27-06-2023

Milestone M7.2

Orchestration Platform Architecture Review and Documentation

Contractual Date:	30-06-2023
Actual Date:	27-06-23
Grant Agreement No.:	101100680
Work Package:	WP7
Task Item:	Task 2
Nature of Milestone:	(R) Report
Dissemination Level:	PU (Public)
Lead Partner:	GÉANT
Document ID:	GN5-1-23-486ed9
Authors:	Simone Spinelli (GÉANT); Hans Trompert (SURF)

Abstract

This document provides a brief description of Workflow Orchestrator, the orchestration platform architecture adopted by SURF, ESnet and GÉANT and proposed as a reference for other NRENs and R&E organisations.

Co-funded by the European Union

© GÉANT Association on behalf of the GN5-1 project. The research leading to these results has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101100680 (GN5-1).

Co-funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Table of Contents

Table of Figures

Executive Summary

Network Automation and Orchestration is becoming increasingly strategic for many institutions across the R&E community but there is still not enough clarity around what it means and implies for an individual organisation. Many vendors offer specific products in this area promising a standardised approach and interoperability, with some advising that taking a DIY approach is not recommended.

Nevertheless, SURF took this approach several years ago, when it wrote Workflow Orchestrator (WFO) and published it as open-source software under the Common Conservancy foundation: WFO is now available for NRENs and other communities, and has been adopted as GÉANT’s Automation and Orchestration platform.

This document briefly describes the Workflow Orchestrator architecture and approach, how flexibility and modularity are as important as actionability and how these requirements are met. It aims to show how the Workflow Orchestrator architecture can be a perfect fit for organisations that want to start automating and orchestrating their networks.

The introductory section provides a first, high-level description of the architecture. This is followed by a more focused description, specifically of two of the fundamental parts of Workflow Orchestrator: domain models and integration modules.

1 Introduction

This document describes the architecture of the orchestration platform based on Workflow Orchestrator, an open-source framework originally written by SURF, the Dutch NREN. The framework is now used in production at SURF and the Energy Sciences Network (ESnet), has been adopted by GÉANT and is considered by many NRENs to be the cornerstone of their network automation and orchestration solutions.

The growing complexity of network services and the introduction of concepts such as Network Functions Virtualisation (NFV), as well as the increased perception that the network is a commodity, have made it clear that managing networks manually is no longer feasible.

While many reference architectures are available for organisations to guide them in embracing the Digital Transformation process currently underway, these do not always translate into immediately usable solutions, and a having a reference architecture in itself does not guarantee that heterogeneous, homegrown initiatives that inevitably lead to increased complexity and difficulty of maintainability will be avoided.

On the other hand, commercial solutions sometimes lack the flexibility necessary to enable them to be introduced without disruption, or represent a risk of lock-in that must be avoided.

The WFO is an orchestration framework that:

- Is generic enough to be easily integrated into an existing environment.
- Is immediately actionable and can be introduced gradually, reducing disruption.
- Is open source and maintained by the community, hence does not come with the risk of vendor-lock-in.

The main concepts on which the platform relies are:

- **Domain models**: representations of a product offering, the reusable components of said product and the actions that an engineer can take to manage the lifecycle of an instance of a product for a specific customer (called a subscription in WFO terminology).
- **Integration modules**: abstraction layers that allow interaction with OSS and BSS.

The WFO sits in between authoritative sources of information (such as customer relationship management (CRM) or inventory systems for device management) and domain-specific automation systems (such as a Network Management System –NMS) and is responsible for managing the lifecycle of network services (or, more in general, products), keeping all involved systems aligned and consistent over time.

An overview of the architecture is depicted in the following diagram:

