24-10-2017

Deliverable D7.2
Systems and Processes Architecture for e-Infrastructure Integration

Deliverable D7.2

Contractual Date:	30-04-2017
Actual Date:	24-10-2017
Grant Agreement No.:	731122
Work Package/Activity:	7/JRA1>
Task Item:	Task 3
Nature of Deliverable:	R (Report)
Dissemination Level:	PU (Public)
Lead Partner:	HEAnet
Document ID:	GN4-2-17-10F9B6
Authors:	A. Sevasti (GRNET), G. Malone (HEAnet), A. Karaliotas (GRNET)

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 731122 (GN4-2).

Abstract
This document provides an account of the work undertaken by JRA1 T3 in GN4-2 Period 1 towards delivering an orchestration framework for e-infrastructure integration, with particular reference to a proof of concept focused on order management and service delivery for MS Azure ExpressRoute connectivity services.

Table of Contents

Table of Figures

Table of Tables

Executive Summary

The primary goal of Joint Research Activity 1 Network Infrastructure Evolution, Task 3 Integration with Other e-Infrastructures (JRA1 T3) is to deliver a reference systems architecture and framework for operational integration and service delivery orchestration across e-infrastructures and commercial service providers.

To test and demonstrate the initial framework and integration techniques being modelled, a proof of concept (PoC) is being undertaken. While inter-provider integration spans a wide variety of processes and interactions, from onboarding a partnership agreement all the way to in-flight end-to-end service assurance, order management and service fulfilment have been selected as the focus for the PoC work to date; specifically, for a bundled offering from GÉANT/NRENs (as the network connectivity providers) and Microsoft (MS) Azure (as the cloud provider) in the form of Azure ExpressRoute connectivity services. The particular challenges presented by this (production) service as currently delivered include its preconditions; the degree of manual effort involved; and the imperative for rapid provisioning of connectivity, with minimal lead times.

The PoC addresses a real requirement: it is anticipated the GÉANT's Infrastructure as a Service (IaaS) framework is likely to create a volume of service requests across the GÉANT/NREN footprint that it will be impractical to handle with the (manual) processes currently in place.

A standards-based approach has been followed, in order to ensure the framework's widest possible acceptance and interoperability. The key standards utilised include TM Forum's Frameworx and Open API specifications and Metro Ethernet Forum's Lifecycle Service Orchestration (LSO), while OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) is deemed as potentially relevant. Acknowledging the potential difficulties of persuading large commercial organisations and other e-infrastructures to adhere to the framework precisely, it has been designed to be flexible enough that while being fully standards' compliant itself, it allows communications with parties that are not.

Providing an orchestration solution for e-infrastructure / service provider integration requires a multi-dimensional approach. The providers involved need to adhere to a common basic set of information-modelling principles and entities, functional elements, process definitions and inter-provider APIs and at the same time share product specifications that map to their internal implementations of product-underpinning services. The proposed orchestration framework addresses all of these perspectives.

The main principles informing the proposed orchestration framework include:

- Centralised order management, by exposing a self-service portal (SSP) as a one-stop shop to the user. Although “centralised” for the PoC, the proposed orchestration framework is designed such that any participating domain/partner in the GÉANT Cloud Connectivity (GCC) delivery chain can trigger an orchestration instance or undertake the orchestration role in future more advanced deployments of the solution. Equally, it does not limit future possibilities for distributed orchestration.
- Coordination of all business and operational interactions between involved parties, through the invocation of open, commonly agreed standards-based APIs that each party should expose.
- Process logic decoupled from system integration logic. Process logic is undertaken by the Activiti Business Process Model and Notation (BPMN) solution; integration logic is handled by Apache Camel.
- For the purposes of the PoC, the GCC Orchestrator component does not confine itself to the role of issuing and managing Service Orders across partners; it also undertakes the role of preparing and issuing Service Configuration and Activation requests to the underlying partner domains, in an effort to offload complexity from the underlying domains to itself and thus preserve the principle that underlying domains expose the minimum functionality and state. Further decoupling will be introduced in subsequent versions of the solution.

For the purposes of the PoC, two high-level processes have been modelled and are supported: order qualification process and order fulfilment process. Further processes, such as partner onboarding, incident management, services decommissioning or maintenance management, as well as service assurance, can be added to the framework progressively.

From its initial focus on MS Azure ExpressRoute connectivity service, the reference implementation of the orchestration framework is, by design, sufficiently flexible and extensible to apply to a wide range of e-infrastructures and service providers, and to further aspects of integration, to the benefit of end users and service providers alike.

1 Introduction

The primary goal of Joint Research Activity 1 Network Infrastructure Evolution, Task 3 Integration with Other e-Infrastructures (JRA1 T3) is to deliver a reference systems architecture and framework for operational integration and service delivery orchestration across e-infrastructures and commercial service providers. In order to achieve this, the following approach is being taken:

- Minimum requirements for the operations and business support systems (OSS/BSS) of participating service providers are defined.
- Orchestrated processes across service providers (e.g. order qualification, order fulfilment, service activation) are defined.
- Inter-service-provider application programming interfaces (APIs) for the different types of service provider interactions (e.g. business agreement establishment, order management, service delivery) are adopted.
- A proof of concept (PoC) to showcase the approach across GÉANT, NRENs and an indicative commercial cloud service provider offering is designed and delivered.

The work carried out is heavily influenced by and largely compliant with relevant standards and best-practices, namely the TM Forum (TMF) Information Framework (SID) and Open APIs as well as the Metro Ethernet Forum (MEF) Lifecycle Service Orchestration specifications, which ties in with the ongoing collaboration between TMF and MEF for convergence of specifications for service orchestration across multiple providers.

To demonstrate the methodology and further develop it, the Task is working on a reference implementation or PoC, which will manage the service delivery of layer 2 (L2) network connectivity between an NREN-connected institution and a commercial cloud service provider (CSP) via GÉANT and other intermediate network providers. More specifically, Microsoft (MS) Azure ExpressRoute offers private connections between MS data centres and institution premises (or colocation points), in collaboration with independent connectivity providers[1]. Based on the relevant GÉANT-Microsoft agreement (see Section 3.1), GÉANT and NRENs can act as connectivity providers for ExpressRoute delivery between MS Azure data centres and institutions, thus offering the ExpressRoute connectivity services.

Currently, the delivery of MS Azure ExpressRoute connectivity services via GÉANT/NRENs is defined by a set of manual processes with multiple parties involved [MSAzureER-GN1, MSAzureER-GN2, MSAzureER-GN3]. It exploits L2 network connectivity solutions already available from / offered by the participating network service providers (individual NRENs and GÉANT), namely Bandwidth on Demand (BoD) and Multi-Domain Virtual Private Networks (MDVPN). Of course, any other type or

[1] For a list of the commercial connectivity providers for MS Azure ExpressRoute, please refer to [MSAzureER].

flavour of L2 network connectivity solution compatible with ExpressRoute connectivity specifications can potentially be integrated in the future.

For now, the PoC will leverage, via the proposed orchestration framework, BoD and MDVPN capabilities in the GÉANT/NREN domains involved to demonstrate coordinated order management and service delivery. Orchestration will invoke automation *where available* and manual tasks where automation is not an option. The ultimate goal is to streamline service delivery, by minimising lead times as much as possible, providing to the user:

- A one-stop-shop experience for end-to-end cloud connectivity.
- Visibility and transparency regarding the state of his service orders as they are handled by all the providers involved.

Although the PoC presented in this document is focused on the particular case of L2 connectivity in the context of MS Azure ExpressRoute, the framework is designed so that it can be applied "as is" to any other case of orchestrated delivery of connectivity services between an R&E institution and a cloud provider participating in the GÉANT Infrastructure as a Service (IaaS) framework, utilising the intermediate network domains, thus in the general context of GÉANT Cloud Connectivity (GCC) services. It can also be applied to the delivery of connectivity services between R&E institutions and other (not necessarily cloud) e-infrastructure providers and IaaS/Platform as a Service (PaaS) solutions such as INDIGO DataCloud [INDIGODataCloud], EGI FedCloud [EGIFedCloud] and European Open Science Cloud (EOSC) [EOSC] service providers.

In the longer term, the framework can be extended to support orchestrated delivery of other types of services across the GÉANT/NREN fabric and the other e-infrastructures.

1.1 In this Document

This document provides an account of JRA1 T3's work towards delivering an orchestration framework and PoC, and is structured as follows:

- Section 2, Relevant Standards, covers TM Forum's Frameworx and Open API suite, Metro Ethernet Forum's Lifecycle Service Orchestration (LSO) and OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA).
- Section 3, Orchestration Problem Statement: The Case of GÉANT Cloud Connectivity Services, discusses the selection of MS Azure ExpressRoute for the PoC, GÉANT/NRENs as the connectivity providers for the service, and key aspects of orchestration and of modelling technical capabilities.
- Section 4, Orchestration Framework Overview, covers the integration aspects addressed by the orchestration framework, including information/entity-modelling principles, functional architecture, processes and APIs. This section also presents the product and service specifications modelled for the PoC, and orchestrated workflows for BoD and MDVPN.

- Section 5, Orchestration with Non-GÉANT/NREN Service Providers, considers the options available for accommodating parties that are not fully TMF-compliant, with particular reference to INDIGO DataCloud.
- Section 6, Conclusions, offers an overall assessment of the orchestration framework effort to date.

2 Relevant Standards

The goal is to deliver a solution that spans service provider borders and can be adopted regardless of their internal processes, systems and services. Therefore, a standards-based approach, which has been adopted by a vast majority of communications service providers and has matured over the years in operational environments, has been followed, in order to ensure the widest possible acceptance and interoperability.

Regarding business process modelling, orchestration, OSS/BSS software architecture, information modelling and functional specifications, the TM Forum Frameworx has been followed. This is a choice endorsed by several other Tasks throughout the GN4-2 project, namely:

- JRA2 Network Services Development, Task 2 Service Provider Architecture (JRA2 T2), focusing on a TMF Frameworx-compliant architecture and prototype for the OSS/BSS software stack of a single service provider (Service Provider Architecture (SPA)).
- JRA2 Task 4 Network and Service Monitoring (JRA2 T4), focusing on network and service monitoring solutions that adhere to TMF practices for service quality monitoring and TMF APIs.
- Service Activity 2 Trust & Identity and Multi-Domain Services (SA2) work on service operational processes following the TMF Business Process Framework (eTOM).

With regard to orchestrating end-to-end Ethernet services connectivity in particular, the recent developments within the MEF, especially the Lifecycle Service Orchestration (LSO) specifications, are of the utmost relevance to the GÉANT Cloud Connectivity (GCC) PoC.

Furthermore, OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) [TOSCA-NFV] is being introduced as an increasingly adopted modelling and deployment framework in cloud environments.

The following sections highlight the TMF Frameworx and Open API specifications as well as the MEF LSO and OASIS TOSCA principles relevant to the proposed framework for operational integration and service delivery orchestration.

The framework design has followed the specifications lightly (i.e. as/when required) in order to achieve adherence to standards without inheriting more complexity than necessary. In the case of TMF and MEF specifications, it needs to be mentioned that detailed specifications of the relevant standards are accessible by TMF and MEF members only. Therefore, this document does not provide a detailed transcript of the differences between the proposed framework and the original TMF and MEF specifications.

2.1 TM Forum's Frameworx

The TMF Frameworx model [TMFFrameworx] is a set of abstract best practices and standards to describe service-oriented process interactions with a view to overall management and orchestration of those processes.

The suite encompasses four main domains:

- Business Process Framework (formerly known as eTOM): a hierarchical catalogue of the key business processes required to run a service-focused business. At the conceptual level, the framework has three major areas, reflecting major focuses within typical enterprises:
  - Strategy, infrastructure and product.
  - Operations.
  - Enterprise management.

  Operations is the area dealing directly with service delivery (fulfilment) and monitoring (assurance), while at the same time introducing all processes required to manage users/customers, services, resources and suppliers/partners in a service provider environment.
- Information Framework (SID): a reference model and common vocabulary for all the information required to implement Business Process Framework (eTOM) processes. It reduces complexity in service and system integration, development and design by providing an off-the-shelf information model that can be quickly adopted by all parties and modelling all entities involved in business and operational interactions.
- Application Framework (TAM): describes the software architecture for service-oriented delivery, namely the software components division and capabilities providing common software application ground.
- Integration Framework: brings everything together. When used in conjunction with the other frameworks and reference architectures, it allows digital service providers to identify the key integration points in their architectures, and define them in terms of widely adopted industry-standard APIs.

2.2 TM Forum's Open API Suite

TMF's Open API suite is a set of standard representational state transfer (REST)-based APIs enabling integration among operations and management systems for creating and operating complex services. The suite [TMFOpenAPIS] is endorsed and used by several communications service providers (who have agreed to position the specifications as a preferred requirement in their IT RFPs from January 2017 onwards) and technology partners. It brings together different stakeholders from across industries to collaborate and build key partnerships to create the APIs and connections. The TM Forum REST-based APIs are technology agnostic and can be used in any digital service scenario, including business-to-business value fabrics, network functions virtualisation (NFV), next-generation OSS/BSS and more.

2.3 Metro Ethernet Forum LSO

MEF has defined the term “Third Network” to reflect the evolution and transformation of network connectivity services and the networks used to deliver them. “Third Network” services in the context of MEF are services that follow the Carrier Ethernet 2.0 (CE 2.0) specifications, including the E-Line, E-LAN, and E-Tree service types. Third Network services can be controlled by the users via software and can be enabled not only between physical (Ethernet) ports but also server interfaces to provide access to virtual machines (VMs) within the cloud. MEF is defining within the Lifecycle Service Orchestration (LSO) specifications the requirements and APIs for service ordering, fulfilment, performance, usage, analytics and security across multi-operator networks.

Orchestrating the lifecycle of Carrier Ethernet services delivery across providers, as specified by the LSO, provides a strong basis for the design of the orchestration framework of L2 Ethernet-based VPN services across network service providers in the context of GÉANT. A particular strength is that MEF LSO is largely compatible with the TMF specifications in several dimensions, namely definition of entities such as products, services, resources, lifecycle of entities, APIs, processes (e.g. fulfilment, control, assurance, billing) [MEF-55].

LSO defines a reference architecture (shown in Figure 2.1) that identifies the roles of the participating entities as well as the functional entities that enable LSO capabilities. It also identifies the logical points of interaction between functional entities (management interface reference points (MIRPs)), in the form of interface profiles implemented by APIs.

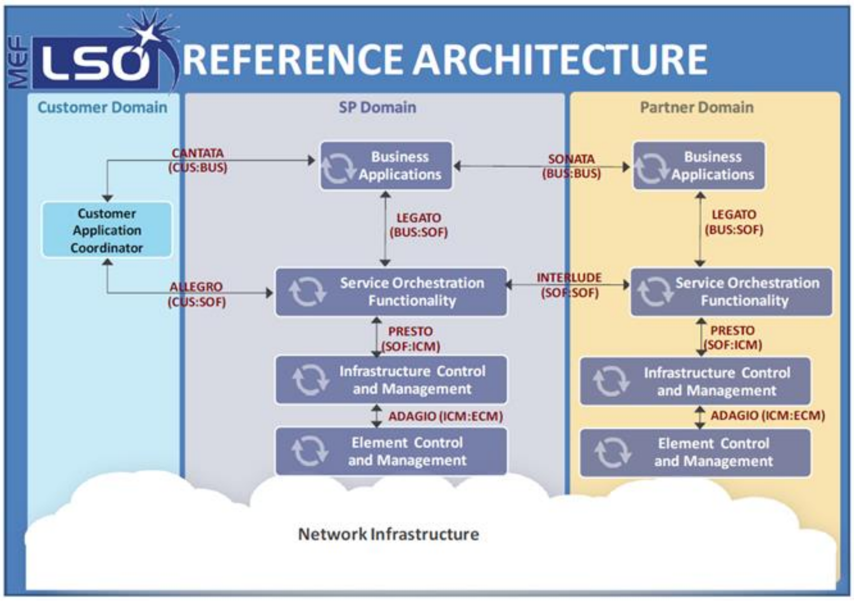

Figure 2.1: MEF LSO reference architecture

As briefly shown in the MEF LSO architecture (Figure 2.1), the MIRPs directly relevant to the GÉANT requirements are SONATA (BUS:BUS) and INTERLUDE (SOF:SOF).

- SONATA (BUS:BUS) [SONATA]: defines the management reference point supporting the management and operations interactions (e.g., serviceability, ordering, billing, trouble ticketing, etc.) between two network providers (e.g. a Service Provider and a Partner Domain). In other words, LSO SONATA supports non-control cross-domain interactions between the Service Provider's business applications (BUS) and the Partner's business applications.

  LSO SONATA connects between the BSS functions of the Service Provider and Partner domains respectively. Example interactions include:

  - Service Provider browses the Partner's product catalogue (e.g. wholesale catalogue) for Product Offerings that are available for the Service Provider to select. This may include some geographical and service information to support availability queries of a Product Offering in a specific geographical area.
  - Service Provider develops (based on Product Offerings), places, tracks and exchanges Product Orders with the Partner.
  - Service Provider requests modification of Product Instances.
  - Service Provider receives Product Instance performance and fault information provided by the Partner.
  - Service Provider receives information from the Partner about the scheduled maintenance that may impact their Product Instances.
  - Service Provider places and tracks trouble reports.
  - Service Provider exchanges usage and billing information.

  SONATA can be used, for example, to place an order to a Partner provider for an access service that is needed as a part of an end-to-end Connectivity Service.

- INTERLUDE (SOF:SOF) [INTERLUDE]: defines the management reference point that provides for the supervision of a portion of LSO services within the Partner Domain that is coordinated by a Service Provider LSO. In other words, LSO INTERLUDE supports control-related management interactions between the Service Provider and the Partner. LSO INTERLUDE connects between the OSS functions of the Service Provider and Partner domains respectively. Example interactions include:
  - Service Provider controls aspects of the Service within the Partner Domain (on behalf of the Customer) by requesting changes to dynamic parameters as permitted by Service policies.
  - Service Provider queries operational state of the Service.
  - Service Provider requests change to administrative state of a service or service component (e.g. Service Interface).
  - Service Provider requests update to defaulted Service parameters that are permitted to be customised (policy-controlled).
  - Service Provider requests creation of connectivity between two Service Interfaces as permitted by established business arrangement.
  - Service Provider queries the Partner's Service Inventory for services provided by the Partner to the Service Provider.
  - Service Provider receives Service-specific event notifications from the Partner.

