# FEDERATED TEST-BEDS FOR LARGE-SCALE INFRASTRUCTURE EXPERIMENTS
# FELIX EU-JP

Collaborative joint research project co-funded by the European Commission (EU)and National Institute of Information and Communications Technology (NICT) (Japan)

| | |
|---|---|
| Grant agreement no: | 608638 |
| Project acronym: | FELIX |
| Project full title: | ”Federated Test-beds for Large-scale Infrastructure eXperiments” |
| Project start date: | 01/04/13 |
| Project duration: | 36 months |


# Deliverable D3.5
# Consolidated Report From FELIX Development Activities

## Version 1.0

| | |
|---|---|
| **Due date**: | 31/03/2016 |
| **Submission date:** | 04/04/2016 |
| **Deliverable leader:** | i2CAT |
| **Author list:** | Carolina Fernandez (i2CAT), Umar Toseef (EICT), Takatoshi Ikeda (KDDI), Atsuko Takefusa (AIST), Jason Haga (AIST), Fumihiro Okazaki (AIST), Bartosz Belter (PSNC), Krzysztof Dombek (PSNC), Damian Parniewicz (PSNC), Roberto Monno (NXW), Gino Carrozzo (NXW), Vicent Borja Torres (iMinds), Brecht Vermeulen (iMinds) |

**Dissemination level**

| | | |
|---|---|---|
| ☑ | PU: | Public |
| ☐ | PP: | Restricted to other programme participants (including the Commission Services) |
| ☐ | RE: | Restricted to a group specified by the consortium (including the Commission Services) |
| ☐ | CO: | Confidential, only for members of the consortium (including the Commission Services) |

**<THIS PAGE IS INTENTIONALLY LEFT BLANK>**

# Table of Contents

# List of Figures

# Abstract

This document presents an overview of the development efforts carried out on the FELIX Management Stack (FMS) during the third year of the project. This document complements the information provided in the previous deliverable documents D3.1, D3.2, D3.3 and D3.4 by explaining which new features and extensions or modifications have been performed since the previous reports. This deliverable describes also the final status of the deployment of the modules from FMS and documents modifications on the installation and configuration steps per module. In the end we provide an analysis and conclusions of the work and its status during the last period.

# Excecutive Summary

Deliverable D3.5 aims to explain, from a high-level perspective, the most important developments carried out on all the FELIX software modules since the previous description in the deliverable reports D3.1 [1], D3.2 [2], D3.3 [3] and D3.4 [4].

We start with a brief explanation of the objectives and list of key functions of each software module in the FELIX stack, identifying those developed during Y2 and reported in the previous reports, and those new features or extensions carried out during this last period; some of them as part of the feedback provided by the Use Cases. We also describe the testing procedure followed to assess the new features or developments performed during Y3.

After the status of the FELIX software modules is described, the next section provides an overview of the status of the final stage of deployment and a description of any changes on their deployment, i.e. any new requirement, dependency or change in the installation or configuration procedures. We document later on the status of development-related tasks such as the organisation and tasks to manage the codebase and tools in the repository, and also the procedures to determine compliance and code quality and how this has evolved during this period.

Finally, in the last part of the document we summarise the status of the FELIX stack, provide our conclusions and describe what we identify as potential valuable to be reused by other initiatives, beyond the FELIX lifetime.

# 1 Introduction

As introduced in previous WP3 deliverables, the FELIX Management Stack consists of a number of Resource Managers (RMs) which are managed, monitored and authenticated at different levels (typically, ”master” and *base/child*); and provide users with the ability to access resources requested at each test-bed.

Summing up, the software modules in FELIX enable i) users with access and basic management of resources, and ii) administrators with advanced management and configuration of the computing and networking resources. The computing (XEN-CRM/KVM-CRM) and networking resources (SDN-RM, SE-RM, NSI-TNRM/GRE-TNRM) are managed by the (Master) Resource Orchestrators ((M)ROs), and their monitoring information is aggregated in the layer of (M)RO; where those layers can be stacked/set up recursively. The Monitoring System ((M)MS) and Authentication (M/CBAS) modules interact in a different way with (M)ROs to obtain general information on resources and to confirm or deny access, respectively.

Specific details for the design, configuration, installation and usage of each software module in FELIX can be found in previous WP3 deliverables. That is: (M)RO and XEN-CRM are explained in D3.1 [1]; (M)MS and related tools, in D3.2 [2]; SE-RM and TN-RM, in D3.3 [3]; and user tools, CBAS and public monitoring are described in D3.4 [4].

In the following sections we summarise the list of provided features from the previous and current reports and describe how we tested the proper working of those features. We identify then the modifications carried out in the software modules, as well as their installation and configuration procedures since the previous reports. We end this deliverable providing a summary on the final stage of development and deployment per module and island, respectively; along with our conclusions and the potential we foresee for the FELIX modules to be applied in similar initiatives beyond the lifetime of this project.

# 2 Modules

After having a broad look at the final stage of deployment, we describe in this section the final stage of development per software module.

Figure 2.1 depicts the software elements that are part of FMS and the communication expected between them.

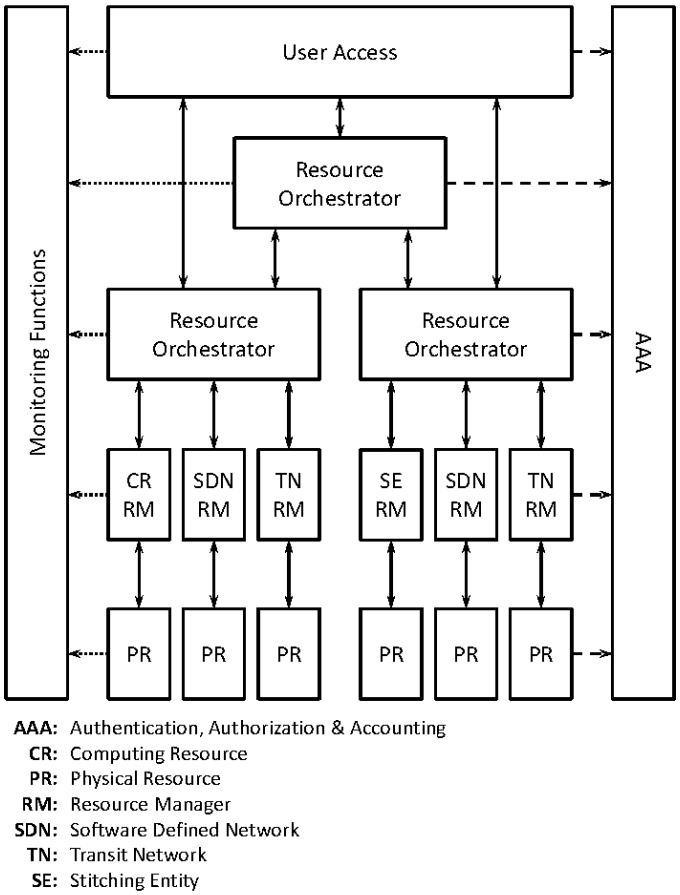


Figure 2.1: General architecture

For each module (XEN-CRM and KVM-CRM, SDNRM, SERM, NSI-TNRM and GRE-TNRM, M/RO, M/MS and M/CBAS) we will describe the list of features offered by each module and which of those have been newly added or are improved from last year, as well as the procedures we have followed to ensure proper behaviour on those features.

## 2.1 CRM

### 2.1.1 Description of the module

The Computing Resource Manager (C-RM) module allows the experimenter to provision and manage VMs on a number of physical servers running either the XEN or KVM virtualisation infrastructures. The details on its design

and building blocks, the exposed APIs and required RSpecs as well as the interconnections with other modules and workflows are all described in sections 3.2.1 and 3.2.2 on D3.1 [1]. The architecture and building blocks have not changed from D3.1.

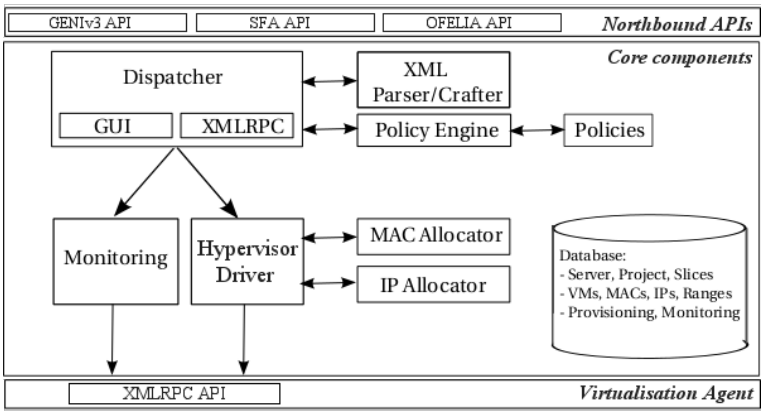


Figure 2.2: Components of the FELIX C-RM module

C-RM can work on top of two different kinds of commonly used infrastructures: XEN and KVM hypervisors. In order to coordinate with a specific server and hypervisor providing the machines for the experiments, C-RM works with different internal APIs, leading to two differentiated flavours of C-RM. The rationale of maintaining two versions of C-RM is to support already existing hypervisors and to be able to put those to work and integrate easily with other FELIX test-beds.

XEN-CRM, being based in the OFELIA Virtualisation Technology Aggregate, supports operations on VMs managed by an installation of XEN hypervisor (XEN-CRM). During Y2, the FELIX project worked on the extension of C-RM to extend support to KVM-based infrastructures, in an attempt to re-use Japanese existing infrastructure and also to address some of the requirements coming from the Infrastructure Domain Use Cases. This resulted in a new flavour of C-RM: KVM-CRM. Among other needs, the IaaS migration Use Case requires nested virtualisation, which is supported by KVM but not by XEN. By introducing KVM-CRM, the flexibility of the C-RM and the overall FELIX federated framework was enhanced.

Both flavours provide a northbound API supporting GENI-based northbound APIs (v2 and v3). The latter API (GENIv3 [5]) has proven to serve as a quick enabler for the integration with the Fed4FIRE First-Level Support (FLS) monitoring and the jFed client [6].

### 2.1.2 Features

The features offered by C-RM have been mainly developed during Y1 and Y2. We identify the latest set of features of C-RM into those existing at the time of writing the last deliverable (D3.1), as well as those added later.

#### 2.1.2.1 Previously supported features

- Ability to request, update and delete VMs
- Support for XEN-based virtualisation infrastructures
- Implements GENI v2, v3 (to account for fine-grained control of resources and easy integration) and OFELIA [7] custom northbound APIs
  - Among others, this provides separate operations for allocation, provisioning and start of resources

- Basic monitoring of the currently existing VMs per project and slice
- Configuration of information for servers, ranges of IPs and MACs and such
  - Available to the administrator through GUI
- Allows uploading user SSH keys and basic configuration in the VMs to enable PKI-based access and root access

#### 2.1.2.2 New features

- Support for nested virtualisation (KVM hypervisor)
  - To achieve that, the hypervisor driver and virtualisation agent (Figure 2.2) were substituted inside KVM-CRM
- Support for new disk images in KVM-CRM, among them Ubuntu 14.10
  - Possibility to pick different templates by passing the local path to the disk image

The development of the KVM-CRM was carried out during Y2 and was designed to be completely transparent to the functionality of the infrastructure. Thus, there were no modifications to the previously described functions in this new flavour. The Agent remains as an interface for managing virtual machines on the desired physical (or virtual) server/host and is still able to communicate with the hypervisor that is running on the physical (or virtual) virtualisation server. All management functions of the virtual machines are still available. This additional functionality did not affect the communication between the blocks, any of the exposed interfaces, or the type of RSpec passed.

Besides the inclusion of the new flavour, C-RM was subject to minor improvements and feature tuning, such as improving the background monitoring and management on expired resources, processing requests more exhaustively, accounting for the timezone in the log traces, enabling the contextualisation of multiple SSH user keys at a time and their inclusion in the *sudoers* group or producing more correct and succinct manifest after each provisioning; among others.

#### 2.1.2.3 Module testing

During the deployment of the two different blocks of Use Cases we have requested VMs to KVM-CRM in different installations (mainly at PSNC and AIST islands); where all new features have been used. Specifically, the Infrastructure Domain Use Case required nested virtualisation to properly operate; and the Data Domain Use Case provisioned VMs in KVM-enabled islands, using specific disk image to account for the needs of some applications, such as the controller and the SMOS viewer.