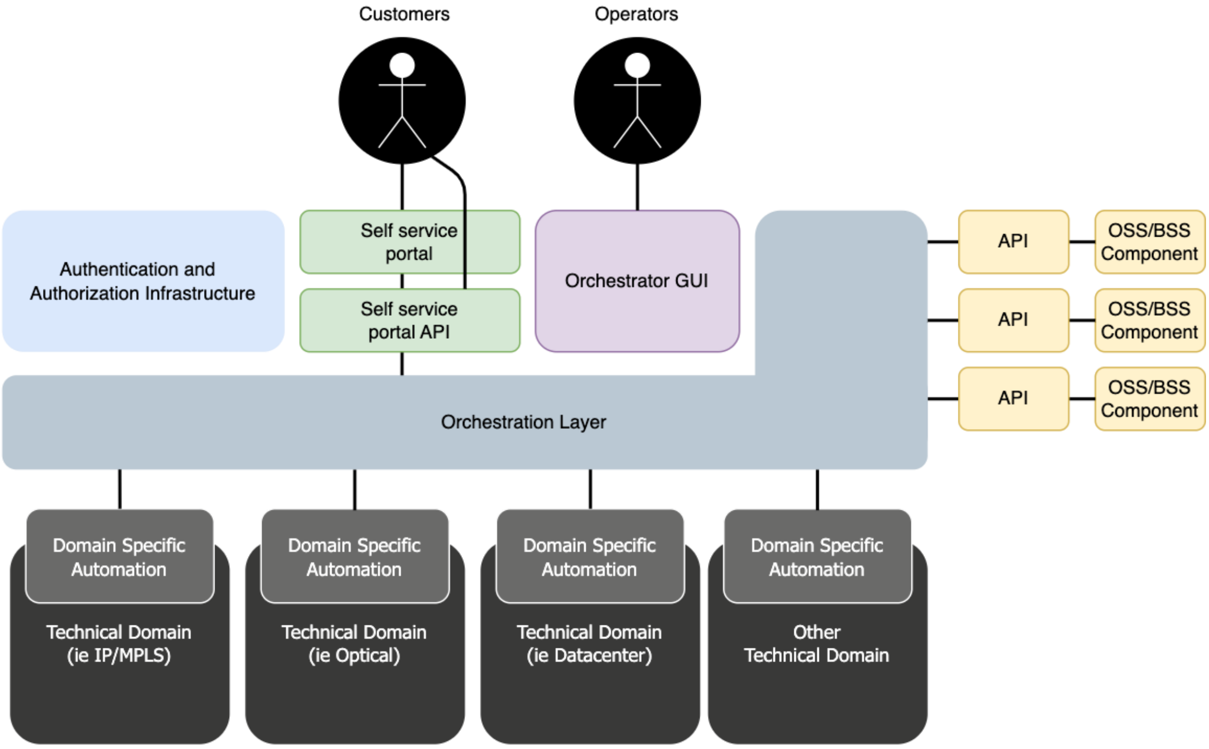
Customers
Operators
Authentication and Authorization Infrastructure
Self service portal
Self service portal API
Orchestrator GUI
API
OSS/BSS Component
API
OSS/BSS Component
API
OSS/BSS Component
Orchestration Layer
Domain Specific Automation
Technical Domain (ie IP/MPLS)
Domain Specific Automation
Technical Domain (ie Optical)
Domain Specific Automation
Technical Domain (ie Datacenter)
Domain Specific Automation
Other Technical Domain

Figure 1: Workflow Orchestrator Architecture Overview

The orchestration layer is responsible for mediating the intent of users and operators and abstracting resources and operations. It functions as a single point of access, and is usable by operators via a GUI that exposes all data and available operations.

WFO supports authorisation and authentication, making it possible to have different roles matching different views and rights.

It is very important to note that one of the benefits of a fully automated and orchestrated approach is to allow the introduction of self-service propositions that can be integrated in the existing architecture, as depicted in Figure 1 above.

2 Orchestrator Architecture

While an overview of the architecture has already been given in the introduction to this document, the following sections will focus on the main components mentioned: domain models and integration modules.

2.1 Domain Models

Domain models are representations of a product offering or rather of an item of the product portfolio that includes the current set of products or services that an organisation offers to its users or customers.

Domain models capture the design of the product and contain the attributes that are necessary to instantiate it, and govern the possible features and variations that might be necessary. In addition, domain models define the standard set of modifications and operations that are allowed for a specific product. This allows WFO to capture and express higher-level processes related to the product, enabling a controlled interaction between multiple systems and domains.

More specifically, a domain model contains a decomposition of the product into reusable **product blocks** so that it can be expressed as a combination of building parts. These building parts can be rearranged to form new products enabling the user to express and model complex products even across multiple technical domains.

A product block is itself an abstraction of the component that it represents, and is composed of a set of attributes called **resources**. This set of attributes can evolve over time making it possible to add or remove features and to adapt them to changing underlays if necessary.