- ○ Service Provider receives Service-specific performance information from the Partner.
- ○ Service Provider requests test initiation and receives test results from the Partner.

INTERLUDE can be used, for example, to request changes to a CE-VLAN ID mapping at a user network interface (UNI) that resides in a Partner Domain.

2.4 OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA)

OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) [TOSCA-TC] allows the applications deployable on multiple clouds to be described, including application servers or VMs, network components, their interconnections and corresponding deployment and configuration workflows.

TOSCA provides mechanisms to control workflows, describe relationships and reflect dependencies between resources. TOSCA defines a service template, where a “service”, for TOSCA, is an application or application component, such as a database server, firewall or virtual router. In a networking context, the term “service” implies the customer-specific configuration of a network function. Examples would be an individual layer 3 VPN router or firewall configured with the customer-specific rules. OASIS defines a special TOSCA Simple Profile for Network Functions Virtualisation (NFV) [TOSCA-NFV] that can be used for network service descriptor (NSD) and NFV descriptor (NFVD) specifications. NSDs/NFVDs can be of particular interest for modelling network connectivity requirements between GÉANT/NRENs and cloud providers, for example in the context of orchestrated L2 network connectivity delivery as addressed in this document.

TOSCA-based service descriptions and orchestration are defined in Cloud Service Archive (CSAR) files. TOSCA can be naturally combined with YANG modelling [IETF-RFC6020] and the Network Configuration Protocol (NETCONF) [IETF-RFC6241] so that TOSCA models the application, network topology and deployment of the network elements or functions and YANG models the actual network nodes/functions and their configuration that can be included in TOSCA template.

The adoption of TOSCA, both by cloud applications developers and the network community, is growing. The latter is especially stimulated by the growing adoption of network virtualisation.

As current standards in MEF, the European Telecommunications Standards Institute (ETSI), TMF, Organisation for the Advancement of Structured Information Standards (OASIS) and Internet Engineering Task Force (IETF) are evolving, TOSCA is growing in popularity and is expected to become an important component for VNF and SDN service delivery description and orchestrating (see for example *IG1141 Procurement and Onboarding of Virtualisation Packages R16.5.1 Standard* [TMF-R16-5-1] from TM Forum).

3 Orchestration Problem Statement: The Case of GÉANT Cloud Connectivity Services

In order to succeed in defining a systems and processes architecture for e-infrastructure/service provider integration, it was first necessary to analyse sample processes and system workflows for how GÉANT/NRENs interact with other service providers and e-infrastructures today.

The Task initially focused on identifying the requirements for business and operational interactions between GÉANT/NRENs/institutions and data centres / cloud service providers, as it was verified with JRA4 Application Services Delivery Development that the latest developments regarding the GÉANT Infrastructure as a Service (IaaS) framework [GNInfraSvcs] are likely to create a volume of service requests across the GÉANT/NREN footprint that it will be impractical to handle with the (manual) processes currently in place. The majority of such service requests are expected not to require specialised network connectivity handling as they will exploit public IP connectivity to the cloud providers. However, a number of the business interactions (e.g. establishment of agreements between institutions and cloud providers) and operational interactions (e.g. coordinated troubleshooting of IP connectivity) are still expected to require provider interoperation. Thus, inter-provider integration spans a wide variety of processes and interactions, from onboarding a partnership agreement all the way to in-flight end-to-end service assurance. Out of this wide scope, order management and service fulfilment have been the focus to date for the JRA1 T3 work presented in this document.

More specifically, analysis of existing processes and system workflows has focused on the – currently in production – order management and service delivery processes for a bundled offering from GÉANT/NRENs (as the network connectivity providers) and MS Azure (as the cloud provider) in the form of Azure ExpressRoute connectivity services. These were reviewed and then modelled to orchestrated processes using the open source Business Process Model and Notation (BPMN) tool Activiti [Activiti]. This modelling and visualisation of the processes currently in place made it easier to identify blocking sub-processes that could be automated, as well as interactions that could be orchestrated in order to provide a seamless experience to the end user. This in turn helped to rework and optimise the processes to adhere to the orchestration framework and thus address the business and operational integration requirements.

The following sections discuss the selection of MS Azure ExpressRoute, GÉANT/NRENs as the connectivity providers for the service, and key aspects of orchestration and of modelling technical capabilities.

3.1 Microsoft Azure ExpressRoute

An important early task was to choose a service provider, external to GÉANT, that could be used to test and demonstrate the initial framework and integration techniques being modelled. This was both to ensure practical application of the techniques and to provide insight into where the framework could be improved.

As part of the previous work undertaken by GN4-2 JRA4 Application Services Delivery Development, GÉANT has signed a partner agreement with Microsoft to act as a network connectivity provider for Microsoft's Azure ExpressRoute service. ExpressRoute allows an end-user institution to peer directly with private resources deployed in the MS Azure Cloud Platform as well as public resources and/or Office365 resources. The service is delivered over a controlled layer 3 link, using private L2VPN connections from Microsoft (ExpressRoute), GÉANT, the local NREN and the institution's local network, stitched together in an end-to-end L2VPN.

To capitalise on this work, and with applicable use cases from participating NRENs, the Microsoft Azure ExpressRoute service was chosen as the initial test product offering for analysis and orchestration. In addition, Microsoft's membership of TMF and the feature-rich and well-documented set of APIs able to support the services required also influenced the decision.

3.2 GÉANT/NRENs as the Connectivity Providers of ExpressRoute

At the moment of writing, GÉANT, as the pan-European R&E backbone, provides direct connectivity to Microsoft Azure facilities via Amsterdam (through NetherLight) and via London (longlined through NetherLight) as shown in Figure 3.1. Via these connections, redundant services can be provided to all NRENs and their connected institutions in Europe towards MS Azure.

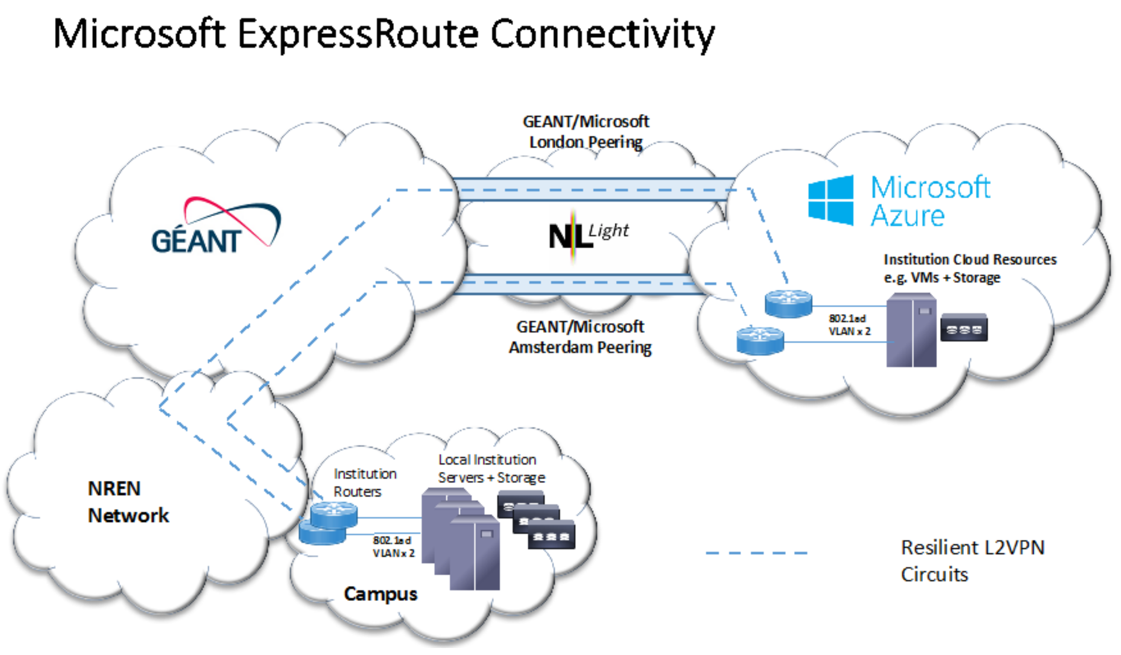
Microsoft ExpressRoute Connectivity
GEANT/Microsoft London Peering
GÉANT
NL Light
Microsoft Azure
Institution Cloud Resources e.g. VMs + Storage
802.1ad VLAN x 2
GEANT/Microsoft Amsterdam Peering
NREN Network
Institution Routers
Local Institution Servers + Storage
802.1ad VLAN x 2
Campus
Resilient L2VPN Circuits

Figure 3.1: MS ExpressRoute connectivity

When a user places an order with MS through the Azure portal and chooses GÉANT as the connectivity partner, Microsoft will configure two 802.1ad VLAN tagged circuits up to the predefined NetherLight handover points.

Microsoft will define the outer tag (S-VLAN) towards NetherLight, and the user (in this case, the institution) is free to choose the inner tag (C-VLAN) – or have it automatically assigned[2].

From that point onwards, a specific case for ExpressRoute connectivity service delivery across the involved service providers is depicted in Figure 3.2. In this case, MS Azure ExpressRoute traffic exiting NetherLight traverses GÉANT, then the home NREN network of the institution before reaching the campus infrastructure of the institution and being delivered to end points. Each of the service providers uses their local technology/service offerings to deliver their part of the end-to-end L2VPN connection (including MDVPN, BoD and manual L2VPN configuration). Thus, multiple business parties are involved in service delivery and at the same time the different parties use different processes and different technologies to deliver the service. Communication complexity among the parties, and also the need for reconciliation/negotiation of technical parameters at intermediate service delivery points, are obvious.

[2] For special cases of handling the tagged traffic at the border of GÉANT, please refer to Appendix A.

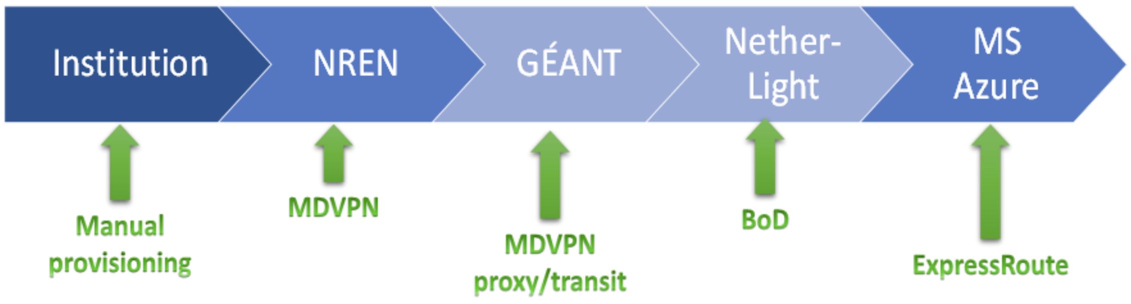

Figure 3.2: Specific case for ExpressRoute connectivity

3.3 Orchestration

Currently, a number of steps are required, involving all related parties, in order to complete an end-to-end ExpressRoute setup successfully.

Firstly, before ordering, some preconditions must be met:

- The user must have a valid Microsoft account with an active Azure subscription.
- The user's institution must have connectivity to an NREN that is connected to GÉANT and has countersigned the ExpressRoute Connectivity Services partner agreement.
- An agreement needs to pre-exist between the institution and the local NREN/regional network for the latter to serve MS Azure cloud traffic of the former.

It has to be noted that fulfilling these preconditions and performing these technology-specific checks are currently done manually, thus adding extra lead time to the ordering process when it is executed between a certain set of parties for the first time.

If these preconditions are satisfied then the user can proceed with ordering the ExpressRoute service via GÉANT by accessing the Microsoft Azure portal directly and choosing GÉANT as the connectivity provider. It is important to note that Microsoft starts billing for the ExpressRoute connectivity at the moment the user submits his MS Azure ExpressRoute order to the MS Azure portal. At that moment the user receives a service instance key (S-key), which he can use to refer to the MS Azure side of the service offering.

Once the order has been placed via the MS Azure portal, the involved network providers and the institution carry out their part of network connectivity service configuration and delivery.

Table 3.1 below shows the overall service delivery schedule that GÉANT has committed to, with steps 1 and 3 belonging to the business domain, while steps 2, 4, 5 and 6 belong to the operational domain of interactions between the involved parties/providers.

Step	Description	Domain	Responsible Party	Expected Time
1	Check availability at your local NREN (optional but recommended)[3].	Business	Institution	1 week
2	Request S-key from Microsoft Azure portal.	Operational	Institution	1 hour
3	Fill out the application form and send it to your NREN.	Business	Institution	1 hour
4	Determine request and configure end-to-end service.	Operational	NREN & GÉANT	1 week
5	Configure layer3 connectivity with Microsoft.	Operational	Institution	1 day
6	Test connectivity with the Azure platform.	Operational	Institution	1 day

Table 3.1: ExpressRoute service delivery schedule

Due to immediate billing for the user, and also the "click and provision" nature of obtaining Azure VMs, it is of paramount importance to provide a similar experience to the user regarding connectivity in the case of ExpressRoute, as well as shorter lead times than those presented in Table 3.1.

3.3.1 Example of Modelling Interactions

Focusing on the end-to-end service provisioning/fulfilment step of the current delivery schedule (step #4 in Table 3.1), and refining it with a view to identifying each business party interaction, the Task was able to model a basic workflow in the Activiti BPMN system as a basis for the orchestration solution design. Figure 3.3 depicts a snapshot of the modelling exercise, relating to executing in parallel and confirming (or not) the steps required for service provisioning across the involved parties, assuming those to be the Network Operating Centre (NOC) of each provider (institution NOC, NREN NOC and GÉANT NOC) – allowing scope for defining more complex workflows in further iterations. Clearly, the process modelling was a very helpful exercise for the team as it revealed business, engineering and software requirements for the orchestration framework.

[3] This first check is performed to see whether the preconditions for quickly setting up the connection have been met. If the conditions have been met, the Microsoft S-key can be requested.

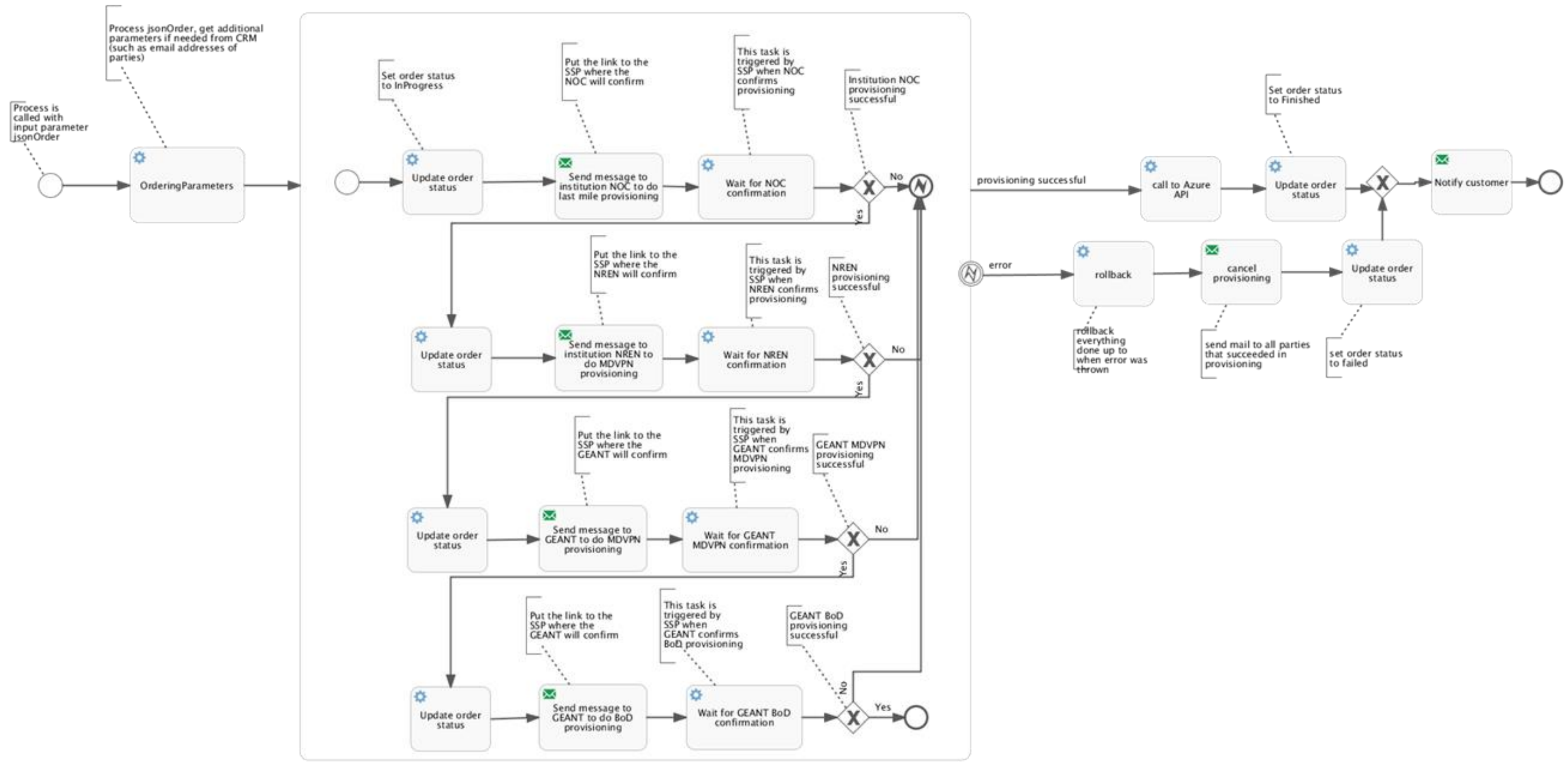
Process is called with input parameter jsonOrder
Process jsonOrder, get additional parameters if needed from CRM (such as email addresses of parties)
OrderingParameters
Set order status to InProgress
Update order status
Put the link to the SSP where the NOC will confirm
Send message to institution NOC to do last mile provisioning
This task is triggered by SSP when NOC confirms provisioning
Wait for NOC confirmation
Institution NOC provisioning successful
No
Yes
Put the link to the SSP where the NREN will confirm
Update order status
Send message to institution NREN to do MDVPN provisioning
This task is triggered by SSP when NREN confirms provisioning
Wait for NREN confirmation
NREN provisioning successful
No
Yes
Put the link to the SSP where the GEANT will confirm
Update order status
Send message to GEANT to do MDVPN provisioning
This task is triggered by SSP when GEANT confirms MDVPN provisioning
Wait for GEANT MDVPN confirmation
GEANT MDVPN provisioning successful
No
Yes
Put the link to the SSP where the GEANT will confirm
Update order status
Send message to GEANT to do BoD provisioning
This task is triggered by SSP when GEANT confirms BoD provisioning
Wait for GEANT BoD confirmation
GEANT BoD provisioning successful
No
Yes
provisioning successful
call to Azure API
Set order status to Finished
Update order status
Notify customer
error
rollback
rollback everything done up to when error was thrown
cancel provisioning
send mail to all parties that succeeded in provisioning
Update order status
set order status to failed

Figure 3.3: Workflow model of service provisioning step

3.3.2 Automating Business and Operational Interactions

As mentioned above, the current process for ordering ExpressRoute connectivity services is heavily manual, requiring the completion of multiple forms and emailing of details, as well as waiting for other parties to respond to requests in an asynchronous manner.