Therefore the Use Cases, acting as ”clients” that request specific functionalities, have been used also to implicitly ensure the proper working of the new features in C-RM. After each provision, the VMs provisioned were started and accessible using the user’s public key passed along with the request. Once VMs were deleted, the hypervisor started the deletion of the machine and associated resources, allowing further similar requests in other experiments.

## 2.2 SDNRM

### 2.2.1 Description of the module

The Software-Defined Networking Resource Manager (SDN-RM) module allows the experimenter to request and manage a defined subset of ports from those exposed by physical OpenFlow-enabled switches. The details of its design and implementation are described in D3.1 [1].

Besides managing OpenFlow switches, SDN-RM mediates between the software controller defined by the experimenter and the inner island software controller (FlowVisor). The former defines the list of rules to be inserted in the physical switches and the latter acts as a proxy, forwarding traffic from/to the appropriate controller.

Unlike CRM, SDN-RM comes in a single flavour and is deployed as it is in each of the FELIX islands. On the other hand, the SDN-RM module also exposes GENI-based northbound APIs (v2 and v3) while keeping the OFELIA [7] custom API for backward compatibility.

### 2.2.2 Features

The features of SDN-RM have been all developed during Y1 and Y2. There was no further work or requirements planned for Y3 regarding this module, as it already fulfilled the needs of the stack and experiments.

#### 2.2.2.1 Previously supported features

- Ability to request and delete OpenFlow resources
- Manual or automatic approval/denial of FlowSpaces
- Proxy between the experimenter and the domain controllers
- Implements GENI v2, v3 (to account for fine-grained control of resources and easy integration) and OFELIA custom northbound APIs
- Basic monitoring of the currently approved FlowSpaces per project and slice

#### 2.2.2.2 New features

No new features on SDN-RM were necessary or expected during late Y2 or Y3 on SDN-RM, as it already fulfils its goals. Thus, the main development work during this last period has been focused in bug solving and feature tuning. Some examples are the improvement of the background monitoring and management on expired resources, a more exhaustive and correct processing of flowspace requests, accounting for the timezone in the log traces and such.

#### 2.2.2.3 Module testing

In the same way as CRM, the SDN-RM module has been manually tested through the implementation of the FELIX Use Cases. Every UC requested SDN resources which were correctly provisioned (to enable the chosen ports and matching conditions) and deleted (to free them immediately for other usage), and consequently there were no further feedback or improvement requests.

## 2.3 SERM

### 2.3.1 Description of the module

The Stitching Entity Resource Manager (SE-RM) module is responsible for the stitching of two domains controlled by the FELIX software stack: the SDN domain (provided by SDNRM) and the inter-domain transport network services (provided by TNRM module or configured statically in another way). Specifically, the traffic from the user coming or arriving to the SDN domain must be directed appropriately to and from the TN domain, so inter-domain connectivity is feasible.

SE-RM, as any other RM in FELIX, exposes a GENIv3 API northbound interface. This allows to reserve and allocate resources of a given stitching switch (i.e., a pair of ports and VLANs), which must be located between one or more SDN switches and the transport network switches.

The main characteristic of the SE-RM module is that it is able to maintain one-to-one relationship between the flowspace defining the traffic isolation within the SDN network slice (i.e., FELIX supports VLAN/port based flowspaces) and the inter-domain transport network service instance (i.e., VLAN terminated service offered by large network provider), which is used for the FELIX experimenter traffic transport between SDN domains. More details of its design and implementation are described in D3.3 [3]. The overview of SE-RM module is presented on Figure 2.3.

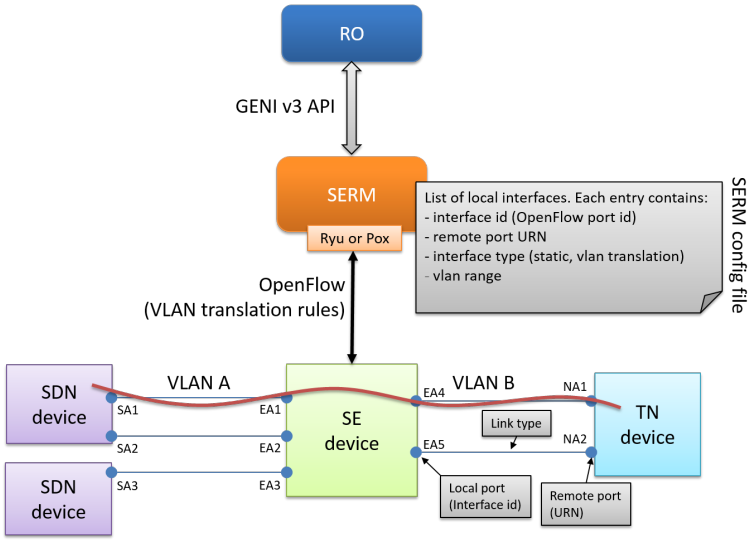


Figure 2.3: SE-RM overview

The SE-RM module was introduced in the FELIX architecture in order to allow a more automatic and seamless usage of inter-domain connections. In practice, the transport networks have their own constrains resulting in a number of limitations for the definition of experiments or slices, e.g., limited range of VLANs available between Europe and Japan (i.e., VLANs: 1779-1799), some VLAN tags must be excluded because of network switch internals or third party services, and such. SE-RM is dedicated to handle such constrains and limitations in order to achieve a fully automatic provisioning of multi-domain links across the virtual infrastructure. To meet that end, the SE-RM module bases its work on using those resources really available within the edge devices of the transport network, and by decoupling the virtual network from the inter-domain transport network (that is, different VLAN tagging can be used for the virtual network within SDN domains and for any related inter-domain transport services).

### 2.3.2 Features

#### 2.3.2.1 Previously supported features

The deliverable D3.3 [3] described the design, functions, interfaces and workflows of the SE-RM. The SE-RM functionalities at the end of Y2 of the FELIX project were as listed below:

- GENIv3 API used as northbound interface to manage the allocation of a pair of stitching switch ports and links, or stitching rules
- East interface (SDN domain) accepting VLAN-based flowspaces traffic coming from the SDN domain
- West interface (TN domain) accepting VLAN-terminated inter-domain network services
- Support for stitching rules in the form of VLAN translation mechanism

- Southbound interface communicating with the APIs of two different OpenFlow controllers, in order to add/remove flow entries and represent stitching rules, in the stitching switch:
  - Ryu controller, using northbound REST/JSON API
  - POX controller, using northbound XMLRPC API
  - Placeholder for any new drivers of control/management plane protocols towards stitching switch

#### 2.3.2.2 New features

Throughout Y3 the SE-RM development was focused on small improvements, bug fixing and adapting to improvements in the definition of resources through the GENIv3 RSpec. Yet, a new feature was added to SE-RM during this period:

- Locally determining the VLAN tag used for static VLAN-terminated links on inter-domain side

#### 2.3.2.3 Module testing

The SE-RM module and the software associated (e.g. the OpenFlow controllers) were tested in three different ways:

- Single module testing performed using OMNI-client and contacting directly SE-RM
- Single module testing performed using the jFed GUI
- FELIX Use Case execution, where (M)RO has identified SE resources and configured in the SE-RM module

SE-RM was deployed and validated with a set of physical switches (Juniper MX, NEC PF5240, HP Procurve) as well as with Open vSwitch. Validation was considered to be successful once i) the proper pair of ports of the stitching switch were reserved, ii) the proper OpenFlow flows were installed and iii) VLAN tag translation occurred in network frames going through the stitching switch.

## 2.4 TNRM

### 2.4.1 Description of the module

As described previously in D3.3 [3], the main function of the Transit Network Resource Manager (TN-RM) is to perform the network provisioning between SDN domains in particular islands or domains, and between them. As with other FELIX RMs, TN-RM can be used either as a stand-alone module or contacted by the (Master) Resource Orchestrator (RO), which orchestrates different types of Resource Managers in the island. The TNRM receives requests from (M)RO and notifies back the (M)RO about status of the resources and of the successful or failing events. The TN-RM module is based on the concepts of the GENI architecture and the OFELIA Aggregate Managers. It uses the GENIv3 API to provide operations of reservation (*allocation*), provisioning (*activation*) and others to manage the resources, as well as to provide easier federation and to maintain consistency with the implementation of other FELIX RM modules.

The TN-RM leverages on two technologies to provide inter-island connectivity. The first, described in the previous deliverable, is the Network Service Interface (NSI) provisioning service that allows to establish a path via Research and Education Networks providing NSI-based connectivity, i.e. it acts as a proxy between the SDN and NSI based networks. The second technology is the Generic Routing Encapsulation (GRE) protocol to set up tunnels over the Internet. This was added during the third year of the project (Y3) in response to recommendations from the Y2 Project Review.

The added value of this module is that it maintains and extends the technology-agnostic notion of the FELIX architecture. The NSI-based paths are able to provide higher bandwidth and quality than the Internet and

this module allows experimenters to take full advantage of these networks. The addition of the GRE tunnelling extends the functionality of the FELIX architecture and enables other SDN islands that may not have access to NSI-based domains to join the FELIX federated framework. Moreover, it adds value to users as it allows them to experiment with different combinations of network interconnectivity and to assess how these combinations affect the performance of their applications.

### 2.4.2 Features

The following subsections summarise the features supported by the TN-RM module. The enhancements of the module during Y3 are identified, and their functionalities have been fully tested, concurrently with the other modules and systems.

#### 2.4.2.1 Previously supported features

The deliverable D3.3 [3] described the design, functions, interfaces and work-flows of the TN-RM. Below is a brief summary of the main key functionalities available at the time of its writing:

- Management of transit network connectivity between SDN islands
- Support for GENIv3 requests and workflow, which were previously mapped to the corresponding NSIv2 counterparts
- Ability to list available resources, allocate, modify, provision, delete, and emergency shutdown links

#### 2.4.2.2 New features

During Y3, the key new feature added to TN-RM has been the Generic Routing Encapsulation (GRE) tunnels over the Internet.

To this matter, virtual GRE switches are instantiated using an OpenFlow controller RYU [8] and Open vSwitch [9]. The TN-RM module controls the traffic encapsulation via the Ryu REST API. In FELIX, multiple paths between the same SDN islands, sharing a single GRE tunnel, are distinguished by different VLAN tags. Optimisation using smaller MTU size is required to overcome the reduction in communication throughput due to the attachment of GRE-related data to each packet. The ability to create GRE tunnels was added to increase the flexibility of the type of transit network connections that can be made and in response to Y2 project review recommendations.

#### 2.4.2.3 Module testing

The TN-RM functions were validated by deploying the module in the FELIX testbed and leaving it ready for internal experiments and the set up of the FELIX Use Cases. TN-RM was successfully used to request and provision a NSI-based path between Japan and Poland. This path was subsequently used in the Infrastructure Domain Use Case. At the conclusion of the each experiment, the network link was terminated and could be re-provisioned for other experiments.

## 2.5 (M)RO

### 2.5.1 Description of the module

The (M)RO module provides experimenters with functions to provision resources (in slices) and to monitor both physical and virtual resources within the FELIX test-bed and associated infrastructures.

The design of the (M)RO allows it to work in two different levels: **Master Resource Orchestrator** (MRO) or **Resource Orchestrator** (RO). This is achieved through a couple of configuration parameters, and allows to act either as a top-level software module, capable of overseeing the whole infrastructure within a *zone*; or to act as a top-level aggregate that manages the RMs within an island or domain. The main difference is the kind of

resources managed by the module, rather than its position in the FELIX architecture layout. Specifically, the RO takes care of the computing, SDN (OpenFlow) and stitching resources in each island of the FELIX facility, performed via communicating with the technology-specific Resource Managers; and provides the first level of monitoring on those resources to the MS. On the other hand, the MRO manages the ROs in the federation, realising the hierarchical approach implemented in the project, and aggregates the monitoring information that is passed to the MMS.

The interface exposed to the users implements the GENIv3 (SFA) standard. This allows to manage the resources, that is, retrieving the list of all available resources (servers, switches or devices), allocate/reserve them, provision/activate them and start, stop, restart and release resources. After those heterogeneous slices are activated it is possible to monitor the experiments and the status of their encompassed resources. Consequently, (M)RO acts as the entry point for each domain and *zone*, accepting requests and enforcing the correct work-flow. It validates first each request against its expected schema, then (M)RO extracts the user certificate in order to authenticate and authorise the user through the AAA system. If the user is authorised to perform the required actions, then the request is analysed, the context evaluated and individual requests are generated and directed against the competent RO or MRO.