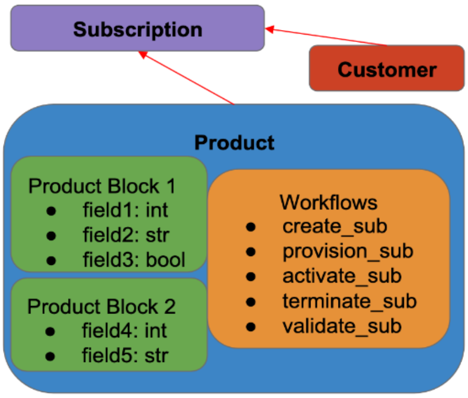

Figure 2: Products and subscriptions

Product blocks are defined in WFO as pydantic objects where attributes (resources) are strictly typed.

Resources might be information coming from the operator – such as a description of an interface – or from external authoritative systems such as for example an IP address supplied by the IPAM system. In every case, input is strongly validated against the model, making sure that both user input and input coming from other systems is compliant with the validation rules.

Subscriptions are then instances of a product and are linked to the customers that use them.

All the information coming from different sources is hence linked together within the subscription so that it can be periodically verified that all the sources of information as well as the real instantiation of the product are synced and coherent.

As mentioned, a domain model contains the set of actions that are applicable to the product to manage its lifecycle, from creation to termination, including special operations such as migration and validation. These sets of actions are called workflows. Workflows are defined by product and are run to create new, or update existing subscriptions for a specific customer.

Every workflow that is run by WFO generates a trail of information. This information includes who by and when the workflow was run, the product and type of workflow, customer and subscription details, and all workflow steps together with the information that is passed between workflow steps. The GUI makes available a convenient representation of this information for tracing and debugging purposes. Everything is stored inside the database and can be inspected in the future.

Workflows can be seen as ordered sets of atomic actions (steps) that are used to manipulate attributes, and lifecycle states – promoting the subscription from one to another – and interact with external systems to reflect the intent expressed by the model.

Lifecycles states, just like domain models, are customisable and adaptable to the specific needs of the organisation. An example of subscription lifecycle is depicted in the following diagram:

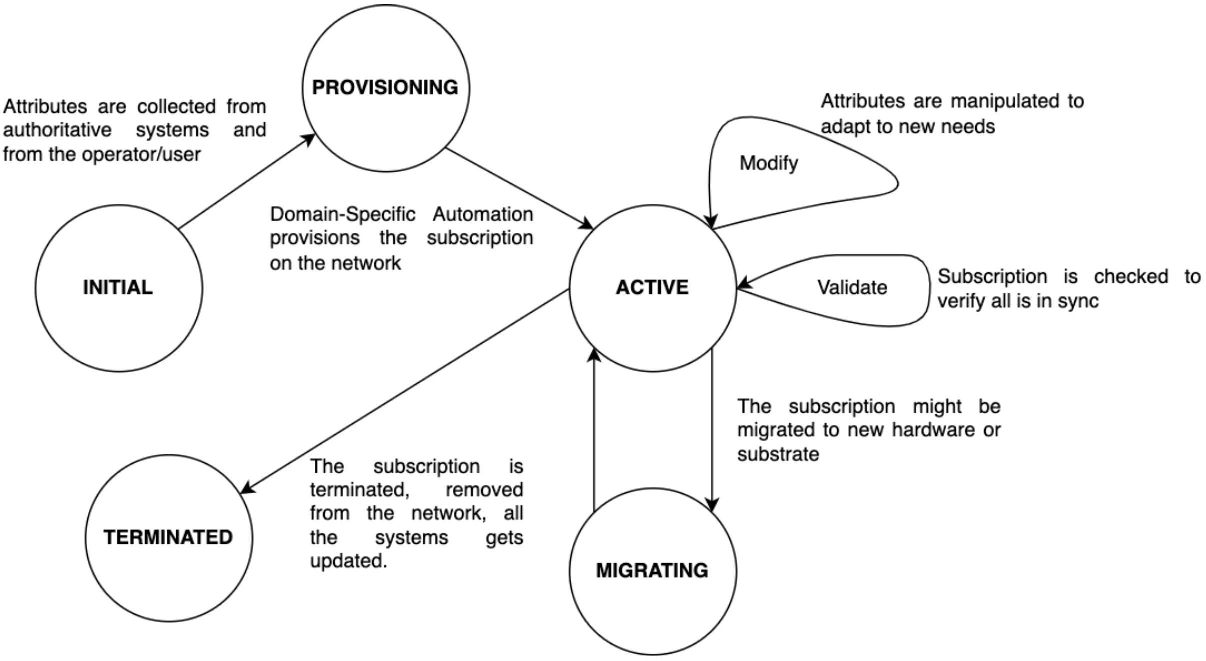
PROVISIONING
Attributes are collected from authoritative systems and from the operator/user
INITIAL
Domain-Specific Automation provisions the subscription on the network
ACTIVE
Attributes are manipulated to adapt to new needs
Modify
Validate
Subscription is checked to verify all is in sync
The subscription might be migrated to new hardware or substrate
TERMINATED
The subscription is terminated, removed from the network, all the systems gets updated.
MIGRATING