Based on the analysis carried out by the Task, the process steps to be automated in the **business interactions domain** are:

- Pulling user, institution, and upstream NREN information via the user's federated login details.
- Investigating availability of the Azure ExpressRoute connectivity service offering for the institution from the upstream NREN.
- Retrieving Azure subscription details directly via the Microsoft Azure API using the user-entered S-key.
- Automatically notifying NREN and GÉANT NOCs of the submitted order for GÉANT Cloud Connectivity (GCC) Azure by the user.

However, **operations integration** is also a requirement. Further to identifying and notifying the involved parties, the technical effort to configure the services is considerable, especially taking into account technical parameters that span multiple partner domains, for example, in the case of the MDVPN fabric, where edge devices at different domains need to be configured in coordination. Beyond automatically emailing the involved parties with details of the required resource configuration and service activation requests, which will always be an option, the orchestration solution could trigger and coordinate automation capabilities in the participating domains. For example:

- Utilise Network Service Interface (NSI) to automatically provision BoD circuits.
- Push MDVPN Proxy configlets to domains via the Junos Space [JunosSpace] APIs (in the case of GÉANT) or in the form of Ansible [Ansible] playbooks (for example).

3.3.3 Conceptual Overview of Orchestration

Based on the MS Azure ExpressRoute case as presented above, Figure 3.4 below provides an overview of how an orchestrator system would coordinate service delivery between a user and the involved service providers.

The main principles informing the proposed orchestration framework are as follows:

- Centralised order management, by exposing a self-service portal (SSP) as a one-stop shop to the user.
- Coordination of all business and operational interactions between involved parties, through the invocation of open, commonly agreed standards-based APIs that each party should expose.

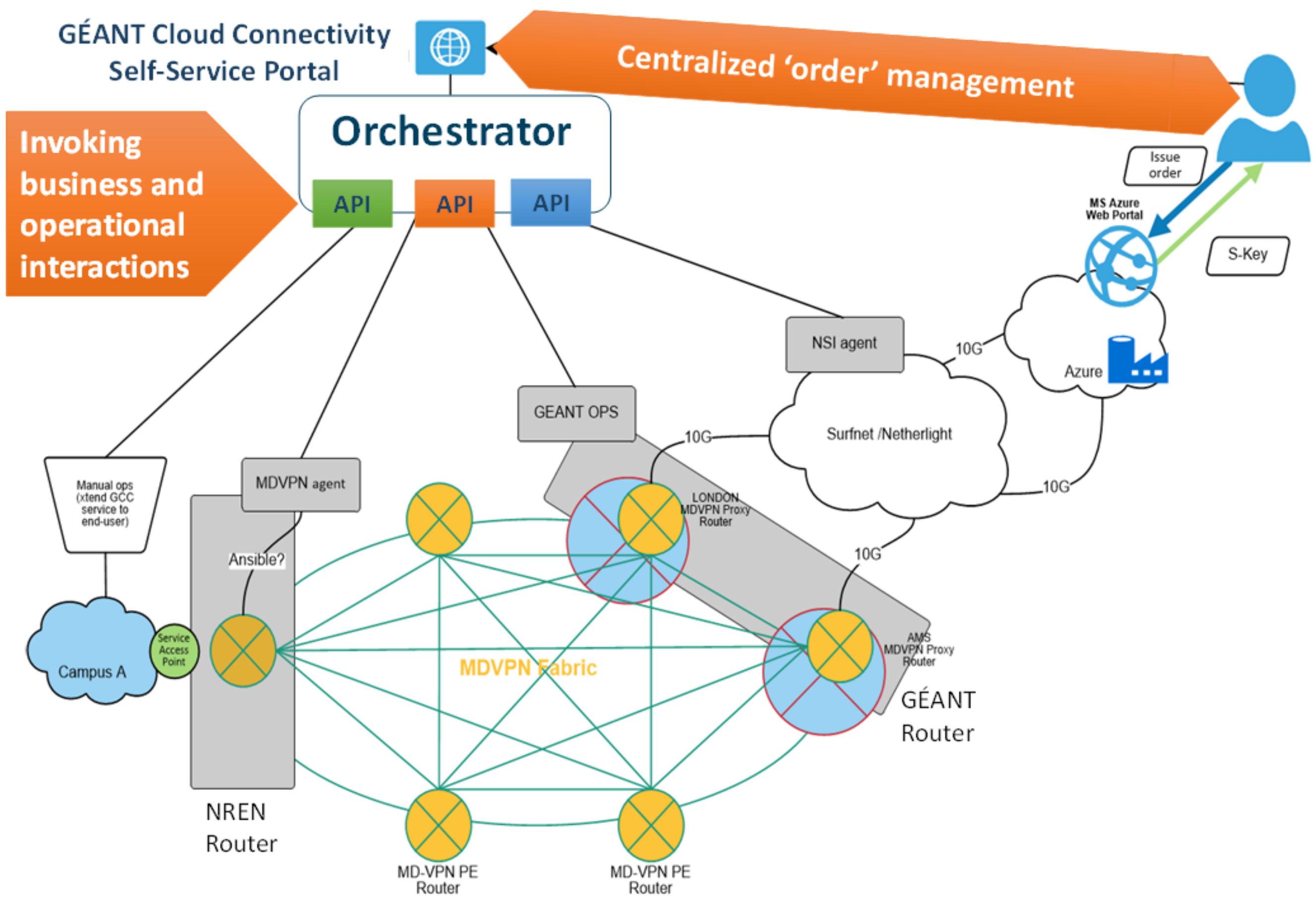

Figure 3.4: Overview of service delivery orchestration

Table 3.2 below describes the orchestrated steps in more detail.

Step	Responsible Party/System	Description
1	User	Places an ExpressRoute connectivity service order directly with Microsoft using the Azure portal and receives an S-key and confirmation of service creation within Azure.
2	User	Logs in to the GCC self-service portal (SSP) and submits an end-to-end order for the cloud connectivity service using the provided S-key to link it with the MS Azure ExpressRoute provisioned service instance (making this, in fact, an integrated two-stop-shop case).
3	GCC Orchestrator	Through federated authentication and authorisation infrastructure (AAI) services in GÉANT/NRENs, programmatically determines the user, his institution and corresponding NREN by the user's AAI login credentials.
4	SSP	Executes some basic validation of the order and, using the provided S-key, retrieves the ExpressRoute connection parameters from Microsoft using the Azure APIs. Passes all these details to the GCC Orchestrator for order qualification and fulfilment.

Step	Responsible Party/System	Description
5	GCC Orchestrator	Uses appropriate inter-provider APIs to: • Check which network connectivity services (e.g. MDVPN, BoD) are available in GÉANT/the NREN/institution involved in the order. • Determine the chain of involved providers (or provider "path") to deliver the service using information about business agreements between the providers exposed by each provider via the APIs. • Invoke OSS/BSS interactions through the APIs to signal each provider domain to configure their elements for service provisioning and carry out any technology translations/stitching required. Where automation is available (e.g. in BoD domains by invoking NSI end points, or in MDVPN domains by committing configuration playbooks), operations can be completed synchronously; where manual steps are required (e.g. by campus NOCs), the orchestrator coordinates asynchronous interactions.
6	GCC Orchestrator	Records the outcome of the service activation workflow across the providers and reports it back to the SSP and user.

Table 3.2: Orchestrated service delivery steps

It needs to be noted that, although Figure 3.4 shows a centralised orchestration function, the proposed orchestration framework (modelling, functional architecture, processes, APIs, etc.) as presented in Section 4 allows any OSS/BSS stack of any partner in a product-offering partnership agreement to undertake the orchestration role.

In Section 4, the architecture and modelling of the orchestration framework is described in detail, with illustrative references to the specific ExpressRoute connectivity service, as a representative case of GÉANT/NREN and e-infrastructure/cloud service provider (CSP) service delivery integration.

3.4 Modelling Technical Capabilities in the Context of Orchestration

In an orchestration framework across multiple service providers, the technical specifications and special conditions for underlying services (modelled in the form of underpinning services and resources, in accordance with TMF SID) are crucial and require very efficient means of orchestrating operational (i.e. non-business-related) interactions. In order to demonstrate the complexity involved and present the Task's approach to addressing it through the proposed framework, the following sections give an overview, from the orchestration requirements' point of view, of the selected GCC underpinning services (BoD, MDVPN) in the PoC under development.

At this point, it is important to mention that the equivalent work is undertaken in the context of MEF LSO by delivering elements and underpinning service templates based on the CE 2.0 standard [MEF-CE2] for all provider services (e.g. E-Line) involved in LSO service delivery.

3.4.1 Bandwidth on Demand

Bandwidth on Demand (BoD) is a Network Service Interface (NSI)-compatible network connectivity service available on the GÉANT backbone network and in a small number of NREN domains. For the purposes of the PoC, it is crucially also available on the NetherLight exchange. BoD supports fully automated end-to-end Ethernet-based connectivity, including advance network resource reservation between topology locations (modelled as Service Termination Points (STPs) of a provider's infrastructure). The agent representing a domain in terms of BoD capabilities is defined as a Network Service Agent (NSA).

3.4.1.1 *Topology Information and Service Activation*

Network topology and resource information for BoD domains is provided using the NSI Document Distribution Service (DDS). The information includes which domains support NSI, their STPs, and configured VLANs for each STP. For the purposes of orchestration and in order to retrieve this information, the Task has developed a number of Representational State Transfer (REST)-based wrappers [RESTW-nsi_ports, RESTW-nsi_conns, RESTW-GÉANT-bod] to provide easy access to BoD-specific port and connection details via DDS. Via these wrappers, DDS end points can be queried where available (e.g. for NetherLight at [NL-DDS]) and feed the orchestrator with the information required. It is important to mention that the wrappers, especially *nsi_connections*, implement TMF Activation and Configuration Open API v1.0, and so demonstrate how NSI can be wrapped to adhere to a TMF-compliant orchestration environment.

For service activation, BoD domains can interact with the orchestrator individually via their local NSI end points or via NSA Aggregators (AGs). An Aggregator is an NSA that has more than one child NSA, and is responsible for aggregating interactions on behalf of its child NSAs. An NSA Aggregator's DDS can be queried for topology information across many domains. An AG also acts as both a requester and a provider NSA, to service connection requests, perform path finding, and distribute segment requests to other NSAs for processing. This allows the orchestrator to use one single AG to determine topology and submit service requests on behalf of all BoD-capable parties for the purposes of the GCC PoC.

As Aggregators in the global NSI service map contain STP details from many NSI-capable domains (not just GÉANT and connected NRENs), for the purposes of the PoC service availability and topology information can be collected by querying an AG and parsing for:

- GÉANT STPs: `urn:ogf:network:geant.net:2013:topology`
- STPs aggregated under GÉANT (European NRENs): `urn:ogf:network:geant.net:2013:nsa`

As mentioned above, for service activation an Aggregator NSA can act as both a Requester Agent (RA), when it requests a service from another NSA, or a Provider Agent (PA), when it services requests from another NSA, performing path finding, distributing segment requests to child NSAs, etc. This allows a decentralised service where service requests can originate from an RA/PA not associated with either end domain. Both GÉANT and NetherLight provide NSA Aggregators that can be used, with appropriate configuration, for GCC PoC purposes.

Indicatively, using an NSI Requester Agent [NSIRequester] with an AG, a service reservation can be requested between MS Azure ExpressRoute delivery ports in NetherLight and the GÉANT ports in Amsterdam and London using the following parameters:

```
nsi version: 2
description: MS ExpressRoute circuit in AMS
start date: <some date + time in future>
end date: <some date + time in future>
service type: http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-
GOLE
Source STP ID: urn:ogf:network:geant.net:2013:topology:GEANT-Netherlight-
AMS-MS-Express?vlan=1000
ERO STP ID: urn:ogf:network:netherlight.net:2013:production7:geant-
ams?vlan=1000
Destination STP ID:
urn:ogf:network:netherlight.net:2013:production7:microsoft-ams?vlan=1000
Bandwidth: 100
unprotected: yes
path calculation: default


nsi version: 2
description: MS ExpressRoute circuit in LON
start date: <some date + time in future>
end date: <some date + time in future>
service type: http://services.ogf.org/nsi/2013/12/descriptions/EVTS.A-
GOLE
Source STP ID: urn:ogf:network:geant.net:2013:topology:GEANT-Netherlight-
LON-MS-Express?vlan=1000
ERO STP ID: urn:ogf:network:netherlight.net:2013:production7:geant-
lon?vlan=1000
Destination STP ID:
urn:ogf:network:netherlight.net:2013:production7:microsoft-lon?vlan=1000
Bandwidth: 100
unprotected: yes
path calculation: default
```

Creating a new service has three stages: *reservation*, *reserve commit* and *provision*. The above reservation request will result in a *connectionId* that can be used in the next steps of service setup.

3.4.1.2 *Path Diversity*

Azure ExpressRoute is delivered by Microsoft in two diverse end points (denoted by different VLANs) for customer resilience. Ideally, resiliency should be preserved throughout the end-to-end circuit, or at least in some denoted handover point across the providers' chain (see Figure 3.2).

Within an NSI connection request for a point-to-point service there is an optional Explicit Routing Object (ERO) parameter [OGF-GFD-RP212]. The ERO is an ordered list of STPs (both advertised edge and domain-internal hidden STPs) that describe the route that should be taken by the request. The path determination engine will use the STPs listed in the ERO as constraints during the path-finding process. By explicitly setting diverse intermediate EROs in ExpressRoute connectivity service requests towards BoD domains, path diversity within individual NSI segments is ensured.

```
Source STP ID:
urn:ogf:network:geant.net:2013:topology:GEANT-Netherlight-AMS-MS-
Express?vlan=1010
Destination STP ID:
urn:ogf:network:netherlight.net:2013:production7:microsoft-ams?vlan=1010
(via) ERO STP ID:
urn:ogf:network:netherlight.net:2013:production7:geant-ams?vlan=1010
```

3.4.1.3 *VLAN Tagging*

BoD supports VLAN retagging, allowing greater flexibility with regard to service delivery options, depending on what the NREN/client network can support.

For the PoC, Microsoft uses 802.1ad where the inner C-VLAN tag can be chosen by the client but the outer S-VLAN tag is defined by Microsoft and cannot be changed. A retagging action can take place at NetherLight (or somewhere else if required). As Microsoft uses 802.1ad, the institution campus should support that as well, otherwise the NREN will need to strip the S-VLAN tag before delivering the ExpressRoute connectivity service traffic to the campus.

3.4.2 MDVPN

Multi-Domain Virtual Private Network (MDVPN) is a transport fabric based on multi-protocol label switching (MPLS) and border gateway protocol (BGP) to deliver L2VPNs across multiple domains by configuring label switched paths (LSPs) between the two provider edge (PE) routers only. Two service label distribution (SLD) methods can be used: BGP using route reflectors for maximum scalability, or bi-directional label distribution protocol (tLDP) sessions for simplicity.

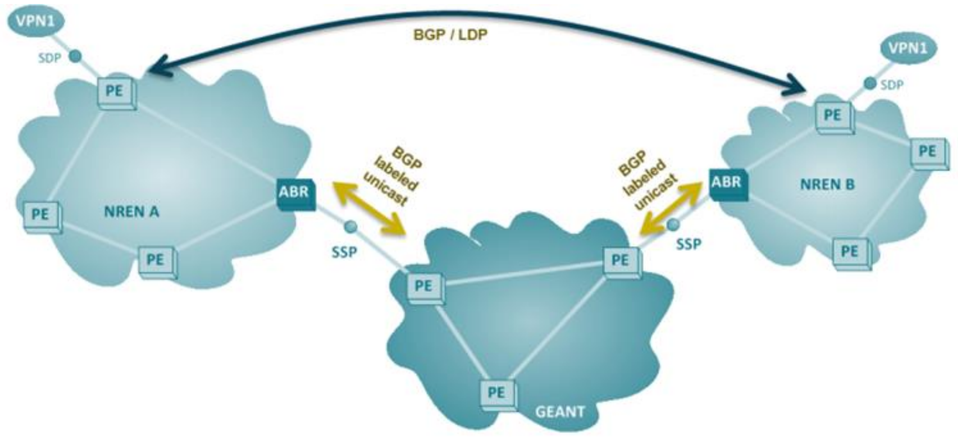

Figure 3.5: L2VPN delivered via MDVPN

3.4.2.1 *Topology Information*

In the MDVPN context, and assuming a fabric that extends to the NREN border router towards an institution, a fundamental component of orchestration is a topology database providing not only technical information on PE devices and their capabilities (available VLANs, translation ability, maximum transmission unit (MTU)) but also corresponding relational detail on which institutions are served by those devices. At the time of writing, MDVPN offers only a simplified MDVPN Register [MDVPNReg] which lists PE lookbacks for circuit end points within NRENs.