One of the main functions of (M)RO is the management of the underlying topology. In order to maintain a consistent view of the physical devices, both synchronous and asynchronous (time-based) procedures have been developed. The retrieved information is stored locally in the internal database, as well as sent to the (Master) Monitoring System to instrument the component on the resources to be (periodically) monitored.

It is worth noting that the unified management of heterogeneous resources and the provisioning of end-to-end services over distributed network segments, while abstracting inner details to the final user, are a prerogative of the FELIX Management Framework within the FIRE and GENI communities; where it can provide value where aggregated information of the RMs or its top-level coordination are expected. The design of the (M)RO is the realisation of this strict requirement of the project. This design and the associated features will advance the current status of the art by providing the island administrator the capability to facilitate a potential federation of his/her resources over a large-scale infrastructure, and the experimenter to request resources in a lighter way.

### 2.5.2 Features

The following subsections summarise the features supported by the (M)RO module from a high-level perspective. We highlight the key enhancements of the module in Y3, in which the (M)RO has been deployed in the FELIX testbed and its functionalities fully tested along with those of the other modules and systems.

#### 2.5.2.1 Previously supported features

The deliverable D3.1 [1] described a complete overview of the functions, the design, the interfaces and the work-flows of the (M)RO developed within the second year of the project.

Here below we provide a brief summary of the main key functionalities we presented in the document.

- Management of different kind of resources (i.e computing, SDN, stitching and transport network resources) interacting with the Resource Managers
- Mediation between the user and the technology-specific Resource Managers to reserve, configure, monitor, release and operate on the virtual slices
- Maintenance of an updated high-level and cross-island topological view
- Management of end-to-end services initiated on the federated testbed and coordination of the correct sequence of actions to instantiate the service
- Interaction with the Monitoring System to provide the details of the physical topology and instrumentation of the measurements of the virtual resources

#### 2.5.2.2 New features

The functionalities of (M)RO have been extended during Y3. During this period the development of the expected missing features was combined with the feedback from the internal usage of (M)RO as the entry point for the requests to define the resources for the Use Cases in the FELIX testbed. This has allowed us to refine some existing features, the aspects themselves of the initial design or even implement new functions.

Here we present the new features introduced in the (M)RO code:

- Verification of user credentials before performing any action on the underlaying Resource Managers. This exerts a first control access based in the credentials issued by the CBAS module and provided by the user, and is carried out prior to even processing the request within (M)RO
- Asynchronous (time-based) procedures have been introduced to retrieve the status of the managed resources and to monitor their utilisation in order to update the internal database. This is a key aspect for all the long-term requests, i.e. the vlan assignment in the transport network domain
- Implementation and integration with a Policy Engine to enforce access control on criteria specified or chosen by the administrator
- More complete information on the monitoring details regarding the underlying links and being sent to (M)MS
- Simple CLI and GUI access to show basic data, i.e. the configured peers at each (M)RO and the VLANs requested to TN from the (M)RO. This is integrated with access-enabled policy that depends on the request originating address, e.g. to show information only locally
- Further configuration options to define which information make available to the users (e.g. hide or show SE resources to allow explicit requests).
- Abstraction of the stitching and transport network resources to the users.
  - This is the scope of the *virtual links*, which allow the experimenters to request links through the TN domain, between SDN-based resources in different islands. An experimenter can issue requests with different levels of granularity, i.e. from the most basic request (without the need to exactly known the resources in the middle of the endpoints, or the exact endpoints) to a explicit request of every resource in between (in case absolute control of the underlying resource is required).

We consider the implementation of the *virtual links* as the key feature developed during Y3; as it translates into a much easier request to the final user. Originally, (M)RO allowed full, explicit requests (case 3). This was extended during Y2 to accept requests without the hidden SE resources (case 2). Finally, during Y2 and Y3, case 1 was supported to accept the simplest requests from the user. In this case the user identifies just the computing, SDN-based resources and the domains to be linked; leaving the identification of the underlying STPs (Service Termination Points).

Summing up, the *virtual links* provide support in (M)RO for the following kind of requests:

- Implicit requests in various degrees of granularity or completion:
  - a) Requests containing computing and SDN-based resources, when the experimenter does not care at all about specific ports or STPs
  - b) Requests containing computing, SDN-based and TN-based resources. This allows the experimenters to request specific STPs at each end domain; but not to worry about defining the ports at the SE device

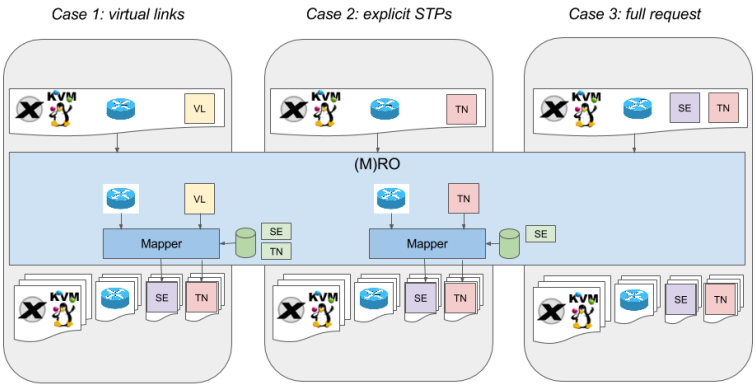


Figure 2.4: MRO: overview on the working of virtual links

- Explicit, low-level detail requests (with computing, and SDN, SE, TN based resources), so the users can request specific ports in the stitching plane and STPs at each domain

An example of a simple request and its translation after being interpreted by (M)RO is provided in "*MRO (virtual links)*", within the "*Appendix*" section.

#### 2.5.2.3 Module testing

The functions from (M)RO have been validated by making the module available per islands of the FELIX test-bed, so it is used for experiments and Use Cases. Its northbound API has been used every time a slice was created across some of the islands in Europe and Japan. The feedback provided by the definition of the experiments has been key to improve the internal working of (M)RO, as well as the format of the RSpecs supported.

The provisioning of the reserved resources has been performed using ad-hoc configuration parameters. The connectivity and reachability between the servers used to deploy (M)ROs have also been tested as well to assess successful interworking. The integration between (M)RO and the (Master) Monitoring System or (M)MS has been verified using the graphical section of Expedient implemented for the (M)MS module.

## 2.6 GUI

### 2.6.1 Description of the module

The GUI module for FELIX encompasses a number of different tools, according to its specific objective and public. The graphical tools used in the FELIX project are listed below:

- jFed [6] tool allows to easily reserve (*allocate*), activate (*provision*) and manage resources of each island from its graphical interface
- Expedient is used to access the monitoring summary of the resources in a slice (as an experimenter) and to configure the CRM and SDNRM modules
- (M)RO small, simple GUI offers quick information on the list of associated peers (whether other ROs or RMs) and the list of VLANs requested to the TN domain from a specific (M)RO

### 2.6.2 Features

During Y3, some effort was dedicated to the proper configuration and extension of the mentioned GUI tools to properly interact with FMS (FELIX Management Stack). It is worth noting, though, that we decided not to dedicate

extensive effort on completely redefining the source base of the GUI to support the API offered by (M)RO and the RMs [4]; but rather to dedicate most effort on finalising and improving the key software modules from FMS and the Use Cases.

An alternative GUI is provided to the experimenters in the form of jFed, a framework widely used in the FIRE community and actively maintained. Such framework is composed of multiple tools that allow, among others, the provisioning and management of slices or experiments to the final user. jFed offers a well-tested set of tools that already implement the same API as the FMS and acts as an inter-domain federation trust, granting access to multiple FIRE experimental infrastructures.

#### 2.6.2.1 Previously supported features

At the time of writing the previous WP3 deliverable [4], some features were already implemented and integrated with the Expedient GUI [10], namely:

- Features inherited from the OCF stack (management of projects, slices and resources lifecycle using custom APIs on legacy RMs; log history, user and permission management by administrators)
- Monitoring section to show resource metrics per slice

#### 2.6.2.2 New features

During Y3, most work related to the GUIs was accomplished in the form of integration with already developed features or tools, but also some minor development aimed to quickly check resources managed by each island and FELIX. That is:

- Integration of jFed with the FELIX test-bed, in order to interwork with the available infrastructures in FELIX
- Allow log-in through CBAS certificates in Expedient and show status of CBAS
- GUI in (M)RO to have a quick glance at the list of peers added to the specific (M)RO

Firstly, the integration of FELIX with jFed made possible to send requests to RMs existing in the FELIX testbeds. This was achieved by including specific pairs of endpoints and URNs to the list managed by the jFed framework. The figures below show some steps on the definition of experimenting slices: one consisting of computing, SDN-based and TN resources; and the other of a simple VM.

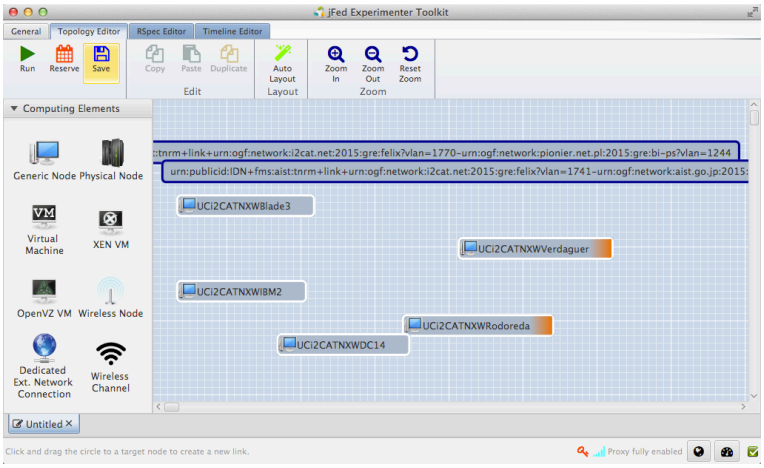


Figure 2.5: Defining resources in jFed

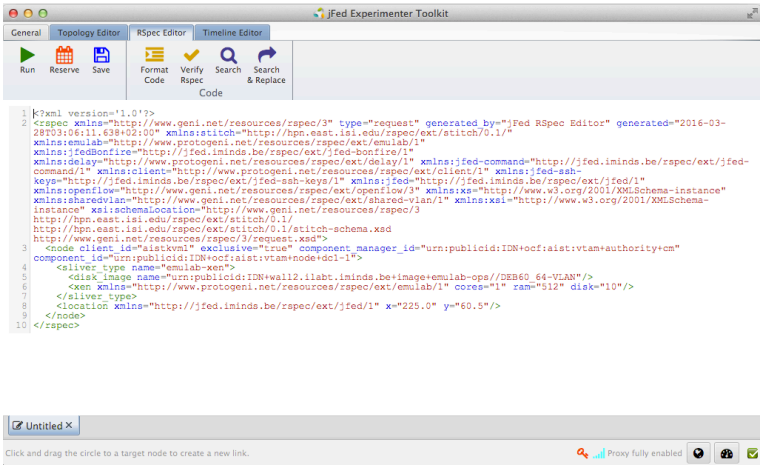


Figure 2.6: Defining resources in jFed

On the other hand, it was made possible to access the Expedient user interface by using properly generated (and trusted) certificates. Figure 2.7 shows the three possibilities given to an experimenter: a) log in with user and password (Expedient legacy), b) log in using certificate generated by corresponding clearinghouse, and c) directly through the jFed tool. A direct link is provided through Java Network Launching Protocol (JNLP), via a specific XML file, in order to run the *jFed Experimenter GUI* Java application from the browser.

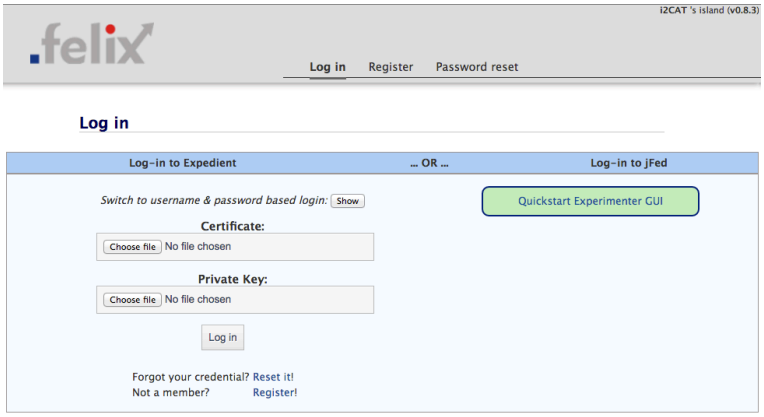


Figure 2.7: Expedient log-in screen

Finally, a simple GUI was added to (M)RO in order to have a quick look at the list of peers added to a specific instance of MRO or RO. This is a simple way to consult which RMs or ROs are managed by each orchestrator, e.g. to asses quickly where to ask about specific resources when using a CLI-based client.