Figure 3: Example of lifecycle management

One last important property of workflows is that every single atomic step can fail and be retried, and this gives robustness to the entire system, allowing for the delays involved in waiting for external events to happen – for example for a field engineer to connect a cable – until the relevant step succeeds and the workflow proceeds.

It is worth remembering that all the information regarding how product blocks are linked together to form products and thus subscriptions, as well as the workflows available against subscriptions, are accessible and inspectable from the GUI, giving complete control of the lifecycle and change history including after a subscription has been terminated.

2.2 Integration Modules

WFO interfaces with external systems via a modular approach: this is realised with an abstraction layer between the workflow engine and the external service, the OSS/BSS proxy API. Figure 4 shows a close-up detail of Figure 1 illustrating the OSS/BSS interaction: on the left, the methods exposed to WFO are generic while on the right the interaction leverages system-specific APIs.

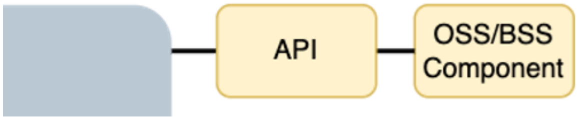

Figure 4: Detail of OSS/BSS interaction

Modularity and decoupling make it possible to:

- Introduce WFO in phases, integrating it with an initial set of critical OSS/BSS and then expanding the integration to other OSS/BSS.
- Swap one OSS/BSS for another transparently since the specific APIs to interact with the final system are abstracted by the adaptation layer.
- Adapt WFO to the specific needs of the OSS/BSS.
- Handle OSS/BSS-specific authentication and authorisation if needed.

While being a generic framework, adaptable to many situations, it is worth mentioning that WFO is immediately actionable also thanks to the knowledge-sharing initiatives that are active in the community.

As an example, the following is a non-exhaustive list of platforms that have been successfully integrated with WFO in the community:

- NSO – Network Automation System
- VC4 IMS – Inventory Management System
- Ansible (through lightweight client) – Generic Automation System
- Ansible AWX – Generic Automation System
- NetBox – DCIM/IPAM management
- Jira – Ticketing system
- Infoblox – DDI (DHCP/DNS/IPAM)
- Microsoft Dynamics – CRM

3 Conclusions

While this document does not examine in detail all the moving parts and the design choices behind Workflow Orchestrator, it clearly shows that WFO offers the necessary modularity and flexibility to introduce orchestration and automation for managing networks and more in general complex service platforms. This is especially valid in brownfield situations where automation has been introduced partially, or in specific technical domains only, but works in isolation.

WFO allows an organisation to translate business processes and operational practices into automated actions that ensure predictability in terms of results and quality of execution of changes. Moreover, WFO ensures that all OSS/BSS systems are in sync and that data are validated and sanitised at every step of the process.

The growing interest in and adoption of WFO in the European and international R&E community show that this platform has the potential for longevity, maintainability, and expandability and thus is a natural choice for the automation and orchestration of R&E networks.

Glossary

API	Application Programming Interface
BSS	Business supports systems
CRM	Customer Relationship Management
DCIM	Data Centre Infrastructure Management
DDI	DHCP/DNS/IPAM
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
GUI	Graphical User Interface
IMS	Inventory Management System
IP	Internet Protocol
IPAM	IP Address Management
MPLS	Multi-protocol label switching
NFV	Network Function Virtualisation
NMS	Network Management System
NREN	National Research and Education Network
NSO	Network Service Orchestrator
OSS	Operations support systems
WFO	Workflow Orchestrator