Each NREN that delivers an MDVPN-based product offering to its institutions in the context of an orchestrated solution must maintain local topology/service availability per (institution) location information, which is made available via (preferably TMF-compliant) RESTful APIs. (See Section 4.1 for information on how agreements are modelled.)

In this way, each MDVPN domain can expose to the orchestrator information on PE resources (routers) at the edges of its MDVPN fabric and any direct customers (institutions) served on those PEs (modelled as a pre-existing UNI type of agreement between the MDVPN domain, usually NREN, and each institution).

3.4.2.2 *Path Diversity*

MDVPN circuits are just MPLS LSPs so they enjoy the built-in resiliency available on the underlying MPLS mesh network provided by GÉANT. Depending on NREN configuration, these circuits may have single points of failure (SPoFs) at the handover point between GÉANT and NREN or at the end termination points. Since for the ExpressRoute connectivity service two diverse circuits are delivered by Microsoft, ideally these should be terminated on separate PEs within the NREN.

3.4.2.3 *VLAN Tagging*

As MDVPN is a transparent transport technology it does not consider VLAN tagging in any way. A label is applied to data at source and stripped at destination. The benefit of this is that VLANs are invisible to the core; however, the downside is that VLAN retagging, should it be required for some reason, cannot be applied within the core. It can only be applied at the ingress/egress interfaces of the MDVPN fabric, for example in the points where MDVPN L2 circuits are stitched or terminated. A solution, in the form of MDVPN Proxy, does exist for this issue and is discussed in Section 3.4.2.5.

3.4.2.4 *Other Considerations*

Security and Trust

As mentioned above, MDVPN can be configured with different service label distribution methods: BGP using route reflectors for maximum scalability, or bi-directional LDP for simplicity. The PoC will use LDP, which requires PEs to speak directly to one another. Firewall rules need to be changed in order to permit a targeted LDP session (UDP/TCP port 646) between local PE loopbacks and GÉANT hand-off PE to CSPs. In the PoC use case these will be GÉANT routers in London and Amsterdam.

Point-to-point (P2P) L2VPN LDP protection can be applied, for example, by using prefix-lists for PE router filters.[4]

[4] For further information about MDVPN security aspects, please see *GÉANT MD-VPN: Multi-Domain Virtual Private Network service - a seamless infrastructure for NRENs, GÉANT and NORDUnet* (pp. 12–14) [MDVPNWhitePaper].

MTU Consistency

When establishing a L2 circuit over the MDVPN infrastructure, the MTU that is supported needs to be defined. In LDP negotiation, an MTU value is negotiated between the PE routers but concerns only the control plane. It should be matched (or ignored by configuration). The data plane MTU value can differ from that of the control plane but it needs to be verified and communicated to the user institution. Each NREN participating in this service should notify the orchestrator of the supported MTU, as needed. MS Azure supports 1,500 bytes, so this should be consistent.

3.4.2.5 *MDVPN Proxy*

In the typical MDVPN architecture, GÉANT operates as a carrier of carriers, only transporting the overlay across its MPLS backbone, therefore it cannot natively terminate or partake in MDVPN circuits.

In the edges of the MDVPN fabric, a resource entity is needed to act as a bridging function between MDVPN and any other transport technology that carries Ethernet Frames. This entity is called MDVPN Proxy.

In the context of GÉANT, the MDVPN Proxy is deployed on a GÉANT PE router as a logical system and acts as an MDVPN PE router to terminate MDVPN circuits. It is then possible to stitch or "proxy" this termination point on the logical system to another L2VPN service on the same physical or logical PE router.

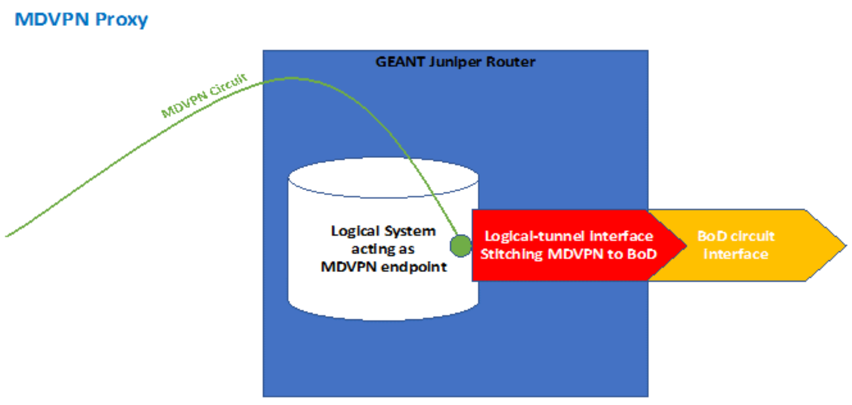
MDVPN Proxy
GEANT Juniper Router
MDVPN Circuit
Logical System acting as MDVPN endpoint
Logical-tunnel interface Stitching MDVPN to BoD
BoD circuit interface

Figure 3.6: MDVPN Proxy

This hybrid model reveals the complexity of orchestrated service delivery at the operational (OSS) level as it requires the orchestrator to undertake service activation of two different services (e.g. MDVPN and BoD) and to stitch them together.

In the context of the PoC, as mentioned above, MS Azure interconnects via the NetherLight exchange, which is BoD-enabled only. The orchestrator therefore needs to invoke the BoD service to provision the Exchange to anywhere in GÉANT but most likely to the closest MDVPN PE router (MDVPN Proxy) and then invoke an MDVPN circuit from that PE router to the NREN serving the institution that placed the order as well as stitching of the two services together on the GÉANT PE router.

4 Orchestration Framework Overview

Providing an orchestration solution for e-infrastructure / service provider integration requires a multi-dimensional approach. The providers involved need to adhere to a common basic set of information/entity-modelling principles, functional elements, process definitions and inter-provider APIs and at the same time share product specifications that map to their internal implementations of product-underpinning services. This section makes detailed reference to the proposed orchestration framework from all of the perspectives above, with tangible examples taken from the GÉANT Cloud Connectivity (GCC) proof of concept (PoC) in the particular case of MS Azure ExpressRoute connectivity services. For simplicity, in the remainder of the document the specific case of MS Azure ExpressRoute connectivity services in the context of the GCC offering is referred as “GCC Azure”.

4.1 Information Modelling

All participating service providers need to adhere to a basic set of modelled entities. This section outlines the entities adopted from TMF’s Information Framework (SID)[5] for the purposes of the GCC PoC, all of which (with appropriate extensions) are expected to apply to future, more advanced cases of orchestrated service delivery across providers.

- **Partners and Suppliers**. In the GCC case, all participating network service providers are considered partners, while cloud service providers are considered suppliers.
- **Agreements**. These are types of SID business interactions used to model (in the information systems of each participating organisation/partner) the arrangements for the delivery of GCC with other partners or suppliers. For example, the agreement between an NREN and an institution (of type UNI) models the arrangements for the NREN to serve the cloud traffic of that institution, while the agreement between GÉANT and an NREN (of type eNNI) models the arrangements between those two networks to serve cloud traffic in transit. Agreements can also be used to model variations of the GÉANT IaaS cloud offering, especially regarding the NREN’s involvement in service delivery (i.e. as reseller, referrer or underwriter) in different contexts. For the purposes of the PoC, it is assumed that the role for the institution-serving NREN is referrer without commission.
- **Request/Response Interactions**. Interaction objects of type request/response have been used to model communication between the individual systems in the orchestration framework architecture. Different types of request/response interaction objects include: Customer/Product Order, Service Order and Provisioning Request.

[5] For the detailed SID specifications, the reader is referred to *GB922 Information Framework (SID) – R16.5* [TMF-SID].

- **Product and Product Offerings**. For the purposes of delivering connectivity to cloud providers, the GCC product specification is defined. For simplification, it is assumed throughout this document that GCC refers solely to point-to-point Ethernet (L2) connectivity. The GCC product specification includes parameters such as the service VLAN (and its rewrite option), capacity and MTU. For the purposes of the orchestrator PoC, the GCC Azure product specification is defined, including the additional Azure ExpressRoute-specific parameter S-key. Based on the GCC Azure product specification, different product offerings can be made available to customers/institutions, upon which agreements can be established (e.g. differentiating in service delivery terms). For simplification, it is assumed for the purposes of the PoC that GCC Azure product offerings follow the GCC product specification, without any additional details.
- **Customer-Facing Services** (CFSs) (i.e. L2VPN in the case of GCC Azure) are used to model customer-facing, functional service parameters, while **Resource-Facing Services** (RFSs) (i.e. MDVPN, BoD or manual L2VPN delivery in the case of GCC Azure) are used to model infrastructure/operations-facing, technical service parameters of each corresponding product offering, such as the signalling protocol supported (BGP or LDP) in the case of the MDVPN RFS. For the purposes of the PoC, where the CFS is a single, non-composite service, it is omitted/merged into the RFSs used.
- **Resources** (e.g. edge devices, ports and device interfaces) are used to model the resources participating in each product offering/agreement via the corresponding CFS/RFSs.

An example of a subset of entities participating in product, CFS, RFS and resource specifications for GCC Azure is provided in Figure 4.1.

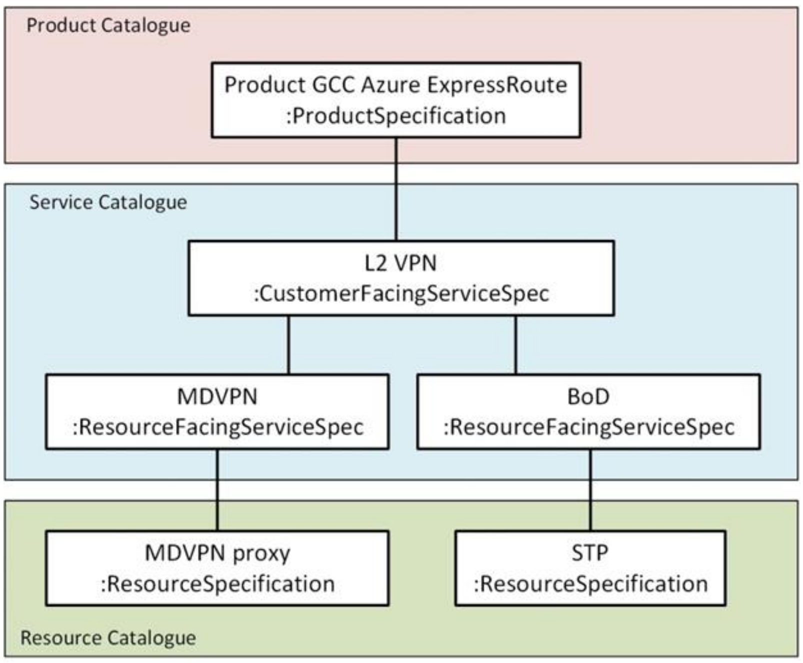
Product Catalogue
Product GCC Azure ExpressRoute :ProductSpecification
Service Catalogue
L2 VPN :CustomerFacingServiceSpec
MDVPN :ResourceFacingServiceSpec
BoD :ResourceFacingServiceSpec
MDVPN proxy :ResourceSpecification
STP :ResourceSpecification
Resource Catalogue

Figure 4.1: Example of a subset of entities in product, CFS, RFS and resource specifications for GCC Azure

The modelling (attributes, types and relationships) of entities is applied consistently throughout the orchestration framework, in all OSS and BSS systems participating on behalf of each domain, including systems that provide the user-order capturing functionality and end-to-end orchestration. They are also consistently used in all APIs across providers that are invoked by systems and orchestrators to achieve the streamlined end-to-end delivery of GCC.

It needs to be noted that simplifications have been applied to TMF SID modelling using object-oriented model implementation patterns. For example, instead of using the extensive class hierarchies of SID, a type attribute has been introduced, while significant attributes of the sub-classes have been rolled up to the super-class used.

Based on the MEF LSO documentation available and its compatibility with SID, the modelling adopted as presented in this document is compliant to a large extent with the MEF LSO management views and involved entities.

4.2 Functional Architecture

The proposed lightweight functional architecture (see Figure 4.2) follows TMF best-practices, while at the same time complying with the following fundamental assumptions:

- Orchestration is conducted centrally for the purposes of the PoC and minimal state details about the customer/product orders for GCC Azure, the corresponding products, the RFS service orders across the domains and RFS instances are also maintained centrally. The set of software components involved in end-to-end orchestration are logically grouped in Figure 4.2 under the name “GCC Orchestrator” and will be referred to as such from now on in the text.
- Central orchestration for PoC purposes does not limit future possibilities for distributed orchestration. The design of the GCC Orchestrator is such that any participating domain/partner in the GCC delivery chain can trigger an orchestration instance or undertake the orchestration role in future more advanced deployments of the solution.
- The individual GCC Orchestrator functional elements are what would normally be found within a service provider’s OSS/BSS architecture, thus the design by JRA1 T3 is entirely compatible with the GÉANT/NREN OSS/BSS architecture that JRA2 T2 is working upon [Filiposka].
- Each participating/partner domain (NREN, GÉANT, others) that is enrolled in the GCC Azure product offering should support a set of basic functionalities via stateless TMF-compliant APIs. Partner domains and the required functionality for each of them are depicted at the bottom of Figure 4.2 (see also Section 4.7.1).

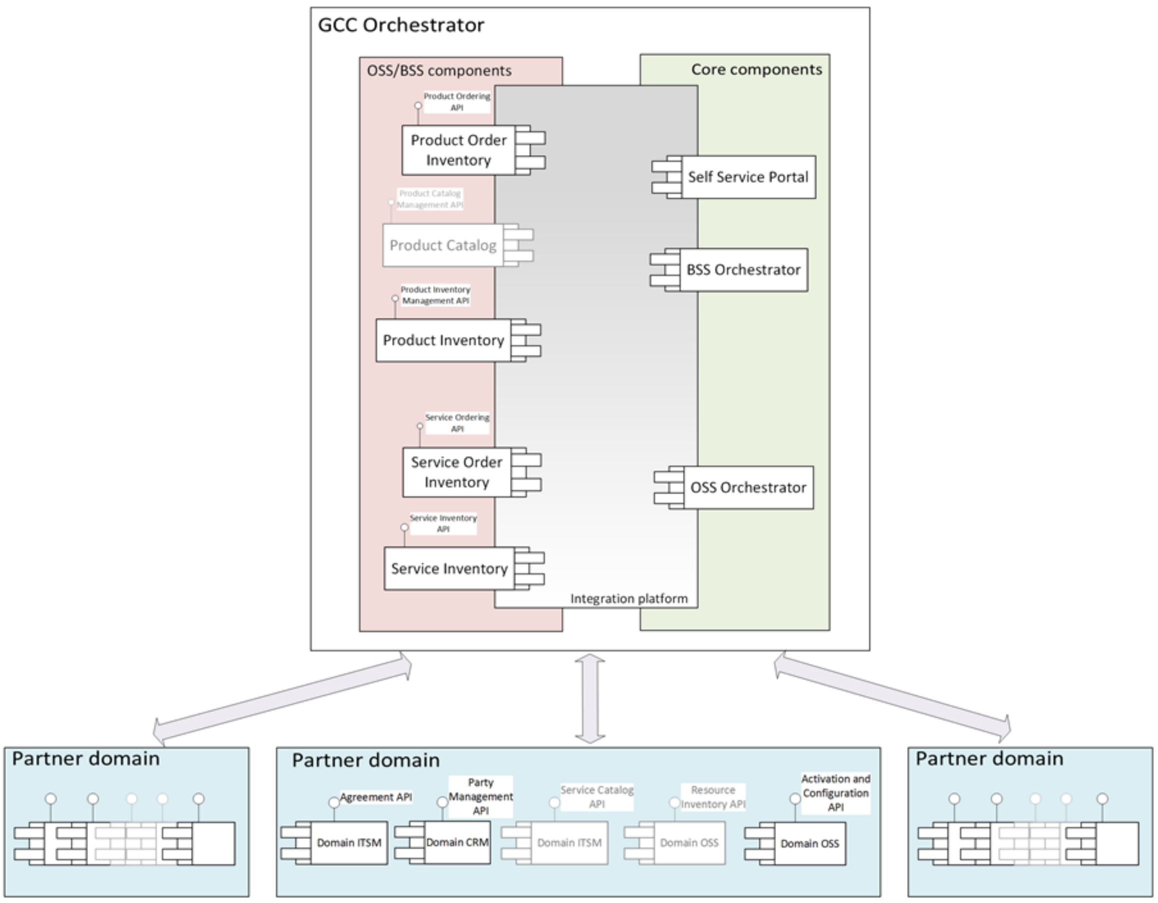

Figure 4.2: Functional architecture overview

4.2.1 GCC Orchestrator

More specifically, the components involved in the PoC software architecture, on behalf of the GCC Orchestrator, include:

- **Self-Service Portal (SSP)**. Consists of product/customer order capturing and management. The SSP generates Product/Customer Orders, including the involved parties and Product.
- **BSS and OSS Orchestrators**. Orchestrate business and operations interactions required for end-to-end service delivery both within the GCC Orchestrator and between the GCC Orchestrator and partner domains, by invoking internal GCC Orchestrator components or partner API end points correspondingly.

  The BSS Orchestrator receives Product/Customer Orders from the SSP and generates Service Orders comprising Resource-Facing Services (RFSs) which are then passed on to the OSS Orchestrator. The BSS Orchestrator is responsible for storing and managing the lifecycle of Product/Customer Orders.

  The OSS Orchestrator:

  ○ Undertakes the role of receiving Service Orders comprising RFSs from the BSS Orchestrator.