Figure 2.8 shows the peers (ROs) managed by the European MRO, and for each of them, the list of the lower layer (RMs) is also returned. Figure 2.9 shows the peers (RMs) managed by the i2CAT RO. This information can be accessed from GUI and CLI, each with a format of its own. In the case of the graphical interface, the following address schema returns the information just mentioned:

```
https://<island_mro_host>:18440/gui/peers
https://<island_mro_host>:8440/gui/peers
```

### 2.6.2.3 Module testing

Similarly to other FELIX components, the capabilities provided by each GUI have been checked and tested through


Figure 2.8: Quick-access GUI in MRO: list of peers

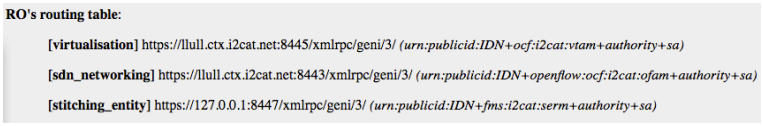


Figure 2.9: Quick-access GUI in RO: list of peers

manual use of each graphical tool.

The integration with jFed was followed by each RM developer in order to spot communication or permissions problems when defining the slice or reserving the resources, respectively.

The log in using certificates has been checked for some FELIX islands, and the invocation of the jFed client from the Expedient GUI was also conveniently checked manually and fixed after few iterations.

Finally, the GUI in (M)RO is simple and the testing was done concurrently with its development, so as to correctly retrieve peers from the (M)RO database and present them to the user in the expected format, independently of the call being made against GUI or CLI endpoints.

## 2.7 AAA

### 2.7.1 Description of the module

A comprehensive authentication and authorisation mechanism is of vital importance for any SDN experimental facility (SEF) that allows the use of finite computational and network resources for designated R&D purposes. An access control mechanism for a SEF serves a dual purpose. First, it enforces the application of policies; where access control to SEFs is required to enforce the usage policies on legitimate users and restrict access for others. Second, it ensures that SEF remains operational and experiment results are not affected by misbehaviours.

Certificate-based AAA, used by C-BAS, fulfils not only the basic needs of SEFs, i.e. enabling fine-grained access controls upon critical resources, but also overcomes inherent shortcomings of legacy solutions like LDAP and Kerberos, such as relying on single authoritative source of trust, inflexible system of authorisation, cumbersome process of synchronization of AAA entities to realise federations, and so on. In addition, the evolved architecture of C-BAS makes it secure against disruptions and interference from attackers and enables the support of different member roles and permissions. Furthermore, C-BAS is highly scalable and suitable for large experimental facilities and their federation. C-BAS is extensible through plug-ins, autonomous to minimize system administration efforts and modular to ease software maintenance and further developments.

To the best of our knowledge, C-BAS is the only open source AAA solution for SEFs that is actively being maintained and developed, and which is freely accessible to the research community. There are few other clearinghouse implementations that have been developed for particular SEFs and are often restricted to specific deployments or requirements. For example, Expedient [10] is a GENI control framework with a tightly coupled clearinghouse and, therefore, lacks in terms of distribution and flexibility when compared to C-BAS. Similarly, the ProtoGENI [11] clearinghouse has been based on a particular Slice Facility Architecture (SFA) [12] and designed to support only small-sized federations. Likewise, the GENI clearinghouse [13] with restricted dissemination policy implements its own clearinghouse API which is followed within the GENI federation. In contrast, C-BAS access interface supports the Common Federation API version 2 (FAPIv2) [14] set of standard APIs that any GENI-compatible Federation should offer.

When adopted from EU ALIEN project, C-BAS was just a simple clearinghouse with limited functionality. Within FELIX it has been enormously extended to fulfil the needs of the projects. Major extensions of C-BAS with FELIX include full support for FAPIv2, compatibility with GENI and FIRE tools (e.g. OMNI, Expedient, jFed, etc.), support for federation with other GENI/FIRE testbeds, development of Admin Tool to ease management tasks, support for speaks-for and credential delegation, and flexible management of member credentials like membership extension, revocation or managing privileges.

### 2.7.2 Features

#### 2.7.2.1 Previously supported features

C-BAS implements all the needed functionality of a clearinghouse in FELIX. It offers (i) an intuitive user access control, which is key to bring experimenters to SDN experimental facility (SEF); (ii) distributed authoritative source of trust that is inevitable in scenarios with geographically dispersed resource managers; (iii) flexible and extensible system of authorisation to ease the management of large-scale experiments; (iv) straightforward process for joining or leaving a SEF federation; (v) enforcing local policy regarding trust establishment with federating SEFs; and (vi) compatibility with software tools and APIs developed by related initiatives like GENI and FIRE.

#### 2.7.2.2 New features

During Y3, new features related to AAA were deployed on different modules. We highlight the following:

- Support for speaks-for credentials within C-BAS
- Integration of C-BAS with jFed
- C-BAS Admin Tool
- Autonomous exchange of root certificates
- Policies enforcement

**Speaks-for credential**

Speaks-for credential is primarily intended for supporting tools that "speak-for" their users. Speaks-for credentials allow a tool to be authorised through the user's privileges. Although the user is accountable for each action taken, the information about the tool used is also logged in the system. This way, speaks-for credentials allow for tracing which tool was used to perform any operation on the experimental facility. Speaks-for credentials play an important part in the recursive orchestration supported by FELIX. For example, they allow MRO and RO to replay user requests for resources requested to RO/RMs and provide requesting user's credentials as means for authorisation. In such cases users grant permission to involved (M)ROs to speak on their behalf by providing their speaks-for credentials.

GENI supports two credential formats, i.e. SFA (Slice Facility Architecture) [12] and ABAC (Attribute-Based Access Control) [15]. The support for SFA format is mandatory, while ABAC is an encouraged alternative. Complying with GENI, FELIX employs SFA credential format for authorisation purposes. However, SFA format does not support speaks-for type of credentials. Due to this limitation of the SFA format, it was decided to use ABAC format for speaks-for credentials. This decision led to the implementation that supports ABAC format in (M)RO and RMs, which enabled them to perform authorisations based on both SFA and ABAC credential formats.

With ABAC authorisation, decisions are made based on rules where actors (or principals) prove possession of required attributes for accessing resources. This way, a principal is mainly the subject of authorisation decisions with a certain attribute set. Within GENI, ABAC rules are visualised using the following graphical conventions. There, a principal is represented by an oval shape labeled with its name. An attribute is represented by a rectangle containing the principal that asserts the attribute and the attribute name in dotted notation. Finally an arrow connects a principal with its attribute, indicating the possession. For example, in Figure 2.17, Bob asserts an ABAC rule which assigns speaks-for attribute to an RO.

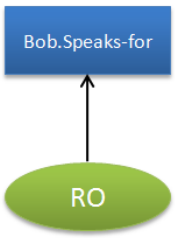


Figure 2.10: ABAC credential issued by Bob to RO assigning it with Speaks-for attribute

To maintain flexibility in FELIX, and to allow recursive resource orchestration, users issue speaks-for credentials to all entities which are trusted by its clearinghouse, aka C-BAS. Figure 2.11 (a) graphically depicts such speaks-for credential. To complement this, C-BAS issues an ABAC credential (see Figure 2.11 (b)) to its local RO assigning it with the "*Trusted*" attribute. This would enable RO to speak for user Bob when requesting resources from local RMs, as depicted in Figure 2.11 (c).

Similarly, when a user from one island reserves resources from another island, a number of ABAC credentials are employed to realise speaks-for functionality, as shown in Figure 2.12 (a-d). When the RO of the foreign island provides all these credentials to any of its local RMs, a chaining of these credentials is performed, as shown in Figure 2.12 (e), to verify that RO is authorised to speak for user Bob.

A small utility program has been developed, based on GENI library, that facilitates the generation of different ABAC credentials. In addition, a mechanism was implemented to verify ABAC credentials and attributes.

**Integration of C-BAS with jFed**

Since jFed is integrated with the FELIX test-bed, it is possible to log-in using certificates signed by the C-BAS

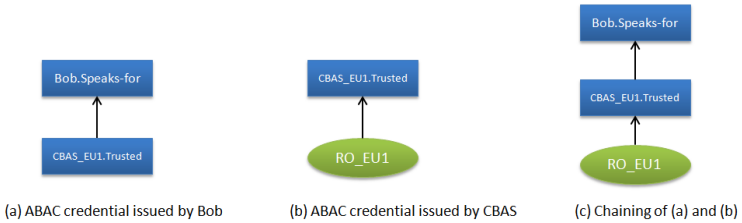


Figure 2.11: ABAC credential used in FELIX to realise speaks-for functionality within local island

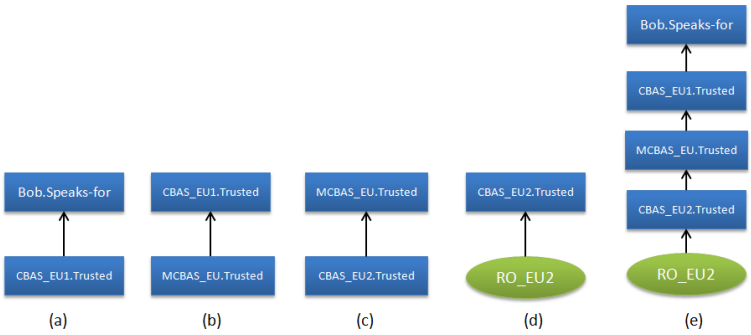


Figure 2.12: ABAC credential used in FELIX to realise speaks-for functionality across EU islands

authorities. For any client (e.g. jFed), to be *integrated* means that it shall be (i) able to identify and reach the infrastructures, (ii) trusts the infrastructures and the infrastructures trust it, and iii) can correctly interact with the APIs exposed by the infrastructures. To fulfil the conditions expressed above, the jFed development team added for each FELIX infrastructure the list of RMs and (M)ROs containing their endpoint and URN; and the procedure of trusting was performed in both sides.

An experimenter can thus use a certificate provided by the iMinds authority [16] or request a certificate provided by the C-BAS of a particular FELIX island by submitting a request through a simple form [17].


Figure 2.13: Log-in with jFed

Using the jFed client, either from the jFed website [6] or using the desktop version, the user provides its certificate and private key by means of a "*.pem*" file (see Figure 2.13). After verification, jFed will enable the user to request resources.

**C-BAS Admin Tool**

In order to facilitate administrative tasks for C-BAS, a Java-based application was developed. This standalone application (named C-BAS Admin Tool) offers a way to manage clearinghouse members, slices and projects. In addition, it offers an interface for browsing event logs through C-BAS for accountability purposes. This is a self-contained application that only requires Java Runtime Environment to run. Figure 2.14 shows log-in screen for Admin Tool, which asks for following pieces of information.

- *C-BAS Host Name / IP Address*: address pointer to the C-BAS running host. Default value is "*localhost*"
- *C-BAS Host Port*: TCP port where C-BAS is listening for connections. Default value is "*8008*"
- *Root Member Certificate*: certificate generated during C-BAS installation/configuration, usually located at *<path_to_cbas>/admin/root-cert.pem*
- *Root Member Certificate Key*: generated along with the above certificate, can be found at *<path_to_cbas>/ admin/ root-key.pem*

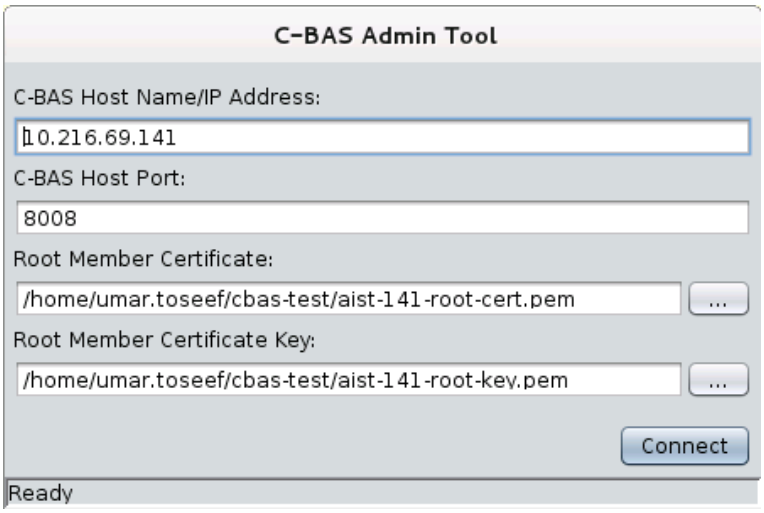


Figure 2.14: Log-in screen of Admin Tool

Once connected with C-BAS, a number of administrative tasks can be performed on clearinghouse objects. An example can be seen in Figure 2.15.

