviceId*: The related service ID that refers to the created resource.
    2. *org.etsi.osl.resourceId*: The related resource ID that is the custom controller is expected to update.
    3. *org.etsi.osl.serviceOrderId*: The related service order id of the deployment request.
- **WaitForResourceStatus**: OSOM waits for a status response indicating success or failure in creating the resource.

3. **Resource Controller Process (lower-middle swimlane)**
    - **ProcessRequest**: After the resource is created, the Resource Controller processes any additional instructions or updates related to the resource.
    - **ResourceUpdate**: The resource is updated accordingly (e.g., changing configurations, state, etc.). The controller needs to update the resource into a valid state, e.g. AVAILABLE, RESERVED, or ALARM. For instance, if it is update in AVAILABLE state, OSOM later will assume that everything is OK, the service is ACTIVE, and the Service Order is COMPLETED, respectively.
4. **OSOM Check Deployment (bottom swimlane)**
    - **GETResource → Check GETResource**: OSOM retrieves the resource information to verify its deployment and status.

Check Resource Status: OSOM checks whether the resource has transitioned into a valid operational state (e.g., AVAILABLE, RESERVED, or ALARM).



A working example of such a generic controller can be found here, written in Java, but in general you can implement one in any language.

2024 SNS4SNS Aftermath

The OSL community gathered at ETSI Headquarters at Sophia Antipolis, France, 12-14 Nov. 2024, alongside other ETSI Software Development Groups (SDGs), for the vibrant and collaborative Software and Standards for Smart Networks and Service (SNS4SNS) Conference and Hackfests. This event provided an excellent platform for networking, exchanging insights, and delving into current implementations.

The Conference's highlights for OSL included:

- Demonstrating synergies between ETSI SDG OpenSlice and LF Sylva
- Open Source for Telco-Cloud: An ETSI SDG-based solution to facilitate zero-touch, multi-slice 5G deployment across the cloud-edge continuum
- Delivering Network as a Service (NaaS) with OSL

A special mention and gratitude must be given to the OSL community for the invaluable feedback and participation in our first OSL#1 Hackfest. Your insights have been pivotal, inspiring significant improvements that were incorporated within Release 2024Q4.

You may find more information and material at the dedicated OSL#1 Hackfect page.

1st OpenSlice Hackfest

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