- ○ Applies logic related to the specifications or values of parameters of RFSs in partner domains (e.g. identifies the partner domains comprising an MDVPN fabric and the MDVPN fabric edge devices upon which configuration needs to be applied to implement the L2VPN).
- ○ Dispatches individual RFS configuration and/or activation requests to the participating partner domain systems, by leveraging the TMF Service Activation and Configuration Open API.

The OSS Orchestrator is responsible for storing and managing the lifecycle of Service Orders. The relationship between each Service Order and its generating Product/Customer Order should also be preserved. Finally, the OSS Orchestrator is responsible for associating and storing each Service Order item with the corresponding Service Activation and Configuration API call result.

Together, the two Orchestrator components deliver the core orchestration functionality within the GCC Orchestrator bundle.

- **Product Catalog**. The Product Catalog is the component managing and advertising – via the TMF Product Catalog Management Open API – the product offerings of the GÉANT/NREN partnership, as well as relevant product specifications and characteristics. For the purposes of the PoC, only the GCC Azure product offering is available through the catalogue. Therefore, the functionality of the Product Catalog is omitted from this document for simplicity.
- **Customer/Product Order Inventory**. This component manages the lifecycle of individual customer orders for products, as captured by the SSP, and exposes the Product Ordering Open API.
- **Product Inventory**. This component manages the lifecycle of individual products instantiated as a result of customer orders and exposes the Product Inventory Management Open API.
- **Service Order Inventory**. This component manages the lifecycle of RFS Service Orders, as they emerge from Customer/Product Orders, exposing the Service Ordering Open API.
- **Service Inventory**. This component manages the lifecycle of RFS instances that are involved in Service Orders, exposing the Service Inventory Open API.

4.2.2 Partner Domains

Partner domains are required to support a minimum level of functionality to participate in orchestrated service delivery. This includes exposing:

- Their served institutions' information (and contacts within institutions) via the Party Open API. This assumes some basic internal customer relationship management (CRM) functionality.
- Their agreements with other partner domains, as well as basic product catalogue information (product specifications and offerings). This assumes some basic contract/agreement and product catalogue management functionality (e.g. via an IT Service Management (ITSM) system). Product specifications should include service specifications and related resources; however, no Service Catalogue and Resource Inventory management API functionality is required as a minimum.

- API end points for the externally invoked service activation and configuration operations. This assumes the ability to consume RFS-specific activation and configuration requests and reply (synchronously or not) with success or failure. Such requests will often include configuration of local resources.

Beyond the PoC scope, in the longer-term scenario, the partner domains are expected to also expose the relevant Open APIs for:

- Service Catalogue management (Service Catalog Open API) to allow querying of (selective) service catalogue contents and service specifications by other partners.
- Resource Inventory management to allow (selective) querying of domain resource information relevant to the establishment of agreements or to the service activation and configuration across partner domains.

In cases where the required functionality exists, but it is not available via the specific Open APIs, the minimum API functionality requirements will be provided to interested partners who will have to create lightweight wrappers around their existing OSS/BSS systems to support them.

Although the detailed specifications of MEF LSO SONATA and INTERLUDE MIRPs are not available to Task 3 at the moment of writing, the functional architecture of the proposed orchestration framework follows the scope of SONATA and INTERLUDE, namely:

- The BSS Orchestrator's scope is equivalent to the LSO SONATA scope of non-control cross-domain interactions between the partnering Service Providers' business applications.
- The OSS Orchestrator's scope is equivalent to the LSO INTERLUDE scope of supervising a portion of services from the partnering Service Providers and control-related management interactions between them.

4.3 Process Logic and the Integration Platform

In designing the software architecture of the orchestration framework solution, the decision to decouple process logic from system integration logic was made, in accordance with relevant best practices.

For the purposes of the GCC Azure PoC, process logic (i.e. the steps to implement each of the processes that the OSS/BSS Orchestrators are executing) is undertaken by the Activiti BPMN solution, selected after thorough evaluation by the Task of open source process engines available. It is noted that Activiti was also selected at a later stage by JRA2 T2 as the process engine for the Service Provider Architecture (OSS/BSS) software suite. Activiti provides a flexible and extendable API to implement custom logic in BPMN processes. At the moment of writing, Activiti is being used as the process engine supporting the functionality of the BSS and OSS Orchestrator of the GCC Azure PoC.

In Activiti, developers can implement a Java service task or an event listener using plain Java to code the logic, or they can leverage the wide set of functionality offered by the Spring framework [Spring] and use expressions or delegate expressions in a service task. This allows the implementation of integration logic to invoke external services or applications from a BPMN process. However, the process engine is not the right place to implement this integration logic, especially if the external service or application uses a data model that is very different from the process data model. Then developers would need to implement transformation logic to be able to invoke the service or application interface.

Therefore, the preferred option has been to introduce a separate, integration platform element to the software architecture that will operate as a Service Bus, by utilising a Message Broker and Apache Camel to define the integration logic. The OSS/BSS software components of the orchestration architecture, as presented above, rely on the integration platform for reliable interoperation and messaging, model transformation and communication protocol abstraction.

Activiti provides out-of-the-box integration with Apache Camel [ApacheCamel], allowing this framework to handle the integration logic details. The addition of an integration platform to the architecture provides a clear separation between the process definition and instances on the one side and the logic to communicate with the solution components on the other. Adding an integration platform introduces an additional learning curve and maintenance requirements, but the Task has found it easy to leverage Apache Camel functionality without a lot of additional effort.

Figure 4.3 depicts how a simple Activiti process that requires integration with a CRM system and a provisioning system should communicate with an intermediate integration platform.

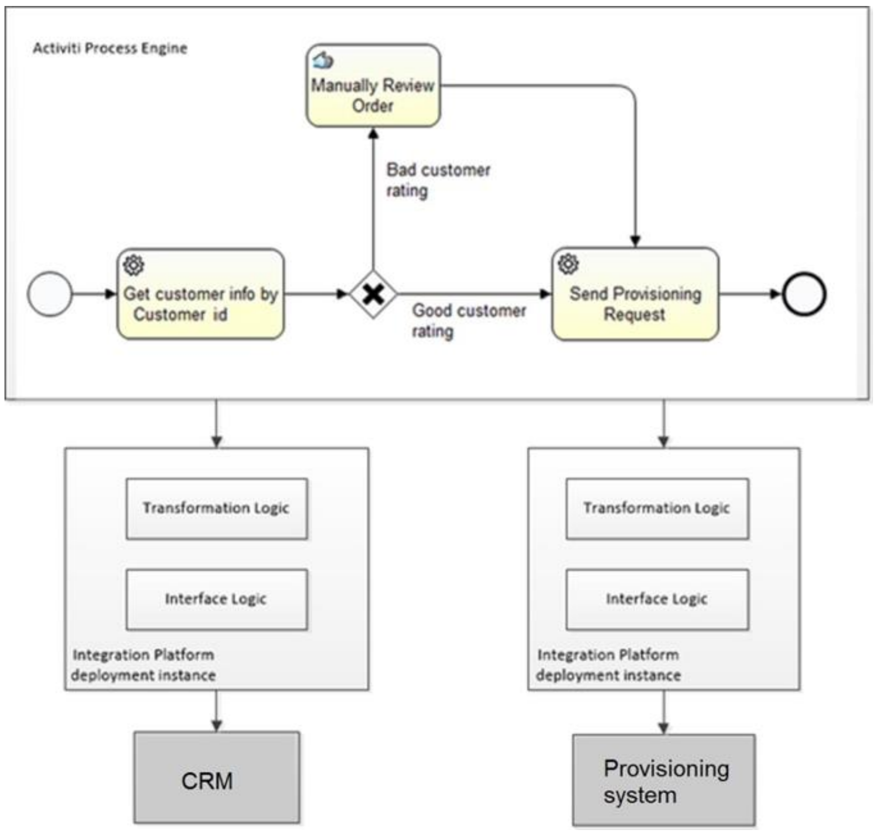

Figure 4.3: Communication between Activiti process, integration platform and systems to be integrated

The integration platform is split into the Message Bus component, which contains the routing and integration logic, and the service-specific components that will perform the translation and transportation logic for each service. It adopts a micro-services architecture, providing development and deployment flexibility as well as reliable integration by integrating the different components as containerised micro-services. Such micro-services play the role of an adapter for each of the OSS/BSS components and are responsible for translating and logging messages exchanged between the different components of the architecture (see Figure 4.4). At the moment of writing, the integration framework of the orchestration solution is under development and it is not expected to be included in early versions of the GCC PoC.

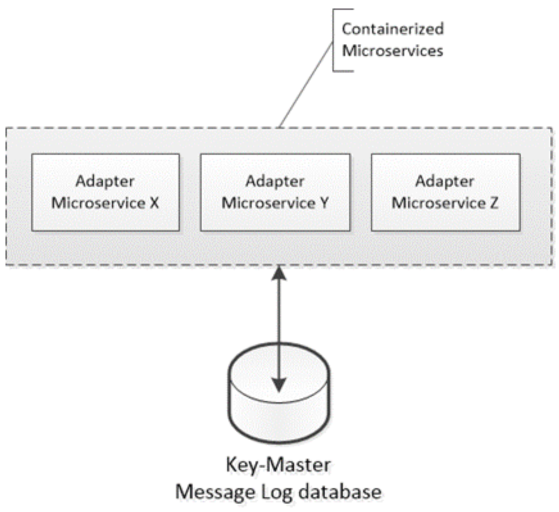

Figure 4.4: Integration micro-services

4.4 Processes

For the purposes of the GCC PoC, two high-level processes have been modelled and are supported, as follows:

- **Order Qualification process**, through which an institution/GCC customer can query the one-stop shop, via the SSP, about the availability of a specific product offering delivered to their premises. For example, in the case of the PoC, the institution can query whether the GÉANT Cloud Connectivity product offering to an MS Azure ExpressRoute site is available at its premises.

  In order for the product offering to be available, it is assumed that all prerequisite business arrangements have been addressed and are in place, namely, MS Azure ExpressRoute is one of the offerings in the GÉANT Cloud catalogue, the corresponding NREN has adopted the GÉANT IaaS framework and the contractual arrangements between the institution (or NREN, depending on the NREN role) and the cloud provider for the MS Azure ExpressRoute services are established, so that the customer institution has a valid Azure subscription.

Order qualification, as defined for the purposes of the PoC, is a business-level qualification of an order and does not address the technical feasibility of implementing a specific service instance across the involved domains.

- **Order Fulfilment process**, which includes the Order Management and Service Activation and Configuration sub-processes. This is the process that translates the Customer/Product Order to a Service Order and breaks it down to individual service items as they need to be addressed by the involved partners/domains. It handles the logic of reconciling cross-partner service parameters (e.g. S-VLAN rewrites in the event that the S-VLAN provided by MS Azure ExpressRoute cannot be supported by all involved partners) and invokes the participating partner domains to activate the individual service instances and configure the resources involved. The process undertakes the role of updating the involved systems (Service Order Inventory and Service Inventory) with the status of the individual Service Orders and service instances, so that the status of service order fulfilment regarding a specific product order can be made available at any point in time to all the involved parties (partners, customer, supplier(s)).

This basic set of processes comprises the minimum viable product for the proposed PoC and complies with the TMF eTOM specifications in a lightweight way. Furthermore, it complies with the Product Ordering and Service Activation Orchestration operational thread of MEF LSO, as outlined in Section 10.3 of *Service Operations Specification MEF 55, Lifecycle Service Orchestration (LSO): Reference Architecture and Framework* [MEF-55].

Further processes, such as partner onboarding, incident management, services decommissioning or maintenance management, as well as processes in the assurance area of eTOM (e.g. service quality management), can be added to the framework progressively, by extending:

- Required functionality from participating partner domains.
- Capabilities of the software systems involved to expose, process and manage additional entities.
- Required APIs to be supported between parties.

One of the assumptions made for the purposes of the GCC PoC is that the GCC Orchestrator does not confine itself to the role of issuing and managing Service Orders across partners, requiring the INTERLUDE MIRP of the MEF LSO. Instead, it also undertakes the role of preparing and issuing Service Configuration and Activation requests to the underlying partner domains, in an effort to offload RFS and resource configuration complexity from the underlying domains to itself and thus preserve the principle that underlying domains expose the minimum functionality and state. This needs to be reviewed in subsequent versions of the PoC, based also on the maturity of partnering domains to support INTERLUDE or more extensive service operations.

In the following figures, indicative sequence diagrams for these two generic processes are provided, as they will be implemented for the PoC.

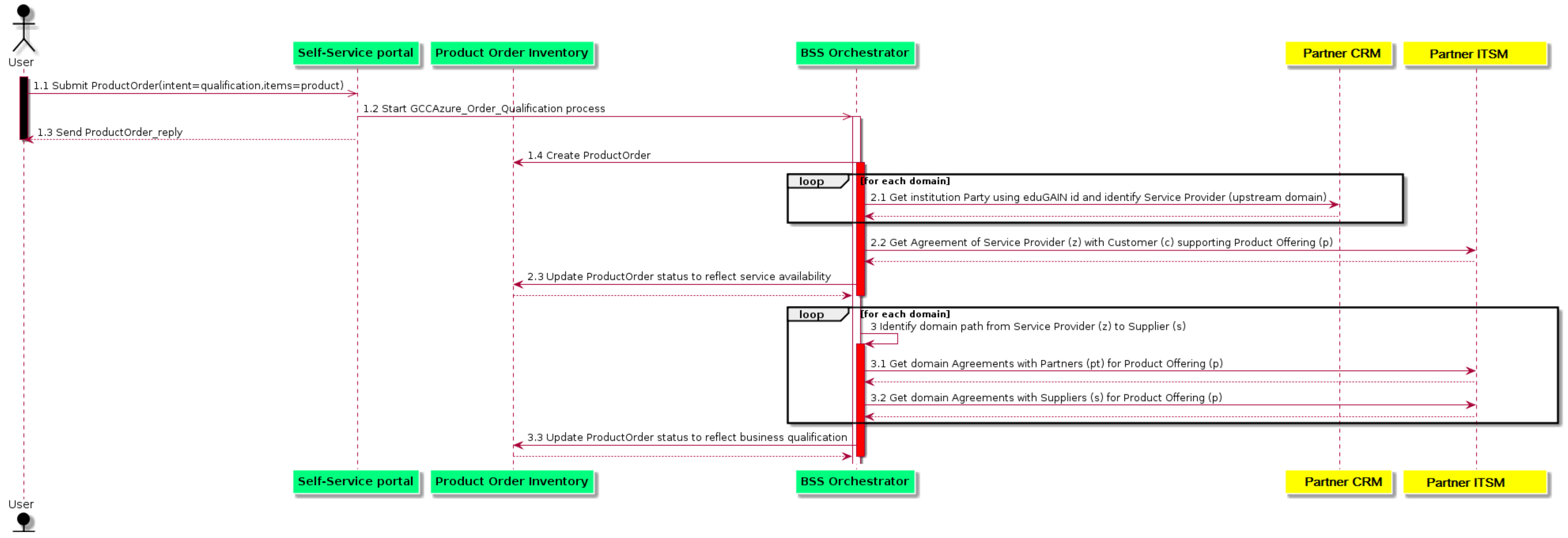
User
Self-Service portal
Product Order Inventory
BSS Orchestrator
Partner CRM
Partner ITSM
1.1 Submit ProductOrder(intent=qualification,items=product)
1.2 Start GCCAzure_Order_Qualification process
1.3 Send ProductOrder_reply
1.4 Create ProductOrder
loop
[for each domain]
2.1 Get institution Party using eduGAIN id and identify Service Provider (upstream domain)
2.2 Get Agreement of Service Provider (z) with Customer (c) supporting Product Offering (p)
2.3 Update ProductOrder status to reflect service availability
loop
[for each domain]
3 Identify domain path from Service Provider (z) to Supplier (s)
3.1 Get domain Agreements with Partners (pt) for Product Offering (p)
3.2 Get domain Agreements with Suppliers (s) for Product Offering (p)
3.3 Update ProductOrder status to reflect business qualification
Self-Service portal
Product Order Inventory
BSS Orchestrator
Partner CRM
Partner ITSM
User

Figure 4.5: Order qualification process sequence diagram

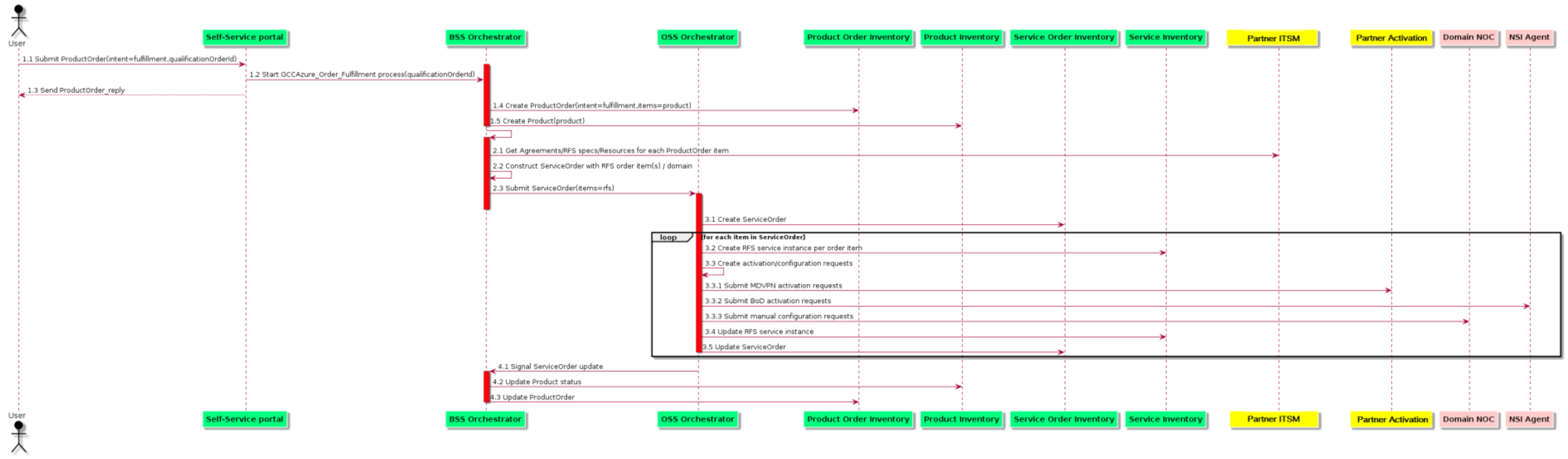
User
Self-Service portal
BSS Orchestrator
OSS Orchestrator
Product Order Inventory
Product Inventory
Service Order Inventory
Service Inventory
Partner ITSM
Partner Activation
Domain NOC
NSI Agent
1.1 Submit ProductOrder(intent=fulfillment,qualificationOrderId)
1.2 Start GCCAzure_Order_Fulfillment process(qualificationOrderId)
1.3 Send ProductOrder_reply
1.4 Create ProductOrder(intent=fulfillment,items=product)
1.5 Create Product(product)
2.1 Get Agreements/RFS specs/Resources for each ProductOrder item
2.2 Construct ServiceOrder with RFS order item(s) / domain
2.3 Submit ServiceOrder(items=rfs)
3.1 Create ServiceOrder
loop
[for each item in ServiceOrder]
3.2 Create RFS service instance per order item
3.3 Create activation/configuration requests
3.3.1 Submit MDVPN activation requests
3.3.2 Submit BoD activation requests
3.3.3 Submit manual configuration requests
3.4 Update RFS service instance
3.5 Update ServiceOrder
4.1 Signal ServiceOrder update
4.2 Update Product status
4.3 Update ProductOrder

Figure 4.6: Order fulfilment process sequence diagram

4.5 APIs

The orchestration framework requires adherence of all parties to selected API specifications from TMF's Open API suite. The reader is referred to the TMF Open API documentation [TMFOpenAPIdocs], where all the APIs (and their current development status) are presented (some access restrictions apply). From the set of TMF Open APIs, the proposed framework is adopting the following:

- **Product Ordering API**. Exposed by the Customer/Product Order Inventory and utilised by the following framework component:
    - BSS Orchestrator, in order to place a Product Order with all of the necessary order parameters and manage the Product Order.