**Autonomous exchange of root certificates**

In order to facilitate the exchange of root certificates with peer islands, C-BAS has an option to enable pulling certificates from its master C-BAS. In this aspect, master C-BAS serves as a repository of root certificates of its peers, where those certificates are added manually by the administrator.

**Policies**

Policy enforcement has been added to FELIX to offer differentiated service to users, based on specific criteria and having been previously defined by the island or domain administrator. For instance, users owning a credential issued by a specific domain may be granted or denied access; or, assuming a pay-per-use schema, a user with a given membership or associated quota could access a different type or kind of resources. The development of a Policy Engine (PE) inside (M)RO in FELIX is a simple proof of concept that could be extended to fulfil other needs.

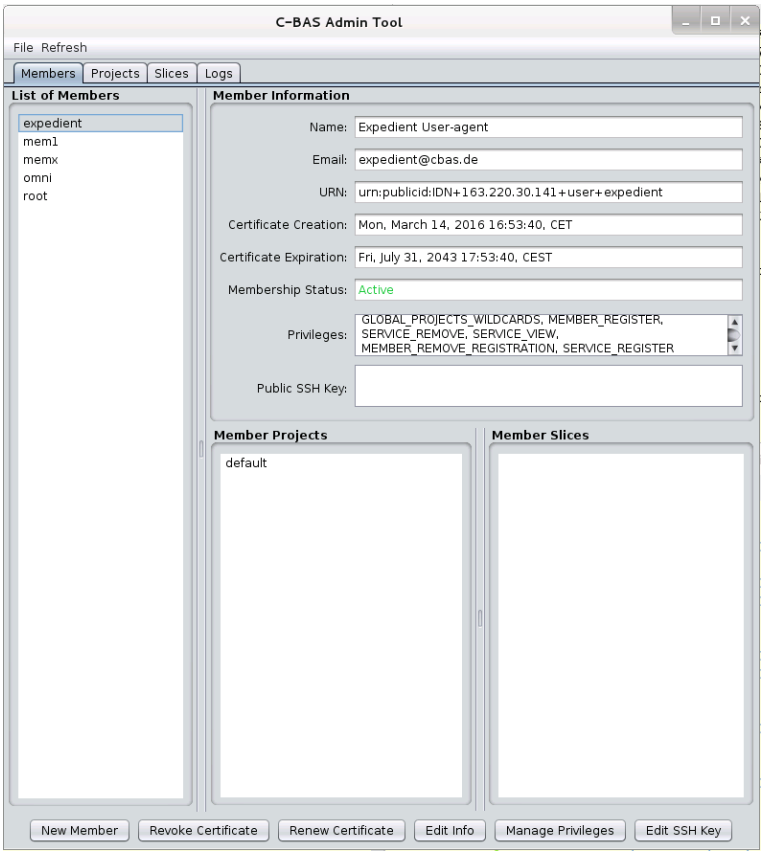


Figure 2.15: Main screen of Admin Tool

Although credentials provide a first level to filter the requests through authentication and authorisation, a PE called pyPElib [pyPElib] and developed within the OFELIA project has been integrated into FELIX. This component provides a new level of management of the requests, leaving island managers with the ability to define conditions that will be evaluated in order to restrict or grant access or resources to the experimenters.

When a request arrives to FELIX (M)RO, some fields of the request are first analysed. If this is successful, (M)RO starts preparing for the method issued by the user, and the credential is authorised and then analysed by the engine. The fields to be checked and the action to undertake are determined by the administrator of each infrastructure; who can choose fields of the request or the credential such as the remote address of the user connecting to the FELIX service, the user agent or the expiration date of its credential.

The conditions to be checked by the engine are internally treated as rules, which are composed of one or more *conditions*, a *return value* and optionally a *deny message* or other conditions. An example is given below:

```
if (cred.privilege != premium and vm.number > 5)
then deny denyMessage Only premium users can request more than 5 VMs
```

The rules evaluate one or more conditions or values, e.g. of the credentials and request. The return value can be "*accept*" or "*deny*". In this case, if the user has no premium privilege and performs a request of more than 5 VMs, the policy denies the request, informing the user about the details on the policy.

Figure 2.16 shows how the Policy Engine works. When some initial data (such as the request or credentials) is parsed, the PE checks the rules and tries to match them in order against the parsed data. Once a rule is matched, the value is returned and depending on it, the request will be accepted or denied.

pyPElib includes a persistence engine in order to store and manage the rules. It is built as a plug-in system with some persistence back-ends already developed; i.e., raw files or database-dependant. In FELIX, the policy

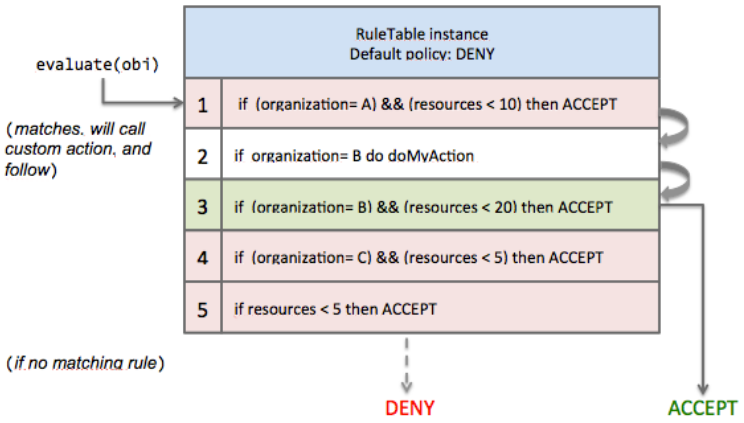


Figure 2.16: pyPElib rule evaluation

rules are identified at load time and evaluated every time a request is issued to (M)RO.

#### 2.7.2.3 Module testing

The speaks-for functionality has been checked by generating the required ABAC credentials and then validating them at both local and foreign RMs, when sent along with requests. Similarly, the integration with jFed was tested by using the certificate and key issued by C-BAS to log onto jFed and then reserving successfully resources at different RMs. As for the Admin Tool, its functionality was tested in a number of ways: among others, ranging from log-in using appropriate and incorrect certificate and key to create slices, projects and members; or changing member privileges and their membership on slices and projects and verify the changes took place.

## 2.8 Monitoring

### 2.8.1 Description of the module

The Monitoring System (MS) is the FELIX software module that collects the monitoring data of the resources available or provisioned in the FELIX infrastructure to provides it to both users and administrators.

As with RO and MRO, MS can also run in two operation modes: standard MS and Master MS (MMS). MS is responsible for the monitoring of each island and aggregate and forward it to the Master MS. As for MMS, it provides all monitoring data of FELIX infrastructure, receiving the aggregate data from the MS in each island.

### 2.8.2 Features

The following subsections summarise the features supported by the (M)MS. The features for the interaction with the upper layer and the configuration of the multiple master layer were enhanced during Y3

#### 2.8.2.1 Previously supported features

In deliverable D3.2 [2] the Slice Monitoring was described, as well as some features of the (M)MS. Below is a summary of the functionalities of the (M)MS reported at that time:

- Provide data on the physical and slice topologies
  - MS receives the physical and slice topology information from (M)RO and provide it to an external entity such as an experimenter or the monitoring GUI.

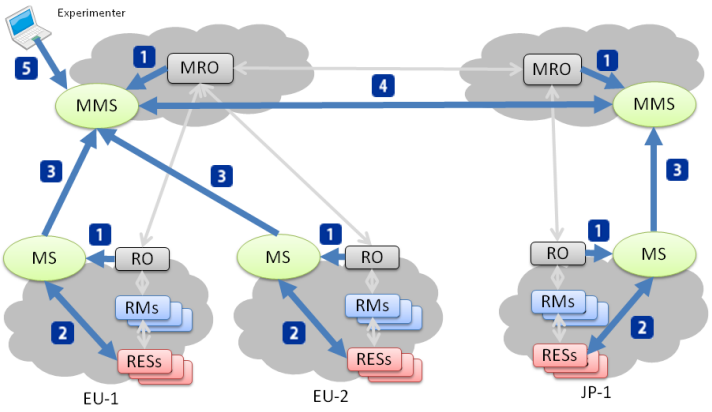


Figure 2.17: Flow of information in the Monitoring System

- Gather monitoring metrics from infrastructure
  - MS gathers monitoring metrics of physical resources by interacting with third-party monitoring tools (e.g., perfSONAR and Zabbix)
  - The infrastructure metrics being currently monitored are the following:
    * CRM: Life status, Load average
    * SDNRM: Life status, Port status, Traffic (tx), Traffic (rx)
    * SERM: Life status, Port status, Traffic (tx), Traffic (rx)
    * TNRM: Connection status
- Provide monitoring metrics
  - Provide monitoring metrics to the external entities, e.g. experimenter or monitoring GUI

#### 2.8.2.2 New features

During Y3 we focused mostly on the integration at the upper layer, i.e. between (M)RO and (M)MS. This leads to the following improvements:

- More complete topology information for the monitoring performed within (M)RO, such as matching information of the actual link and an *abstract link* defining and end-to-end inter-island link
- Interaction with multiple MROs in JP and EU
- Added more logging and test mode features for easier debugging

#### 2.8.2.3 Module testing

In order to validate the (M)MS functions, the module has been deployed in the FELIX test-bed and used to monitor several slices at a time, spanning both Japanese and European infrastructures.

Since (M)RO implements new features, the integration between the Monitoring System and the up-to-date (M)RO was tested. On the other hand, the multiple MMS scenario was validated by using the Monitoring System within the Infrastructure Use Case.

# 3 Deployment

The developments of new features and improvements on the software modules of the FELIX stack are prone to imply some changes in the installation or configuration procedures on the affected modules. In this section we document the differences on such procedures, as well as other relevant information related to the management of the quality assurance, the organisation and repositories containing the FELIX codebases and related automated tasks to manage processes for code analysis and merging. At the end we provide an overview and some analysis about the deployment of each software module in the testbed.

## 3.1 Installation and Configuration

This section specifies the changes performed in the installation or configuration procedures during Y3, since the writing of the last deliverables. If no "*diff*" is provided for a module, it means there are no changes in its configuration, installation or management steps. The complete guides for configuring and installing the FELIX modules can be directly found at the wiki of the Github "*stack*" repository [18].

### 3.1.1 CRM

#### 3.1.1.1 Installation

The *xmlsec1* dependency was found to be missing in the *ocf* stack, and was consequently added to the dependencies file.

#### 3.1.1.2 Configuration

The user-defined settings file (*mySettings.py*) has a new setting (*TIME_ZONE*) in order to provide logs in the timezone defined by the administrator. Its value is follows the *tz* [19] format, e.g. "*Asia/Tokyo*".

### 3.1.2 SDNRM

#### 3.1.2.1 Installation

Same as CRM, the *xmlsec1* dependency was missing in the *ocf* stack, and was added to the modules' dependencies.

#### 3.1.2.2 Configuration

The user-defined settings file for SDNRM (*localsettings.py*) also contain the *TIME_ZONE* directive in order to set up a given timezone. This is useful, for instance, to use the defined timezone in the logs and correctly look up and review events through the logs in the local infrastructure.

### 3.1.3 SERM

#### 3.1.3.1 Installation

During Y3 there were no additional requirements for the deployment of SERM.