  Each Product Order is created based on a product offering that is defined in the Product Catalog. The product offering identifies the product or set of products available to a customer, and includes characteristics such as pricing, product options and market.
- **Product Catalogue Management API**. Exposed by the Product Catalog and utilised by the following framework components:
    - SSP and BSS Orchestrators, in order to retrieve product offering specifications as/when needed (e.g. when assembling product orders).
- **Product Inventory Management API**. Exposed by the Product Inventory and utilised by the following framework component:
    - BSS Orchestrator, in order to create products (e.g. GCC Azure service instances) as a result of order fulfilment and manage products (e.g. update a product's termination date) as/when needed.
- **Service Ordering API**. Exposed by the Service Order Inventory and utilised by the following framework component:
    - OSS Orchestrator, to store Service Orders and order items, as well as update them based on the results of Service Activation and Configuration tasks from partner domains.
- **Service Inventory API**. Exposed by the Service Inventory and utilised by the following framework component:
    - OSS Orchestrator, to instantiate and manage service instances (e.g. a L2VPN within a partner domain).
- **Agreement API**. Exposed by all partner domains and invoked by the BSS Orchestrator in order to retrieve the per-domain business agreements for the delivery of products/services with other partners or suppliers.
- **Service Catalogue API**. Exposed by all partner domains to deliver information on the services offered by the domain to partners or the OSS Orchestrator.
- **Party Management API**. Exposed by all partner domains to deliver information on the parties (organisations and individuals) related to their business, as/when needed by external entities, e.g. the BSS Orchestrator.

- **Activation and Configuration API**. Exposed by all partner domains to receive service activation and service/resource configuration calls based on service specifications as made available through the partner's service catalogue.

4.6 Modelling for GCC Azure Use Case

For the specific use case of GCC MS Azure ExpressRoute connectivity services, the following product specification has been defined:

```
"productType" : "GCCAzure",
      "productSpecification" : {
        "@type" : "ProductSpecification",
        "specificationType" : "GCC",
        "characteristicSpecifications" : [ {
          "@type" : "CharacteristicSpec",
          "name" : "S_VLAN",
          "extensible" : false
        }, {
          "@type" : "CharacteristicSpec",
          "name" : "S_KEY",
          "extensible" : true
        }, {
          "@type" : "CharacteristicSpec",
          "name" : "VLAN_REWRITE",
          "extensible" : true
        }, {
          "@type" : "CharacteristicSpec",
          "name" : "BANDWIDTH",
          "extensible" : false
        }, {
          "@type" : "CharacteristicSpec",
          "name" : "MTU",
          "extensible" : false
        }, {
          "@type" : "CharacteristicSpec",
          "name" : "START_DATE",
          "extensible" : true
        }, {
          "@type" : "CharacteristicSpec",
          "name" : "END_DATE",
          "extensible" : true
        }, {
          "@type" : "CharacteristicSpec",
          "name" : "DESCRIPTION",
          "extensible" : true
        }]
           }
```

For the purposes of the PoC, it is assumed that the GCCAzure product characteristics are agreed across all partner domains, hence the need for a Product Catalog at the GCC Orchestrator is eliminated (as it would be used to manage and advertise a single product offering: GCC Azure). In a future, fully distributed scenario, characteristics of product offerings from partner domains would be defined separately and advertised via each domain's product catalogue. Compatibility or mismatches of product offerings would then be tackled at the BSS orchestration level. In the case of MEF LSO, common product specifications in accordance with the MEF CE 2.0 standard [MEF-CE2] (e.g. Access EPL product specification or UNI product specification) are defined by the standard itself, so all partners operate on the same product types with the same superset of characteristics.

Regarding RFSs, an indicative MDVPN service specification is provided below:

```
{
     "id" : "MDVPN",
     "identifiers" : [ {
       "namespace" : "domain:itsm",
       "value" : "4fa595cc-602c-498a-b153-767f5bf14b29"
     } ],
     "resources" : [ {
       "tag" : "GEANT-2",
       "resourceType" : "PE_ROUTER",
       "deviceInterfaces" : [ {
         "deviceInterfaceType" : "LOOPBACK_INTERFACE",
         "networkAddresses" : [ {
           "networkAddressType" : "DOMAIN_NAME",
           "networkAddressValue" : "eier.grnet.org"
         }, {
           "networkAddressType" : "IPv4",
           "networkAddressValue" : "62.217.102.13"
         }, {
           "networkAddressType" : "IPv6",
           "networkAddressValue" : "2001:648:2ff3::13"
         } ]
       } ],
       "ports" : [ {
         "resourceType" : "PORT",
         "characteristics" : [ {
           "name" : "MTU",
           "value" : "1500"
         }, {
           "name" : "VLAN_REWRITE",
           "value" : true
         }, {
           "name" : "FREE_SVLANS",
           "value" : [ 100, 101, 102 ]
         } ],
         "portName" : "<port-name>",
         "bandwidth" : {
           "measurementType" : "Gbps",
           "totalAmount" : 10
         }
       } ],
       "isProxy" : false
     }, {
       "resourceType" : "VLAN_RANGE",
       "fromValue" : 1,
       "toValue" : 100
     } ],
     "characteristicSpecifications" : [ {
       "name" : "S_VLAN_IN"
     }, {
       "name" : "S_VLAN_OUT"
     }, {
       "name" : "BANDWIDTH"
     }, {
       "name" : "MTU",
       "specificationCharacteristicValues" : [ {
         "value" : 1500
       } ]
     } ],
     "signalingProtocolSpecifications" : [ {
       "protocolType" : "BGP",
       "characteristicSpecifications" : [ {
         "name" : "VRF_TARGET"
       } ]
     } ]
   }
```

Still in the RFS domain, an indicative BoD service specification is provided below:

```
{
    "@type": "BODRfsSpecification",
    "id": "BOD",
    "identifiers": [
        {
            "namespace": "domain:itsm",
            "value": "08bf60c0-c65d-483d-866c-5ca1d130e28d"
        }
    ],
    "resources": [
        {
            "@type": "Source_STP",
            "name": "GEANT-Netherlight-AMS-MS-Express",
            "stp": "urn:ogf:network:geant.net:2013:topology:GEANT-
Netherlight-AMS-MS-Express"
        },
        {
            "@type": "Dest_STP",
            "name": "MICROSOF-AMS",
            "stp":
"urn:ogf:network:netherlight.net:2013:production7:microsoft-ams"
        },
        {
            "@type": "ERO_STP",
            "name": "GEANT-AMS",
            "stp":
"urn:ogf:network:netherlight.net:2013:production7:geant-ams"
        }
    ],
    "characteristicSpecifications": [
        {
            "@type": "CharacteristicSpec",
            "name": "MTU",
            "specificationCharacteristicValues": [
                {
                    "defaultValue": true,
                    "value": 1500
                }
            ]
        },
        {
            "@type": "CharacteristicSpec",
            "name": "S_VLAN_IN"
        },
        {
            "@type": "CharacteristicSpec",
            "name": "S_VLAN_OUT"
        },
        {
            "@type": "CharacteristicSpec",
            "name": "DESCRIPTION"
        },
        {
            "@type": "CharacteristicSpec",
            "name": "BANDWIDTH"
        },
        {
            "@type": "CharacteristicSpec",
            "name": "BOD_STATUS"
        },
        {
            "@type": "CharacteristicSpec",
            "name": "SERVICE_TYPE"
        },
        {
            "@type": "CharacteristicSpec",
            "name": "UNPROTECTED"
        },
        {
            "@type": "CharacteristicSpec",
            "name": "PATH_CALCULATION"
        },
        {
            "@type": "CharacteristicSpec",
            "name": "START_DATE"
        },
        {
            "@type": "CharacteristicSpec",
            "name": "END_DATE"
        }
    ]
}
```

For the purposes of NSI domain integration to the framework, the Service Activation and Configuration calls of the OSS Orchestrator need to be mapped to NSI calls via software adapters. The Task has implemented the adapters to:

- Get BoD domains, get ports, list of VLANs/port [RESTW-nsi_ports].
- Signal NSI connections [RESTW-nsi_conns].

At this point, it needs to be emphasised that, in TMF SID, the distinction between product and service is very important, although it adds complexity to the systems and their interactions. This distinction allows the decoupling of what is offered to the user (product) and its supporting technologies on the side of the providers (services, resources). Modelling the internal OSS/BSS systems, and also the inter-provider orchestration functions and APIs, following the product/service distinction is of paramount importance in maintaining a functional inter-service-provider fabric. Partners can change or replace their service implementations (e.g. the technology to offer L2VPNs) without affecting their partnership agreements to offer end-to-end products such as GCC.

As an additional note, TMF in collaboration with MEF LSO have negotiated the necessary TMF SID extensions to support LSO requirements. One of the extensions is a strongly typed extension pattern (using @type and @schemaLocation), which can be used as a common basis of product/service/resource types and schema locations for each type across the partners. This can be applied as is in the case of GÉANT/NREN partners or any other partnership with other e-infrastructures, for example, for:

- A common type/schema for cloud connectivity products across a partnership.

- Making available each partner's schema for a L2VPN service specification, where differences between schemas might occur.

Based on the type/schemaLocation pattern, inter-provider orchestration functions can implement logic to address different but compatible service specifications, service configurations that span more than one partner, etc.

4.7 PoC-Related External Requirements

4.7.1 Partner (Network) Domains

In the early stages of the PoC, very minimal requirements are expected from partner (e.g. NREN) domains:

- Domains are not required to preserve service state in their own OSS/BSS systems.
- Domains are only required to expose the Agreement, Party Management and Activation and Configuration TMF Open APIs.
- Domains that cannot expose the required APIs should be provided with an "out of band" means to insert the required data into the GCC Orchestrator.

However, in the long term, partner domains are expected to function as service providers with a wider range of capabilities (e.g. service catalogue management). Providing GÉANT/NRENs with a reference OSS/BSS stack internally that will support full-scale inter-domain orchestration scenarios, such as those required by the JRA1 T3-defined orchestration framework, is a responsibility of GN4-2 JRA2 Task 2 and the relevant work therein on the Software Provider Architecture [Filiposka]. The reader can verify full compatibility between the JRA1 T3 work on inter-service-provider orchestration as presented in this document and the JRA2 T2 SPA. For example, the SPA components for order management, service inventory, product/service catalogue, resource inventory, also exposing the TMF Open APIs, can fulfil the requirements of the partner domain functionality (see Figure 4.2) for the orchestration framework. Thus the JRA2 T2 work on intra-service-provider OSS/BSS architecture supports the JRA1 T3 framework with regard to the common goal of evolving GÉANT/NRENs into advanced service providers, capable of supporting inter-service-provider orchestration as denoted by standards and common practices.

4.7.2 MS Azure ExpressRoute Supplier

For the purposes of the PoC, the GCC Orchestrator and/or the SSP will query the Microsoft Azure ExpressRoute Connectivity Provider-Facing API (using the GÉANT account) via the `/services/networking/crossconnections` call, using the corresponding S-key to retrieve the status and parameters of an ExpressRoute connectivity instance on the Microsoft side up to the eNNI between the Microsoft Azure cloud and the upstream GÉANT partner (SURFnet/NetherLight at the moment of writing). The status is provided by the `ProvisioningState` and `Status` parameters, as shown in the sample output below.

```
<CrossConnection>
 <Bandwidth>500</Bandwidth>
 <PrimaryAzurePort>GEANT-AMB-06GMR-CIS-3-PRI-A</PrimaryAzurePort>
 <ProvisioningState>Provisioned</ProvisioningState>
 <STag>3</STag>
 <SecondaryAzurePort>GEANT-AMB-06GMR-CIS-4-SEC-A</SecondaryAzurePort>
 <ServiceKey>3a358b6a-f400-44db-ac6e-131a0250d34d</ServiceKey>
 <Status>Enabled</Status>
 </CrossConnection>
 <CrossConnection>
 <Bandwidth>50</Bandwidth>
 <PrimaryAzurePort>GEANT-AMB-06GMR-CIS-3-PRI-A</PrimaryAzurePort>
 <ProvisioningState>Provisioned</ProvisioningState>
 <STag>6</STag>
 <SecondaryAzurePort>GEANT-AMB-06GMR-CIS-4-SEC-A</SecondaryAzurePort>
 <ServiceKey>b8cc9008-7f48-4c94-82ab-5daef1eef079</ServiceKey>
 <Status>Enabled</Status>
 </CrossConnection>
```

4.7.3 Institution/Campus

For the purposes of the PoC, the following assumptions are made:

- The institution NOC/Operations team submits the GCC order on behalf of the end user.
- Campus is QinQ capable.
- There is only one peering with the upstream NREN (no multi-homing).
- The interactions with the institution NOC/Operations team are orchestrated using emails that include hyperlinks that can be directly used to complete manual functions required by the GCC Orchestrator.

4.8 The Framework in Action

In the following sections, some more insight is provided on how the Orchestrator can handle different partners employing different services to deliver GCC Azure. Several combinations of services and service availability across the end-to-end path can be accommodated by the Orchestrator, with manual tasks being spawned towards the relevant entities when automation is not an option.

4.8.1 BoD Orchestrated Workflow

Assuming a scenario (shown in Figure 4.7) where GÉANT peers directly with two different cloud service providers (CSP 1 and 2) to which an NREN provides GCC for two institutions (Institution 1 and 2), the availability of BoD service determines the orchestrator logic.

In cases where BoD STPs are advertised as BoD resources in NREN to Institution UNI agreements (e.g. STP B for Institution 2) and/or GÉANT to CSP NNI (of type supplier) agreements (e.g. STP W for CSP 2), the GCC Orchestrator can invoke the NSI interface of the Aggregator NSA (NSI-based provisioning) to take care of end-to-end service activation.

In cases where STPs are not colocated with the customer (Institution 1, STP A) or supplier (CSP 1, STP Z), the Orchestrator undertakes the responsibility to spawn the manual tasks towards the corresponding NREN NOC (1.1, 1.3) and/or the GÉANT NOC (3.1, 3.3) to request manual implementation (VLAN stitching) of the last-mile path towards the Institution and the CSP.

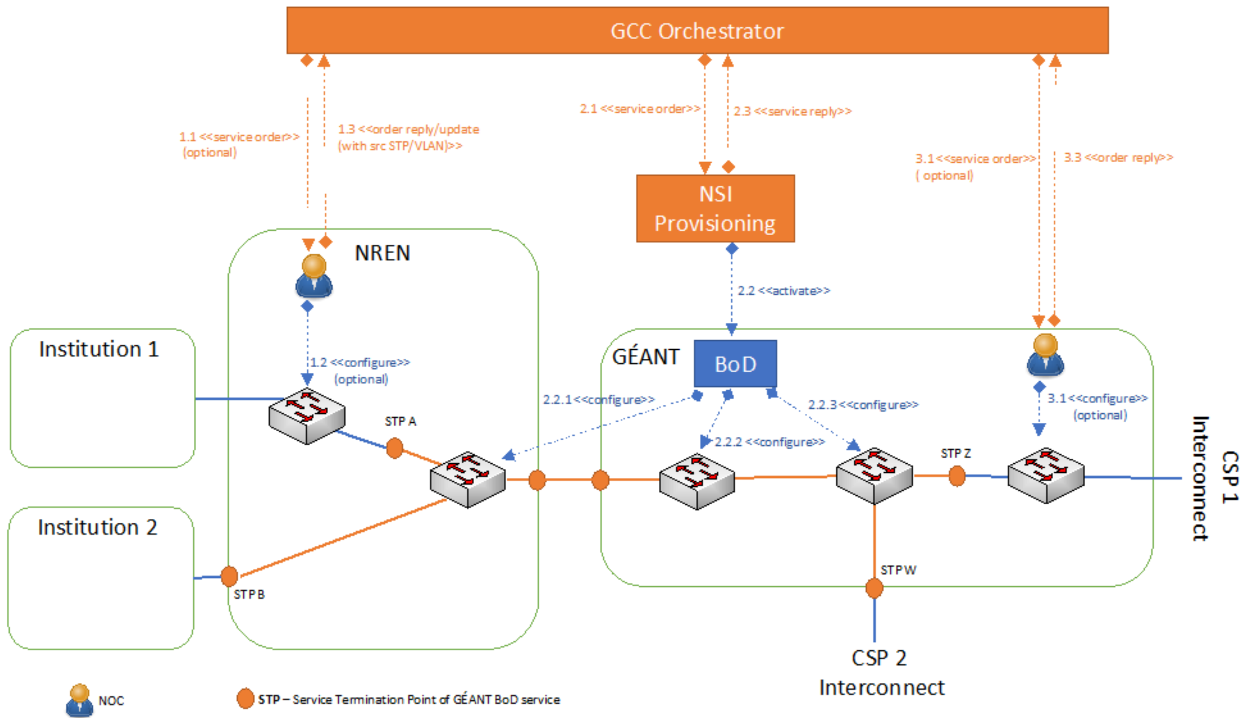

Figure 4.7: BoD orchestrated workflow

4.8.2 MDVPN Orchestrated Workflow

Figure 4.8 below (repeated from page 18 for ease of reference) represents the orchestrated workflow where GÉANT and the institution-facing NREN are using MDVPN as a transport platform. The GÉANT backbone is considered a preconfigured underlying fabric that does not actively participate in the workflow.