#### 3.1.3.2 Configuration

Comparing D3.3 [3], SERM configuration were changed because of addition of Pox OpenFlow controller plugin into SERM:

- A new attribute appeared in **se-config.yaml**:

```
provision_plugin: ryu_rest_of    # other possible values: pox_xmlrpc_of
```

- A new config file **pox-config.yaml** for POX plugin was added in *modules/resource/manager/stitching-entity/conf/* directory:

```
host: 127.0.0.1
rest_port: 8081
```

where *host* and *port* defines where XMLRPC service of the POX controller is available.

### 3.1.4 TNRM

In order to support the new features developed since D3.3 [3], some changes in the installation and configuration were made during Y3.

#### 3.1.4.1 Installation

To maintain persistence of the slice information even if TN-RM is restarted, we implemented a MySQL database to hold this information. Therefore, the installation was modified so that TN-RM can access a MySQL DB with the following parameters:

```
mySQLhost = "localhost"
mySQLuser = "root"
mySQLpass = "felix"
mySQLchar = "utf8"
```

The installation of TN RM was extended as follows for the installation of MySQL:

```
$ sudo apt-get update
$ sudo apt-get mysql-server
  New Password for the MySQL "root" user:
mysql> mysql --user = root --password = felix
mysql> Bye
$ sudo apt-get install python-mysqldb
```

#### 3.1.4.2 Configuration

The advertisement RSpec was updated to reflect the information of the available VLANs, which is acquired from TN-RM.

For the GRE tunnels, the Open vSwitch must be set in advance with the following command:

```
#!/bin/bash
## BR: bridge name of open vSwitch
## SEDEV: an Ethernet device connect to the switch controlled by Stitching Entity
## DPID: datapath id of bridge
## CONTROL: openflow controller, RYU is controller
## OVSDB: database server of open vSwitch

BR = "ovs1799"
DPID = "0000000000000010"
SEDEV = "eth1"
CONTROL = "tcp:127.0.0.1:6633"
OVSDB = "ptcp:44444"
```

```
ovs-vsctl br-exists $ BR
if [$ == 0?]; then
    ovs-vsctl del-br $ BR
fi
ovs-vsctl add-br $ BR
ovs-vsctl add-port $ BR $ SEDEV
ovs-vsctl set bridge $ BR other-config: datapath-id = $ DPID
ovs-vsctl set-controller $ BR $ CONTROL
ovs-vsctl set-manager $ OVSDB
```

Additionally, the GRE endpoint must be specified in the "*config.xml*" file, as shown in the following example:

```
felix_domain_id: felix domain identifier
felix_stp_id: felix stp (end point) identifier
nsi_stp_id: NSIv2 stp identifier
vlan:. vlan id, ex "1000,2000-3000"
capacity: maximum transfer rate (bandwidth), Mbps

type: "ovsgre" just only
sedev: ethernet device name connect the switch controlled by stitching entity
address: local ip address connect from other gre host
dpid: dpid of open vSwitch
ovsdb: ip address of database server for local ovs
ryu: ip address of ryu rest service

if GRE end point,
<Interface felix_domain_id = "urn: publicid: IDN + fms: aist: tnrm + stp"
    felix_stp_id = "urn: ogf: network: aist.go.jp: 2015: gre: bi-ps"
    nsi_stp_id = "urn: ogf: network: aist.go.jp: 2015: gre: bi-ps"
    vlan = "1700-1799"
    capacity = "10000"

    type = "ovsgre"
    sedev = "eth1"
    address = "172.21.100.15"
    dpid = "0x10"
    ovsdb = "tcp: 127.0.0.1: 44444"
    ryu = "http://172.21.100.15:8080"
/>
```

#### 3.1.4.3 Operation

To enable the creation of the GRE tunnels, a Ryu controller offering a REST interface were added. Tyu is started with the following command:

```
$ ryu-manager $RYUDIR/ryu/app/ofctl_rest.py $TNRM/gre-tnrm-rest.py
```

### 3.1.5 (M)RO

Since D3.1 [1], there were some changes in the installation and configuration for (M)RO, as it has been extended and adapted to modifications during Y3.

#### 3.1.5.1 Installation

New dependencies have been added to provide a coloured logging system (*colorlog*) and some restriction on the version used for the (M)RO WSGI server (*Werkzeug==0.9.4*).

#### 3.1.5.2 Configuration

- New configuration files:
  - **auth.conf**: to determine path to trusted certificates (verification of user credentials) and configure location of Clearinghouse
  - **crm.json**, **sdnrm.json**, **serm.json**: sample files to provide information on the devices, which will be passed to (M)MS
  - **policies.conf**: to set up specific policies on control access
- Modified files:
  - **flask.conf**: to configure address where (M)RO listens
    - New field to control automatic reload after source changes
  - **ro.conf**: for (M)RO-related params, such as binding address, frequency on resource refresh and such
    - New fields to define whether (M)RO acts as master or child (to allow coexistence of RO and MRO in the same location), whether SE resources are advertised to the user, etc

### 3.1.6 AAA

Since D3.2 [2], the C-BAS module requires few minor changes in both configuration and operation procedures.

#### 3.1.6.1 Configuration

Optionally, a list of trusted peer islands can be included in "*registry.json*" file found under "*<path_to_cbas>/deploy/*". If provided, C-BAS will periodically pull root certs from these peers.

#### 3.1.6.2 Operation

C-BAS can now be run or operated using a shell script with the *start/stop/status* arguments. The shell script is available under the C-BAS directory (e.g. at "*<path_to_cbas>/deploy/*").

Moreover, C-BAS Admin Tool can be run using following commands:

```
cd <path_to_cbas>
java -jar javaadmintool/bin/admintool.jar
```

### 3.1.7 Monitoring

Since D3.2 [2], the Monitoring System saw small changes in both its installation and configuration procedures.

#### 3.1.7.1 Installation

To install (M)MS more easily, a script was added to generate the required MySQL databases and tables, according to whether the monitoring system is acting as master or child; and allowing the coexistence of MS and MMS on a same location.

#### 3.1.7.2 Configuration

The configuration files were added more descriptive comments on each directive, and the default port to run the service was changed to comply to the agreed port distribution by FELIX software module.

## 3.2 Quality Assurance

To measure the quality of its code base, FELIX has set up two automated tasks via Jenkins CI [20]. Through them, it is possible to measure metrics such as number of lines/directories/files/classes/functions, number and groups of issues according to their severity, estimated number and

### 3.2.1 Detecting issues

The first task leverages on the SonarQube [21] functionality to prepare periodic analysis reporting facts of interest for development teams, such as the size of each software module with different granularity (LOCs, no. of files, directories, functions, etc), the current complexity of the software and how commented it is (both related with maintainability), and the number of issues according to its severity (typically critical, major or minor). Per each measure taken, SonarQube also expresses whether there is a positive or negative trend taking place.

Some examples of the data provided by SonarQube is provided in the next figures. Figure 3.1 and Figure 3.2 feature its dashboard, showing four different sections:

- The upper-left section provides a graphic where each module is picture by a circumference. The no. of comments is defined in the X axis, the complexity is measured in the Y-axis and the no. of issues is represented by the radius of each circle.
- The upper-right section lists each module, along with its lines of code, the estimated time required to solve the issues (technical debt) and the trend followed.
- The lower-left section represents the number of comments (defined by the size of the block) and the comments/code ratio (defined by its colour, ranging from green to red).
- The lower-right section shows the percentage of duplicated lines per module through its size (the bigger, the more code is repeated) and colour (the greener, the better).

With all that, an ideal software module, when compared to others, would be represented as much as possible to the right, as next to the base of the X axis and as smaller as possible. It would also have minimal issues and reduced technical debt.

During the third year, the analysis of each modules was refined to only account for the code developed under FELIX, and thus avoid legacy or third-party code embedded into the stack. Some new analysis tasks were added to check for some other modules (namely M/CBAS and CRM-KVM). Using the results from previous QA analysis (Figure 3.1 and Figure 3.2), it is possible to observe the progression in the code quality.

For instance, the LOCs per module has increased for every module, yet the issues (and the technical debt estimated to solve them) have diminished. The complexity, as expected, has increased proportionally to the LOCs. This is not an optimal trend, but efforts have been made to compartmentalise functionality in separate, shorter

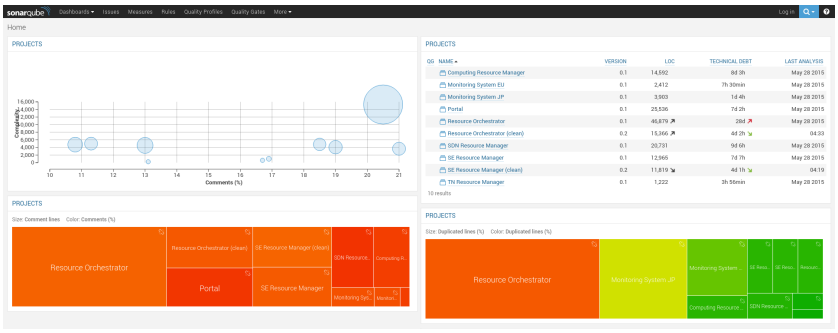


Figure 3.1: QA check per FELIX module (May 2015)

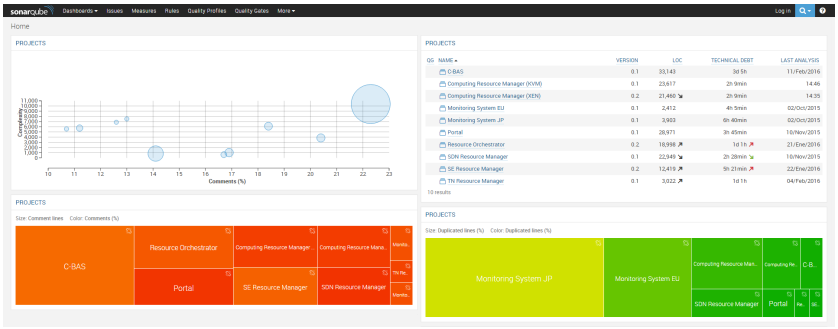


Figure 3.2: QA check per FELIX module (January 2016)

methods, classes and modules. On the other hand, there is an overall rising trend of the proportion of comments to code, and part of the duplicated code has been grouped and refactored so as to improve maintainability.

The metrics defined above are broken down per project (here, a FELIX module). Figure 3.3 and Figure 3.4 show the evolution of QA in the RO module. The data from the quality analyses is in line with the trends inferred; as size of code and complexity have increased, and issues and duplications have decreased.

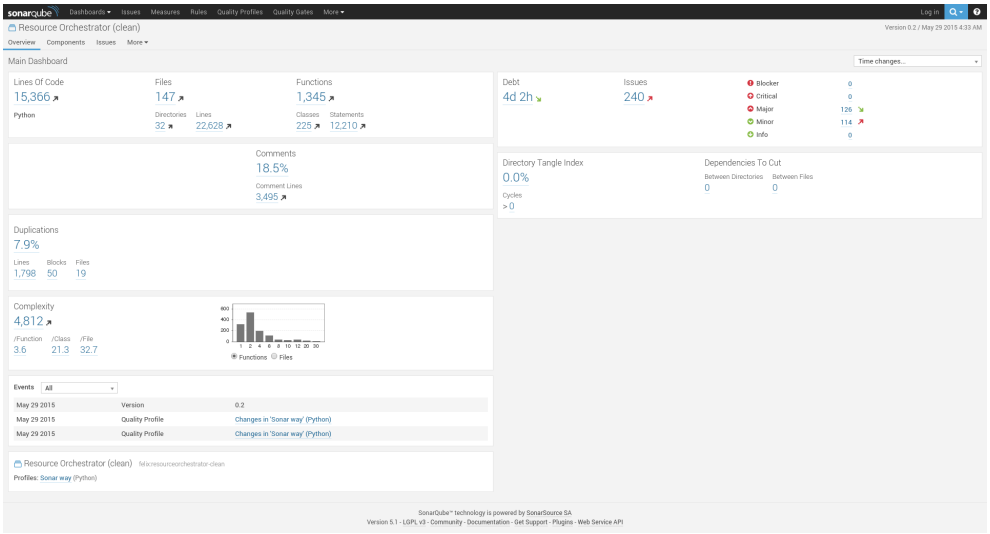


Figure 3.3: QA check for RO (May 2015)

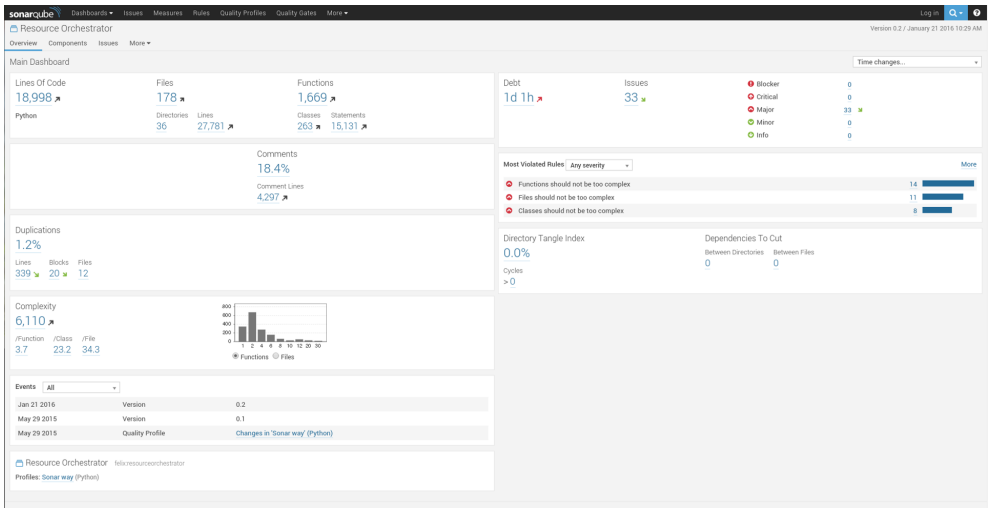


Figure 3.4: QA check for RO (January 2016)

The second automated task complements the quality analysis on the project's stack code. Through Jenkins, it retrieves the code of each FELIX module and runs the flake8 [22] tool to trigger checks via PEP8 and PyFlakes [23] analysis tools. The specific details are shown in the console log for each build in Jenkins. Typical warnings relate to length of lines, incorrect indentation, unused imports, variables and such. A third tool (PyLint) [24] is ran internally and its results are graphically displayed in a chart that shows the trend of non-compliance issues, as shown in Figure 3.5.

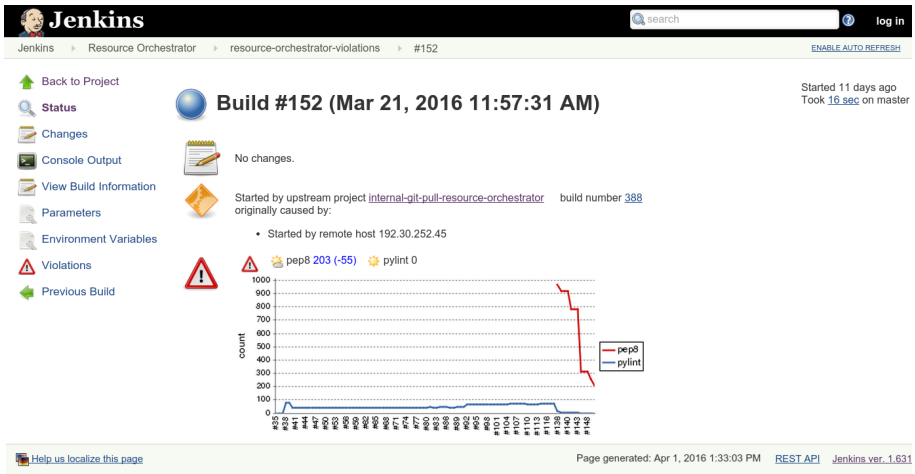


Figure 3.5: Automated task to count violations to coding standards

### 3.2.2 Placing everything together

Finally, we added a last automatic task to take over the job of integrating all code in the *master* branch. The aim of this task is to gather all code from the modules and place it together in the main branch, which is directly visible to external users. Such procedure is performed after every change is pushed to the repository and has been possible thanks to the use of hooks in GitHub and listening to them in Jenkins. Figure 3.6 shows the *git-merge* task, where Jenkins detects and fetches the changes in any given branch that has active hooks (here, the one corresponding to the Stitching Entity) and starts the build. This runs a shell script that, given new changes in

any branch, gathers all changes and put in a single commit (the *squash* operation). In case of conflict, the Git's strategy named "*theirs*" will be automatically applied in order to favour the new changes over the old ones.

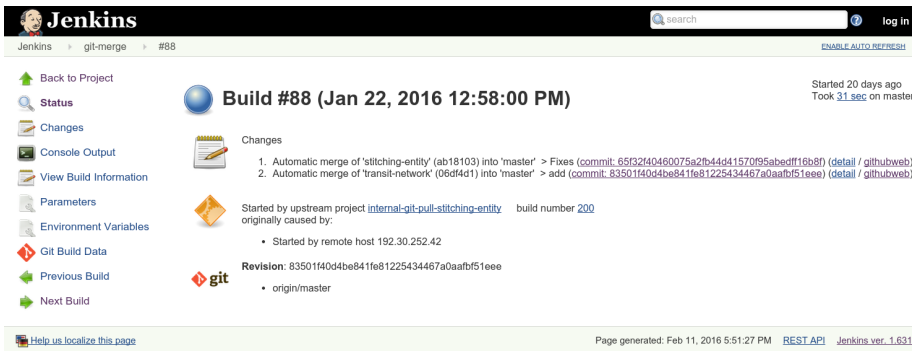


Figure 3.6: Automated task to merge new code into master

As it is an automated procedure, it frees developers from periodic merge processes and conflict solving by making sure every change is included in time. Although, on the other hand, this requires trusting every internal developer and the changes they perform on the code. This, of course, is not always the case; yet this procedure has proven less time costly than performing code reviews and manual integration for every change in the repository.

## 3.3 Code Repository

The codebase used in FELIX, whether it comprises its software stack or the sources used to develop the Use Cases, are made public under the "*ict-felix*" organisation in the GitHub site. During Y3 we have undergone a migration to this new organisation [25] and structure in an attempt to provide a centralised and clear set of repositories for all FELIX-related developments, which will be left to the public to access after the project lifetime.

### 3.3.1 FELIX stack

As this code base consists of multiple software modules, each with its own particularity and authors, the source code for this stack is released over a number of licenses, explicitly defined in the root of the project. Specifically, Apache 2.0 affects most modules (M/MS, M/RO, SERM, TNRM), as those were newly developed or integrated within the project's lifetime. For the modules that already existed or their direct extensions, either the OFELIA Copyright (CRM, Expedient, SDNRM) or the EICT Copyright (M/CBAS) applies. The sources from external libraries which are provided along with the code released in FELIX are accompanied and subject to their own license.

On the other hand, the source code is organised in different branches: one per software module (or *bundle*), where internal developers contribute; and the *master branch*, where code from all modules is merged.

A similar structure is provided for most branches, in an attempt to achieve two goals: i) use a standard, clear organisation of code where the functionality is separated in folders with a descriptive name; and ii) allow the merging procedure with the *master* branch to minimise conflicts by completely isolating contributions per module, i.e. in their respective folder. Figure 3.7 shows a view of structure of the merged *master* branch, along with the license of each module.

The GitHub wiki of the "*stack*" repository [18] provides external users, or possible external developers, with details on the organisation of the code for some of the FELIX modules (see *Contributing* > *Development info*). For those modules developed from scratch under FELIX we provide a clear structure with functionality separated per folder, so as to a) contain trusted CA certificates (folder "*cert*"), b) store configuration files that may be modified by the user (folder "*conf*") and c) any specific documentation (folder "*doc*"), d) provide helping scripts and binaries

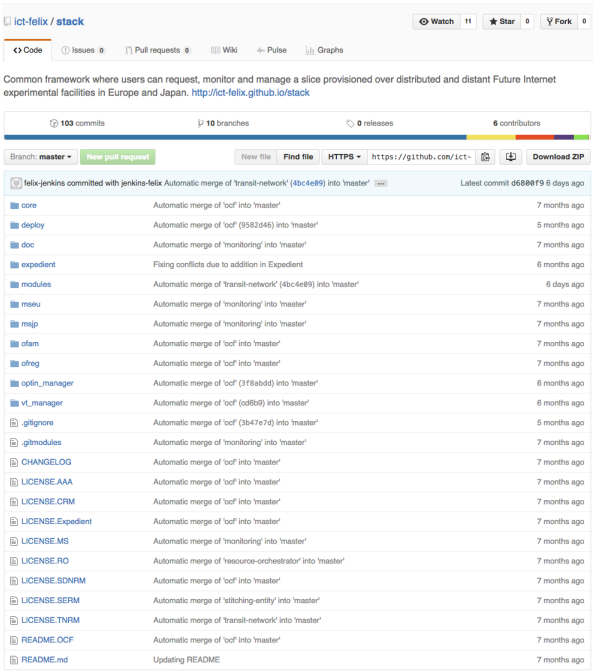


Figure 3.7: View of the ””stack”” repository (”master” branch)

(folder ”*deploy*”), e) keep the log of each module (folder ”*log*”), f) contain the base source (folder ”*src*”) or g) any test provided to rapidly verify proper working (folder ”*test*”).

### 3.3.2 FELIX Use Cases

The FELIX software stack is the main codebase and thus monopolises a great share of the development effort. During Y2 and part of Y3, only this set of sources were made available; but after the migration we decided to provide as well the development works for the tools that are used to run the Use Cases in top of the FELIX test-bed.

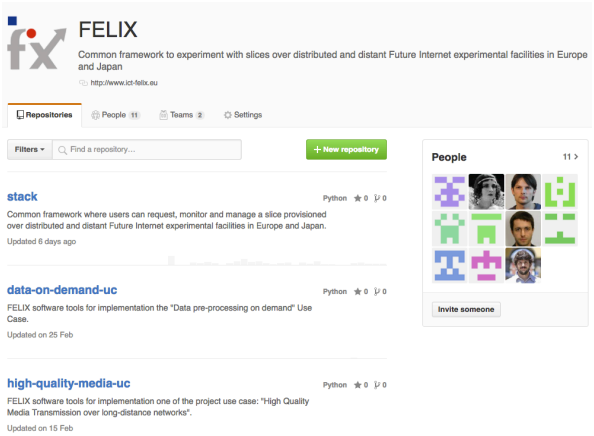


Figure 3.8: View of the ”ict-felix” organisation and repositories

Figure 3.8 shows the currently available repositories under the ”*ict-felix*” organisation, under its GitHub account. So far, the ”*Data pre-processing on demand*” and the ”*High-quality media transmission*” Use Cases share their developed tools publicly.

## 3.4 Module Deployment

Before moving to the conclusions on the development and deployment of the FMS components belonging to the FELIX test-bed, it is convenient to provide an overall view on the deployment of the FELIX Management Stack (FMS). At the time of writing this deliverable, the deployment of the FMS in the different islands is as depicted in Figure 3.9.

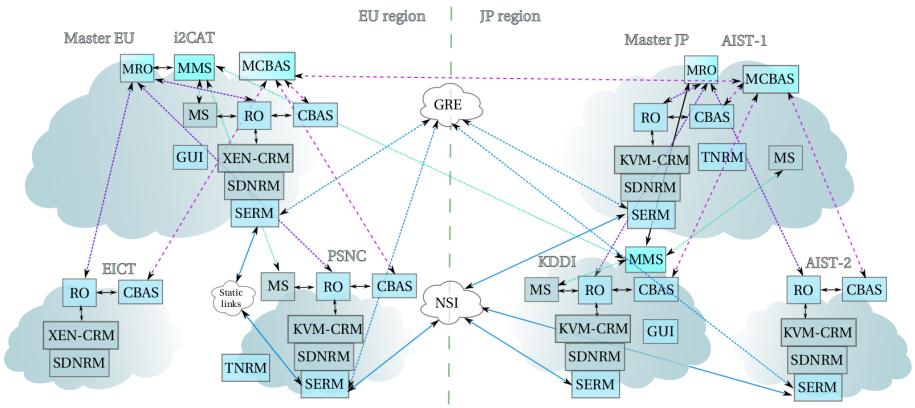


Figure 3.9: Deployment in FELIX islands

Six islands have installed part or the full FMS; and are currently operative (three per continent). In the European side, i2CAT, EICT and PSNC provide islands with elements from the FELIX stack. In the Japanese side, AIST and KDDI each provide their operative island to FELIX; plus a second island in AIST to fulfil the validation of the Infrastructure Migration use case.

All islands are interconnected by an internal VPN (connections not shown in the figure), and some islands also offer public access. This is specially important for those islands containing Master RO, for instance. On the other hand, to provide interconnectivity for users, islands are interconnected by other means, such as static links, NSI domain or GRE tunnels. This can be also appreciated in Figure 3.9.