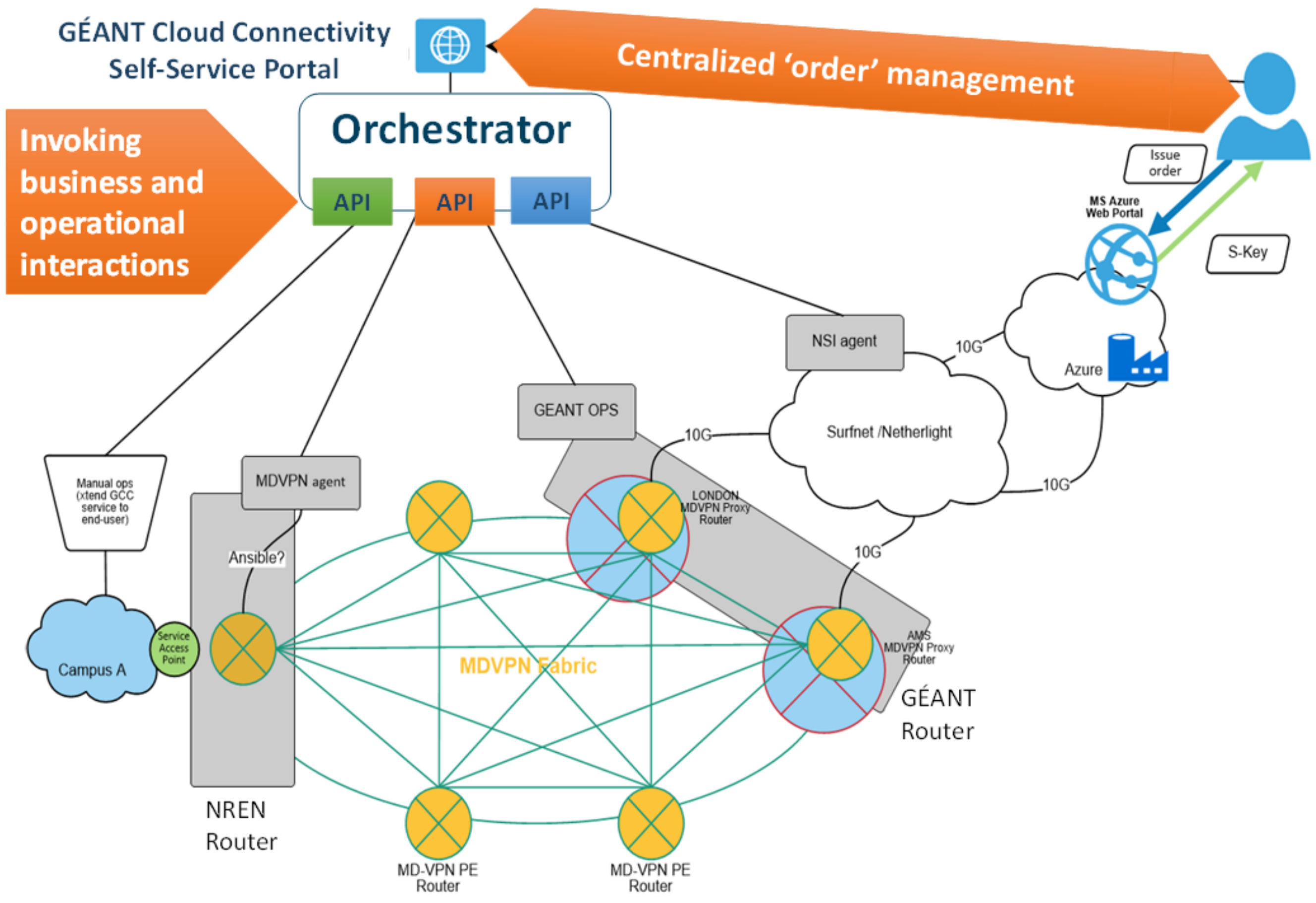

Figure 4.8: Overview of service delivery orchestration

The yellow-marked routers that reside inside the blue-marked routers represent the logical systems inside GÉANT Amsterdam and London PE routers that implement the MDVPN Proxies. The ports that terminate the Microsoft Azure connections from NetherLight should belong to the logical-systems that realise the MDVPN Proxies.

MDVPN Proxies, per definition and configuration, are identical to any other PE router participating in the MDVPN fabric and are operated by GÉANT Operations.

Assuming the user has placed an ExpressRoute order through the Microsoft Azure portal and the S-key details are provided; the GCC Orchestrator then uses NSI to signal towards NetherLight to provision the part of the service between the MS Azure peering points and GÉANT MDVPN Proxies.

Focusing on MDVPN service configuration and activation, the Orchestrator then:

- Notifies GÉANT Operations to configure the MDVPN Proxy end point interconnecting to NetherLight (stitching point between MDVPN and BoD), providing all necessary configuration information required.
- Signals each involved NREN's MDVPN service delivery agent (which can be Operations engineers, provisioning APIs or other) to configure the local MDVPN end point with all necessary configuration information required.

Orchestrator calls via the Service Activation and Configuration API can be mapped internally by each underlying domain to technology-specific configurations to fulfil the request.

There are two options for signalling the service: LDP and BGP. Each of them requires a different kind of configuration in order to be realised.

1. **LDP**. If LDP is chosen, then as a prerequisite it must be ensured that the MDVPN Proxies have their filters configured in a way that permits targeted LDP (tLDP) sessions from any other PE router. Under this option the following configuration snippet should be applied to the selected MDVPN Proxy.

   Data provided by the Orchestrator via the Service Activation and Configuration API calls:

   a. Selected MDVPN Proxy (`proxy_pe-router-ip`).
   b. S-VLAN, as provided by MS (`s-vid`).
   c. Target PE router (`t_pe-router-ip`).
   d. Virtual Circuit ID (this value should be unique per pair of PE routers).
   e. Maximum transmission unit (MTU) (control plane). It is recommended that this value be equal to the actual MTU on the data plane.
   f. Control word or not (this could be a global feature in accordance with GÉANT's policy).
   g. Description – a piece of text suitable for identifying each service instance.

   Using the above parameters and the following configuration template, the final configuration could be produced.

   Interface configuration

```
set logical-systems mdvpn interfaces <ifce-to-netherlight> unit <s-vid>
description "<description>"
set logical-systems mdvpn interfaces <ifce-to-netherlight> unit <s-vid>
encapsulation vlan-ccc
set logical-systems mdvpn interfaces <ifce-to-netherlight> unit <s-vid>
vlan-id <s-vid>
```

   L2 circuit configuration

```
set logical-systems mdvpn-proxy protocols l2circuit neighbor <t_pe-
router-ip> interface  <ifce-to-netherlight>.<s-vid> virtual-circuit-id
<vc-id>
set logical-systems mdvpn-proxy protocols l2circuit neighbor <t_pe-
router-ip> <ifce-to-netherlight>.<s-vid> description "Testing L2 over
MDVPN"
set logical-systems mdvpn-proxy protocols l2circuit neighbor <t_pe-
router-ip> interface <ifce-to-netherlight>.<s-vid> control-word
set logical-systems mdvpn-proxy protocols l2circuit neighbor <t_pe-
router-ip> interface <ifce-to-netherlight>.<s-vid> mtu 9022
```

2. **BGP**. If BGP is chosen, the same infrastructure used by L3 VPN services, namely the VPN route reflectors, can be used. Since the `l2-vpn bgp` family is already enabled between the sessions of the PE routers and the VRRs, this is the only value needed.

   Data provided by the Orchestrator via the Service Activation and Configuration API calls:

   a. Selected MDVPN Proxy (`proxy_pe-router-ip`).
   b. S-VLAN, as provided by MS (`s-vid`).

c. Target PE router (`t_pe-router-ip`).
d. Virtual routing and forwarding (VRF) target (`target:<vrf_target_value>`) (this value should be unique in all MDVPN infrastructure).
e. Maximum transmission unit (MTU) (control plane). It is recommended that this value be equal to the actual MTU on the data plane.
f. Description – a piece of text suitable for identifying each service instance.

In the same way, the following template can be used in order to create a BGP-signalled service instance.

Interface configuration

```
set logical-systems mdvpn interfaces <ifce-to-netherlight> unit <s-vid>
description "<description>"
set logical-systems mdvpn interfaces <ifce-to-netherlight> unit <s-vid>
encapsulation vlan-ccc
set logical-systems mdvpn interfaces <ifce-to-netherlight> unit <s-vid>
vlan-id <s-vid>
```

Routing-instance configuration

```
set logical-systems mdvpn routing-instances service-1 instance-type l2vpn
set logical-systems mdvpn routing-instances customer-1 interface <ifce-
to-netherlight>.<s-vid>
set logical-systems mdvpn routing-instances customer-1 route-
distinguisher <proxy_pe-router-ip>:1
set logical-systems mdvpn routing-instances customer-1 vrf-target
<target:<vrf_target_value>
set logical-systems mdvpn routing-instances customer-1 protocols l2vpn
encapsulation-type ethernet-vlan
set logical-systems mdvpnrouting-instances customer-1 protocols l2vpn
interface <ifce-to-netherlight>.<s-vid>
set logical-systems mdvpn routing-instances customer-1 protocols l2vpn
site PE-A site-identifier 1
set logical-systems mdvpn routing-instances customer-1 protocols l2vpn
site PE-A interface <ifce-to-netherlight>.<s-vid>
```

5 Orchestration with Non-GÉANT/NREN Service Providers

As the model described in this document is fully TMF compliant, it also expects actors participating in the orchestrated workflows to adhere to and understand these specifications and data structures. While there will undoubtedly be successes with GÉANT/NRENs, and possibly even institutions, following the framework, realistically it will be very difficult to persuade larger commercial organisations to adhere to it precisely. Addressing other e-infrastructures with the framework is expected to be equally challenging. Therefore, a compromise will be required whereby the framework can continue to adhere fully to the TMF specifications while being flexible enough to allow communications with other parties that do not.

For interoperation with other service providers, outside of the GÉANT-NREN partnership, it is important to define a minimum set of RESTful stateless APIs and interaction entities that need to be supported. In case such providers do not offer TMF-compliant end points for orchestrated service delivery, different levels of compliance/interoperation capabilities are expected to be made available on a case-by-case basis, making integration with a fully orchestrated workflow more or less possible.

For example, minimum compliance could include the ability for the (centralised or not) orchestrator to query and update the order status with a service provider. This would not include any interaction to actually provision the service, thus there would be no fulfilment capabilities as part of orchestration, assuming fulfilment/service instantiation is out of the control of the orchestrator, as is the case with the MS Azure ExpressRoute provisioning of resources within the Microsoft infrastructure in the GCC Azure PoC.

Another solution is to write specific wrappers or translators for individual service providers as simplified functions sitting between any NREN/GÉANT orchestrator, or the GCC Orchestrator as presented in this document, and the non-compliant service provider. Wrappers would accept TMF standard calls and data payloads from the orchestrator and reprocess (translate) these into provider-specific API calls and payloads, and vice versa for response or incoming API actions and data.

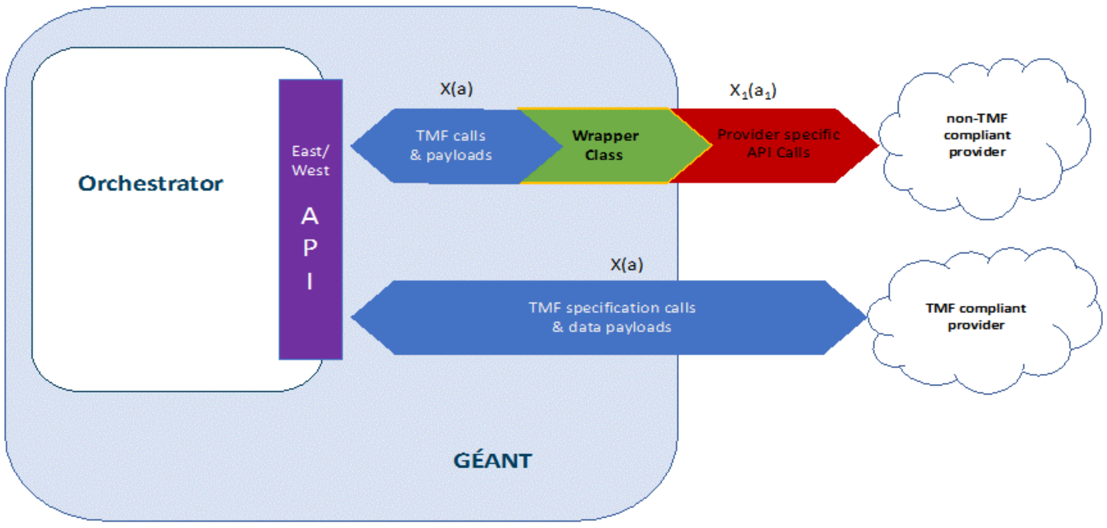

Figure 5.1: Interaction between GCC Orchestrator and non-TMF-compliant SP via a wrapper function

Figure 5.1 illustrates this interaction between the proposed GCC Orchestrator and a non-TMF-compliant service provider via a wrapper function. This case is demonstrated in Table 5.1 using the example of the GCC Azure PoC when making API calls to both the GCC Orchestrator and Microsoft Azure to retrieve a list of existing orders.

As can be seen, the wrapper class will need to contain definition mappings (depicted by matching colours in Table 5.1) between API calls and parameters, and will also need to carry out some data-processing actions on the payloads exchanged to ensure they are presented in a compatible format to the other side.

TMF API Call:	Microsoft API Call:
/product-order-inventory/api/product-order	https://management.core.windows.net/<subscription-id>/services/networking/crossconnections?api-version=1.0

TMF Payload data	Microsoft Azure Payload data

TMF Payload data

```
  <snip>
 "product" : {
      "type" : "GCCAzure",
      "status" : {
        "status" : "NEW",
        "statusDate" : 1501753974163
      },
      "productCharacteristics" : [ {
        "name" : "S_KEY",
        "value" : "xxxx-xxxx-xxxx-xxxx-xxxx"
      }, {
        "name" : "S_VLAN"
        "value" : "3"
      }, {
        "name" : "VLAN_REWRITE"
      }, {
        "name" : "BANDWIDTH"
        "value" : "500"
      }, {
        "name" : "MTU
        "value" : "1500"
      }, {

<snip>
```

Microsoft Azure Payload data

```
<CrossConnections
xmlns="http://schemas.microsoft.com/windowsazure"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
<CrossConnection>
    <Bandwidth>500</Bandwidth>
    <PrimaryAzurePort>GEANT-AMB-06GMR-CIS-3-PRI-A
    </PrimaryAzurePort>

<ProvisioningState>Provisioned</ProvisioningState>
    <STag>3</STag>
    <SecondaryAzurePort>GEANT-AMB-06GMR-CIS-4-SEC-A
     </SecondaryAzurePort>
    <ServiceKey>xxxx-xxxx-xxxx-xxxx</ServiceKey>
    <Status>Enabled</Status>
</CrossConnection>
</CrossConnections>
```

Table 5.1: Interaction between GCC Orchestrator and non-TMF-compliant SP via a wrapper function: GCC Azure

This obviously creates an overhead when onboarding a new service provider that is not TMF compliant, as API documents and data structures need to be studied and understood. However, it should be possible to define a minimum viable wrapper template that can be used as a basis, to speed up this process. In time, given the ongoing work within the MEF and TMF by large service providers, it is hoped that more and more providers will be compliant with these specifications and therefore ease the technical burden of the onboarding process.

As part of the work undertaken by the Task in determining the interoperable components to focus on, a number of other service providers and e-infrastructure platforms were investigated in greater detail, namely INDIGO DataCloud [INDIGODataCloud], EGI FedCloud [EGIFedCloud] and CloudSigma [CloudSigma]. In the next section, more insight into the findings regarding interoperation with INDIGO DataCloud is provided, as the most mature interoperation case in the Task's investigation.

5.1.1 The Case of INDIGO DataCloud

INDIGO DataCloud (DC) provides open source software cloud components specifically aimed at the scientific community. It has a well-established community and publishes a large suite of APIs to facilitate the management of most aspects of the VM resource lifecycle. The network connectivity elements, however, are more focused on internal virtual network manipulation using Open Cloud Computing Interface (OCCI) for OpenStack and OpenNebula. There are more recent attempts at addressing the need for external or WAN connectivity to interconnect disparate sites through use of virtual routes and overlay VPN networks [INDIGO-VR].

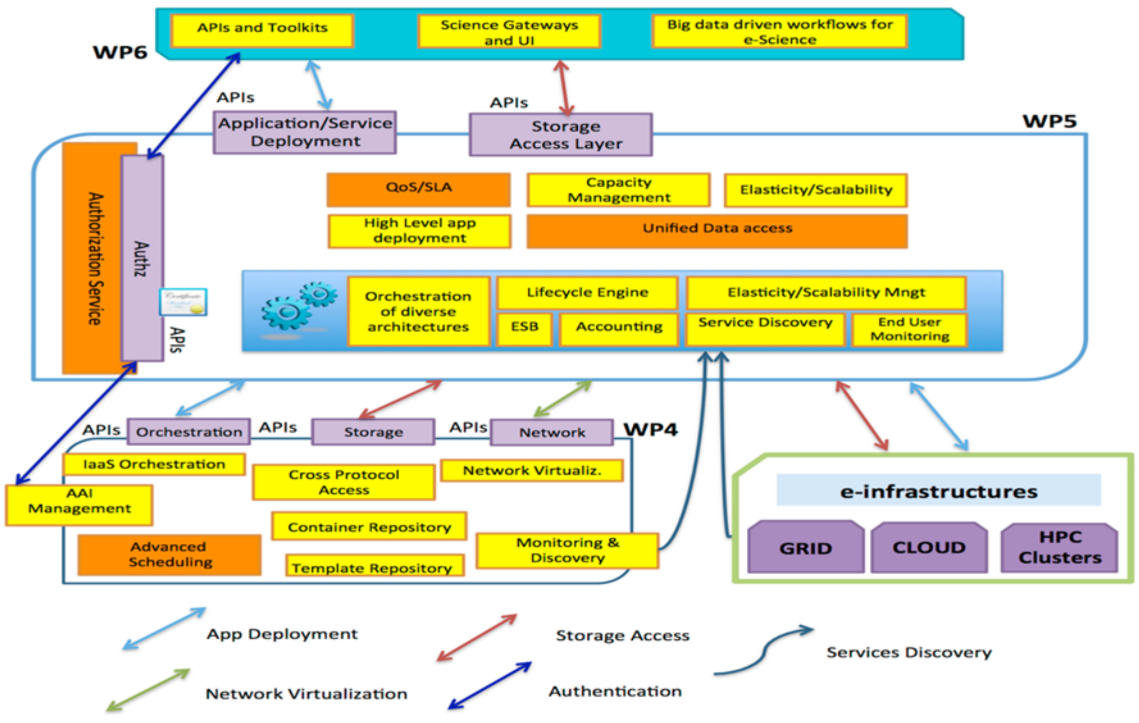

Figure 5.2: Overview of INDIGO-DC components

Figure 5.2 above provides an overview of the components in the INDIGO-DC software stack. Following a review of operating documentation and determining the best possible way to integrate the order qualification and fulfilment workflow, the following elements of the software stack emerged as more relevant:

- Science Gateways.
- Network Orchestration Wrapper (NOW).
- Infrastructure Manager (IM).

These elements currently provide interfaces to internal virtual networking components and it must be determined whether these can be expanded to include WAN configuration abilities.

Support for TOSCA templates (as currently available through the NOW and IM components) by INDIGO-DC is a potential interoperation opportunity, as such templates could be utilised to model network connectivity services.

The current release of the INDIGO-DC PaaS layer includes support of TOSCA-based services description by the INDIGO-DC Platform as a Service (PaaS) orchestrator and IM, which is based on the OpenStack Heat Translator. The released PaaS layer provides automatic distribution of applications and/or services over a hybrid and heterogeneous set of IaaS infrastructures. It can be used with both private and public clouds. The PaaS layer is able to accept a description of a complex set, or cluster, of services/applications by mean of TOSCA templates, and is able to provide the needed brokering features in order to find the best-fitting resources.

The orchestrator delegates the deployment to components such as the IM, while the INDIGO Configuration Management Database (CMDB) Service supports a set of configuration elements that are vital for INDIGO-DataCloud operations:

- Provider: organisational entity that owns or operates the services.
- Service (both computing and storage): main technical component description defining type and location of technical end points.
- Images: local service metadata about mapping of INDIGO-wide names of images, which are necessary to translate a TOSCA description into a service-specific request.

The IM supports the TOSCA Simple Profile in YAML Version 1.0. It provides TOSCA-compliant local site orchestration. It is also used to deploy TOSCA-compliant definitions of application architectures, i.e. TOSCA templates, on cloud sites external to INDIGO-DC, such as public clouds, or even on-premises clouds that are not integrated with the IM service. The latest INDIGO release supports integration with commercial clouds (e.g. the Open Telekom Cloud (OTC) by TSystems and Virtual Private Cloud (VPC) capabilities for integration with Amazon Web Services (AWS)), as well as automatic deployment through Ansible recipes embedded in TOSCA and Heat Orchestration Template (HOT) templates.

It is envisaged that specialised WAN services (e.g. L2VPNs) provided by GÉANT and NRENs can be leveraged by facilities operating the INDIGO-DC stack, by developing solutions in which TOSCA templates from INDIGO-DC lead to network product offering orders in the proposed orchestration framework (and vice versa). Furthermore, internal orchestration within INDIGO-DC deployments will need to be coordinated with inter-provider operations in the same way that the GCC Orchestrator can invoke the MS Azure API in the PoC presented in this document. Further work is needed in this direction, and the Task has established a liaison with the INDIGO-DC representatives accordingly.

6 Conclusions

During GN4-2 Period 1, Joint Research Activity 1 Network Infrastructure Evolution, Task 3 Integration with Other e-Infrastructures (JRA1 T3) has made significant progress towards achieving its primary goal of delivering a reference systems architecture and framework for operational integration and service delivery orchestration across e-infrastructures and commercial service providers.

In selecting order management and service fulfilment as the area of focus from the wide variety of processes and interactions spanned by inter-provider integration, the proof of concept (PoC) addresses a real requirement: it is anticipated that GÉANT's Infrastructure as a Service (IaaS) framework is likely to create a volume of service requests across the GÉANT/NREN footprint that it will be impractical to handle with the (manual) processes currently in place.

Furthermore, in its selection of the type of service to consider – layer 2 (L2) network connectivity between an NREN-connected institution and a commercial cloud service provider (CSP) via GÉANT and other intermediate network providers – and of the specific service to analyse – Microsoft (MS) Azure ExpressRoute connectivity service (underpinned by Bandwidth on Demand (BoD) and MDVPN) – the PoC addresses representative actual challenges (business, operational, technical) and demonstrates the potential to deliver very real benefits. These include minimising lead times as much as possible, and providing the user with both a one-stop-shop experience for end-to-end cloud connectivity and visibility and transparency regarding the state of his service orders as they are handled by all the providers involved.

The framework is both flexible and extensible. It is designed so that it can be applied "as is" to any other case of orchestrated delivery of connectivity services between an R&E institution and a cloud provider participating in the GÉANT IaaS framework, utilising the intermediate network domains. It can also be applied to the delivery of connectivity services between R&E institutions and other (not necessarily cloud) e-infrastructure providers (e.g. storage/repositories or high-performance computing (HPC) facilities).

In the longer term, the framework can be extended to support orchestrated delivery of other types of services across the GÉANT/NREN fabric and the other e-infrastructures.

Similarly, in its current early stages, the PoC imposes minimal requirements on partner (e.g. NREN) domains. However, in the long term, partner domains are expected to function as service providers with a wider range of capabilities (e.g. service catalogue management).

In its adoption of a standards-based approach, the PoC ensures the widest possible acceptance and interoperability. However, it is also flexible enough that while adhering fully to TMF specifications itself, it allows communications with parties that are not.

From its initial focus on MS Azure ExpressRoute connectivity service, the reference implementation is, by design, sufficiently flexible and extensible to apply to a wide range of e-infrastructures and service providers, and to further aspects of integration, to the benefit of end users and service providers alike.

Appendix A Details on Managing ExpressRoute Tagging at GÉANT Ingress

The NetherLight equipment uses virtual switches per port and therefore all VLANs are available and there is no issue with Microsoft defining the S-VLAN tags. However, on handover to the GÉANT ports there could be instances of VLANs already in use, so a retag action would be required – in this case within NetherLight.

Figure A.1 below shows the possible options for transparent or manipulated tagging on the connections; all options can be handled natively by either MDVPN or BoD.

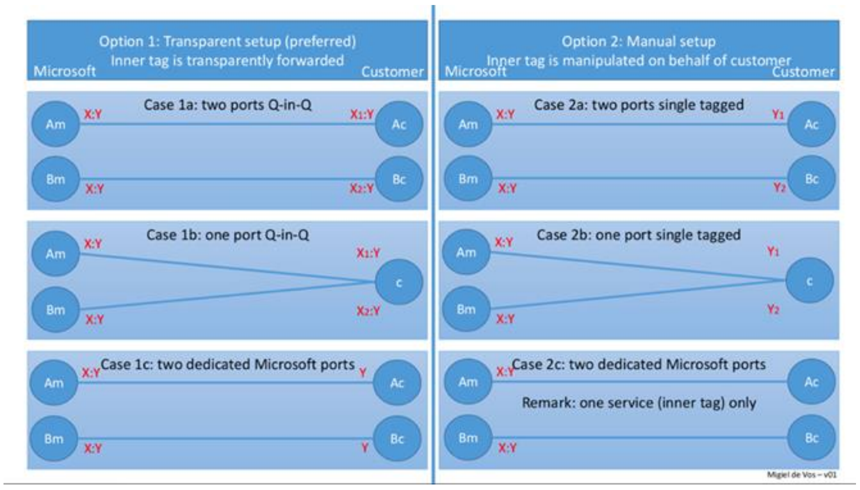
Option 1: Transparent setup (preferred)
Inner tag is transparently forwarded
Microsoft
Customer
Option 2: Manual setup
Inner tag is manipulated on behalf of customer
Microsoft
Customer
Case 1a: two ports Q-in-Q
Am
Bm
Ac
Bc
X:Y
X:Y
X1:Y
X2:Y
Case 2a: two ports single tagged
Y1
Y2
Case 1b: one port Q-in-Q
C
Case 2b: one port single tagged
Case 1c: two dedicated Microsoft ports
Y
Case 2c: two dedicated Microsoft ports
Remark: one service (inner tag) only
Migiel de Vos – v01

Figure A.1: Options for ExpressRoute tagging within NetherLight for GÉANT ingress

References

[Activiti]	https://www.alfresco.com/platform/process-services-bpm
[Ansible]	https://www.ansible.com/
[ApacheCamel]	http://camel.apache.org/
[CloudSigma]	https://waw.cloudsigma.com/ui/
[EGIFedCloud]	https://www.egi.eu/federation/egi-federated-cloud/
[EOSC]	https://ec.europa.eu/research/openscience/index.cfm?pg=open-science-cloud
[Filiposka]	S. Filiposka, R. Łapacz, M. Balcerkiewicz, F. Wein, J. Sobieski, *Transforming silos to next-generation services*, IEEE EUROCON 2017 – 17th International Conference on Smart Technologies http://ieeexplore.ieee.org/document/8011210/
[GNInfraScvs]	https://clouds.geant.org/services/geant-cloud-catalogue/infrastructure-services/
[IETF-RFC6020]	https://tools.ietf.org/html/rfc6020
[IETF-RFC6241]	https://tools.ietf.org/html/rfc6241
[INDIGODataCloud]	https://www.indigo-datacloud.eu/
[INDIGO-VR]	https://github.com/CESNET/IndigoVR
[INTERLUDE]	https://wiki.mef.net/display/CESG/LSO+Interlude
[JunosSpace]	https://www.juniper.net/uk/en/products-services/network-management/junos-space-platform/
[MDVPNReg]	https://mdvpn-inventory.qalab.geant.net/mdvpn-service-inventory/index.jsf
[MDVPNWhitePaper]	X. Jeannin, T. Szewczyk, B. Jakovljevic et al., *GÉANT MD-VPN: Multi-Domain Virtual Private Network service - a seamless infrastructure for NRENs, GÉANT and NORDUnet* https://wiki.geant.org/download/attachments/48496767/GEANT%20MD-VPN%20White%20Paper%20Jun%202015.pdf?version=1&modificationDate=1440059645589&api=v2&usg=AOvVaw0JhSBl6_s-C8BeGMbeFKyA
[MEF-CE2]	https://www.mef.net/carrier-ethernet-services/ce-2-0-overview
[MEF-55]	Service Operations Specification MEF 55, Lifecycle Service Orchestration (LSO): Reference Architecture and Framework, March 2016 https://www.mef.net/Assets/Technical_Specifications/PDF/MEF_55.pdf
[MSAzureER]	https://azure.microsoft.com/en-us/services/expressroute/
[MSAzureER-GN1]	https://wiki.geant.org/download/attachments/59245309/operations-diagram-MS-GNT-NL-v1-0.pdf
[MSAzureER-GN2]	https://wiki.geant.org/download/attachments/59245309/GE%CC%81ANT%20ExpressRoute%20administrators%20guide_v0.02.pdf

[MSAzureER-GN3]	https://wiki.geant.org/download/attachments/59245309/GE%CC%81ANT%20ExpressRoute%20manual_v0.03.pdf
[NL-DDS]	https://agg.netherlight.net/dds
[NSIRequester]	https://github.com/BandwidthOnDemand/nsi-requester
[OGF-GFD-RP212]	OGF GFD-R-P.212 *NSI Connection Service v2.0* https://www.ogf.org/documents/GFD.212.pdf
[RESTW-GÉANT-bod]	https://github.com/damomeen/geant-bod-rest
[RESTW-nsi_conns]	https://github.com/damomeen/nsi_connections
[RESTW-nsi_ports]	https://github.com/damomeen/nsi_ports
[SONATA]	https://wiki.mef.net/display/CESG/LSO+Sonata
[Spring]	https://spring.io/
[TMFFrameworx]	https://www.tmforum.org/tm-forum-frameworx/
[TMFOpenAPIdocs]	https://projects.tmforum.org/wiki/display/API/Open+API+Table
[TMFOpenAPIS]	https://www.tmforum.org/open-apis/
[TMF-R16-5-1]	https://www.tmforum.org/resources/standard/ig1141-procurement-and-onboarding-of-virtualization-packages-r16-5-1/
[TMF-SID]	*GB922 Information Framework (SID) – R16.5* available from https://www.tmforum.org/information-framework-sid/
[TOSCA-NFV]	http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/tosca-nfv-v1.0.html
[TOSCA-TC]	https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca

Glossary

AAI	Authentication and Authorisation Infrastructure
AG	Aggregator
API	Application Programming Interface
AWS	Amazon Web Services
BGP	Border Gateway Protocol
BoD	Bandwidth on Demand – GÉANT Network Connectivity Service
BPM	Business Process Management
BPMN	Business Process Model and Notation
BSS	Business Support Systems
BUS	Business Applications
CE	Carrier Ethernet
CFS	Customer-Facing Service
CMDB	Configuration Management Database
CRM	Customer Relationship Management
CSAR	Cloud Service Archive
CSP	Cloud Service Provider
C-VLAN	Customer VLAN
DC	DataCloud
DDS	Document Distribution Service
E	Ethernet
E-LAN	Ethernet Local Area Network
E-Line	Ethernet Line
eNNI	external Network to Network Interface
EOSC	European Open Science Cloud
EPL	Ethernet Private Line
ERO	Explicit Routing Object
eTOM	TMF Business Process Framework
E-Tree	Ethernet Tree
ETSI	European Telecommunications Standards Institute
GCC	GÉANT Cloud Connectivity
HOT	Heat Orchestration Template
HPC	High-Performance Computing
IaaS	Infrastructure as a Service
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IM	Infrastructure Manager

IP	Internet Protocol
ITSM	IT Service Management
JRA	Joint Research Activity
JRA1	Joint Research Activity 1, Network Infrastructure Evolution
JRA1 T3	JRA1 Task 3, Integration with Other e-Infrastructures
JRA2	Joint Research Activity 2, Network Services Development
JRA2 T2	JRA2 Task 2, Service Provider Architecture
JRA2 T4	JRA2 Task 2, Service Provider Architecture
JRA4	Joint Research Activity 4, Application Services Delivery Development
L*n*	Layer *n*
LAN	Local Area Network
LDP	Label Distribution Protocol
LSO	Lifecycle Service Orchestration
LSP	Label Switched Path
MDVPN	GÉANT Multi-Domain Virtual Private Network – Network Connectivity Service
MEF	Metro Ethernet Forum – Standards Body
MIRP	Management Interface Reference Point
MPLS	Multi-Protocol Label Switching
MS	Microsoft
MTU	Maximum Transmission Unit
NETCONF	Network Configuration Protocol
NFV	Network Functions Virtualisation
NFVD	NFV Descriptor
NNI	Network to Network Interface
NOC	Network Operating Centre
NOW	Network Orchestration Wrapper
NREN	National Research and Education Network
NSA	Network Service Agent
NSD	Network Service Descriptor
NSI	Network Service Interface
OASIS	Organisation for the Advancement of Structured Information Standards
OCCI	Open Cloud Computing Interface
OSS	Operations Support Systems
OTC	Open Telekom Cloud
P2P	Point to Point
PA	Provider Agent
PaaS	Platform as a Service
PE	Provider Edge
PoC	Proof of Concept
QinQ	IEEE 802.1ad Ethernet networking standard
R&E	Research and Education
RA	Requester Agent
REST	Representational State Transfer
RFP	Request for Proposal
RFS	Resource-Facing Service
SA	Service Activity
SA2	Service Activity 2, Trust & Identity and Multi-Domain Services

SID	TM Forum Information Framework
S-key	Service Instance Key
SLD	Service Label Distribution
SPA	Service Provider Architecture
SPoF	Single Point of Failure
SSP	Self-Service Portal
STP	Service Termination Point
S-VLAN	Service VLAN
TAM	TM Forum Application Framework
TCP	Transmission Control Protocol
tLDP	targeted Label Distribution Protocol
TMF	TM Forum – Standards Body
UDP	User Datagram Protocol
UNI	User Network Interface
VLAN	Virtual Local Area Network
VM	Virtual Machine
VPC	Virtual
VPN	Virtual Private Network
VRF	Virtual Routing and Forwarding
VRR	VPN Route Reflector
WAN	Wide Area Network
YAML	Representative markup language to describe data-oriented systems