The deployment in FELIX project shows an heterogeneous environment, both in software deployment and also in setting up the network. This is explained by some circumstances:

- Usage of XEN-CRM and KVM-CRM relies on the underlying infrastructure
  - European islands tend to use XEN hypervisors, whilst Japanese islands make use of KVM infrastructure. The exception is the PSNC island, where XEN was replaced by KVM during Y3
- User interaction with the FELIX infrastructure is externalised to a separate user tool (jFed)
  - Whilst GUI has been deployed along with MS during Y2, that is used to visualise monitoring data. Management of resources is possible through the jFed tool
- NSI domain requires centralised management, thus TNRM is not deployed in all islands
  - During Y2 and part of Y3, a centralised instance of TNRM was running at AIST island. Throughout Y3, a new instance was added in the European region
- GRE-TNRM can set up links between domains that have been properly configured and have running instances of Ryu and OVS

Notwithstanding the exceptions mentioned above, the status on the final deployment of the FMS corresponds to the expected outcome.

# 4 Conclusions and Summary

The development work during Y3 was oriented to cover four main objectives: i) conclusion of the implementation on some key missing development features, ii) realisation of the recommendations during Y2 review, iii) continue with the ongoing integration process and adapt it to the new features, and iv) satisfying the needs of the Use Cases, through provided feedback and requested features.

Throughout this period, the main key features implemented have been aimed to support the generation of different hypervisors and type of VMs in the FELIX infrastructure (KVM-CRM), to include the means for the experimenter to request GRE tunnels over the Internet to interconnect domains (GRE-TNRM), and to support minimal requests from the user, hiding the details of the inter-domain connectivity at both the level of stitching and transit network (MRO – virtual links). Several effort has been dedicated as well to attain more correct and detailed outputs from the APIs, to improve the error conditions management, the persistence and updating of the resources in the databases; as well as the integration and testing of new features, correcting any bug found during the year or dedicate some time to improve the code.

Thereby, the result of this period is an improved FELIX stack, with support for more than one flavour or kind of resources in some modules, and with the capability of understanding requests of different complexity and granularity. It is worth noting that, in order to achieve the missing key features and following recommendations from Y2, the impact and priority of some development works in the project was reassessed during Y3 in favour of the improved features and robustness of key software modules that can be reused in future research efforts.

Having said this, we identified some strengths of the FELIX software stack that should be acknowledged for future works which are subject to similar or related requirements. We consider that some of the RMs developed in FELIX could be used anywhere in the FIRE or GENI communities. Such is the case of **TN-RM**, which addresses the need for NSI-based dynamic paths over Research and Education (R&E) Networks – and a more typical need for GRE tunnels; but also of **(M)RO**, which incorporates a mapping mechanism that analyses the user's request to find out about the internal inter-domain endpoints and set up as needed. In between those modules stands **SE-RM**, a set of software module and controllers able to intertwine or stitch traffic flowing between one domain (SDN, manageable to user) and another (TN, being managed by transport R&E Networks). The usage of this module is also crucial to decouple the SDN domain from technologies and related frame tagging used by the inter-domain transport networks. The three software modules can be used in a stand-alone mode or in conjunction with others. Instructions for their installation, requirements and configurations are defined either in the "*Installing*" or "*Configuration*" subsections under "*Administering*", in the FELIX Github wiki [18]; or directly documented in the module.

Notwithstanding the above reference to some of the FELIX modules, others such as **C-BAS** or **C-RM** are likely to be used in future initiatives with overlapping needs. For instance, the former has been disseminated in communities such as GENI and Fed4FIRE. Other modules or components, such as **SDN-RM** or the perfSONAR tool used to gather metrics by **(M)MS** are already in place in other initiatives or infrastructures.

The deployment of any missing module from the FELIX stack was a work performed in parallel to the development of new and extended features, to the assessment of the proper integration with related modules (e.g. (M)RO with (M)MS) after those software changes; but also to the development and deployment of the Use Cases. In this latter aspect, the more complete the deployment is, the more the functionalities can be provided and validated. We have analysed the status of the deployment within the FELIX test-bed and provided an up-to-date schema reflecting it.

Besides the development, deployment and testing, we dedicated some efforts to improve the structure of the modules in the FELIX stack and the organisation of the codebase as a whole, as well as the structure of the public repositories that will be left available for the future. Therefore, in Y3 we undergone a migration of the codebase to a dedicated organisation in Github, so we can provide the software stack and the sources for the FELIX Use Cases in different, clearly identified repositories. The structure of the "*stack*" repository was revised in order to adapt to automatic procedures for merging code in the "*master*" branch; as this alleviates manual work

that otherwise would be required quite often.

We consider that the FELIX management stack is now a more robust codebase that provides key features to achieve inter-domain connectivity by different means (either through high-speed R&E networks or common GRE tunnels) and to add a a further degree of simplicity on the interaction with the user, allowing minimal information to be requested and completed internally with appropriate links, ports and VLAN tagging. Consequently, the stack, either as a whole or on a module basis, is ready and prone to be used in similar initiatives in the future.

# References


[1] R. Monno, et al., “FELIX Deliverable D3.1, Resource Planning and Provisioning,” FELIX Deliverable D3.1, January 2015.

[2] T. Ikeda, et al., “FELIX Deliverable D3.2, Slice Monitoring,” FELIX Deliverable D3.2, January 2015.

[3] B. Belter, et al., “FELIX Deliverable D3.3, Inter-Domain Networking Between SDN Slices,” FELIX Deliverable D3.3, Janaury 2015.

[4] C. Bermudo, et al., “FELIX Deliverable D3.4, End User Tools and API,” FELIX Deliverable D3.4, January 2015.

[5] “GENI Aggregate Manager API v3.” http://groups.geni.net/geni/wiki/GAPI_AM_API_V3.

[6] “The jFed Experimenter client.” http://jfed.iminds.be/, February 2016.

[7] “OFELIA stack.” https://github.com/fp7-ofelia/ocf.

[8] “Ryu OpenFlow controller.” https://github.com/osrg/ryu.

[9] “OpenFlow Virtual Switch.” https://github.com/openvswitch/ovs.

[10] J. Naous, et al., “Expedient: A centralized pluggable clearinghouse to manage geni experiments,” January 2010.

[11] “ProtoGeni ClearingHouse.” http://www.protogeni.net/wiki/ClearingHouseDesc.

[12] “Slice Facility Architecture 2.0.” http://groups.geni.net/geni/attachment/wiki/SliceFedArch/SFA2.0.pdf, July 2010.

[13] “GENI Clearinghouse.” http://groups.geni.net/geni/wiki/GeniClearinghouse.

[14] “Common Federation API.” http://groups.geni.net/geni/wiki/ CommonFederationAPIv2, November 2013.

[15] “ABAC credentials in GENI.” http://groups.geni.net/geni/wiki/TIEDABACCredential.

[16] “iMinds Authority.” https://authority.ilabt.iminds.be, February 2016.

[17] “FELIX User Registration Form.” http://goo.gl/forms/jikTkmwdGu.

[18] “FELIX wiki (GitHub).” https://github.com/ict-felix/stack/wiki, February 2016.

[19] “Time Zone Database.” https://www.iana.org/time-zones, March 2016.

[20] “Jenkins CI.” https://jenkins-ci.org, February 2016.

[21] “SonarQube.” http://www.sonarqube.org, February 2016.

[22] “Flake8 tool.” https://flake8.readthedocs.org, February 2016.

[23] “PyFlakes checker.” https://pypi.python.org/pypi/pyflakes, February 2016.

[24] “PyLint checker.” https://www.pylint.org, February 2016.

[25] “FELIX organisation (GitHub).” https://github.com/ict-felix, March 2016.

# Acronyms

**API** Application Programming Interface

**CBAS** Certificate-based AAA for SDN Experimental Facilities

**CRM** Compute and storage Resource Manager

**DB** Database

**FLS** First-Level Support

**FMS** FELIX Management Stack

**GENI** Global Environment for Network Innovations

**GRE** Generic Routing Encapsulation

**GRE-TNRM** Generic Routing Encapsulation – Transit Network Resource Manager

**GUI** Graphical User Interface

**JFED** Java-based framework for testbed federation

**JNLP** Java Network Launching Protocol

**KVM** Kernel-based Virtual Machine

**LOC** Lines of Code

**MCBAS** Master Certificate-based AAA for SDN Experimental Facilities

**MMS** Master Monitoring System

**MS** Monitoring System

**MRO** Master Resource Orchestrator

**NSI** Network Service Interface

**NSI-TNRM** Network Service Interface – Transit Network Resource Manager

**PE** Policy Engine

**R&D** Research & Development

**RM** Resource Manager

**RO** Resource Orchestrator

**SDN** Software-Defined Networking

**SEF** SDN Experimental Facility

**SDNRM** Software-Defined Networking Resource Manager

**SERM** Stitching Entity Resource Manager

**STP** Service Termination Point

**TN** Transit Network

**TNRM** Transit Network Resource Manager

**VLAN** Virtual Local Area Network

**UC** Use Case

**VM** Virtual Machine

**Y**∗ Year ∗

# Appendix I

## A MRO (virtual links)

To complement the explanation given in the (M)RO section we provide here an example on the level of detail of a user request that asks for SDN resources in three FELIX islands and asks for two virtual links to generate two GRE tunnel connecting two islands at a time.

First we show the user request, and later we append the final request, as extended internally with the information retrieved from its internal database and consisting on SE information (nodes and links) and TN endpoints (nodes), previously fetched from their respective modules.

### A.1 User request

First, the experimenter request SDN resources in three islands and asks to interconnect them through GRE tunnels from an island to the other two.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rspec  type="request"
        xs:schemaLocation="http://www.geni.net/resources/rspec/3
                           http://hpn.east.isi.edu/rspec/ext/stitch/0.1/
                           http://hpn.east.isi.edu/rspec/ext/stitch/0.1/stitch-schema.xsd
                           http://www.geni.net/resources/rspec/3/request.xsd"
        xmlns="http://www.geni.net/resources/rspec/3"
        xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:openflow="http://www.geni.net/resources/rspec/ext/openflow/3"
        xmlns:felix="http://ict-felix.eu/serm_request"
        xmlns:sharedvlan="http://www.geni.net/resources/rspec/ext/shared-vlan/1"
        xmlns:stitch="http://hpn.east.isi.edu/rspec/ext/stitch/0.1/"
        xmlns:emulab="http://www.protogeni.net/resources/rspec/ext/emulab/1">

  <node client_id="MonitVMi2CAT1"
        component_id="urn:publicid:IDN+ocf:i2cat:vtam+node+Verdaguer"
        component_manager_id="urn:publicid:IDN+ocf:i2cat:vtam+authority+cm"
        exclusive="true">
      <sliver_type name="emulab-xen">
        <emulab:xen cores="1" ram="512" disk="8"/>
     </sliver_type>
  </node>
  <node client_id="MonitVMi2CAT2"
        component_id="urn:publicid:IDN+ocf:i2cat:vtam+node+Rodoreda"
        component_manager_id="urn:publicid:IDN+ocf:i2cat:vtam+authority+cm"
        exclusive="true">
      <sliver_type name="emulab-xen">
        <emulab:xen cores="1" ram="512" disk="8"/>
     </sliver_type>
  </node>

  <node client_id="MonitVMAIST1"
        component_id="urn:publicid:IDN+ocf:aist:vtam+node+dc1-3"
        component_manager_id="urn:publicid:IDN+ocf:aist:vtam+authority+cm"
```

```
    exclusive="true">
    <sliver_type name="emulab-xen">
      <emulab:xen cores="1" ram="512" disk="10"/>
      <disk_image name="/mnt/llvm/template/llvm.qcow2"/>
    </sliver_type>
  </node>

  <node client_id="MonitVMPSNC1"
        component_id="urn:publicid:IDN+ocf:psnc:vtam+node+psnc-ibm2"
        component_manager_id="urn:publicid:IDN+ocf:psnc:vtam+authority+cm"
        exclusive="true">
    <sliver_type name="emulab-xen">
      <location xmlns="http://jfed.iminds.be/rspec/ext/jfed/1" x="202.0" y="209.5"/>
      <emulab:xen cores="1" ram="512" disk="10"/>
      <disk_image name="/mnt/llvm/template/llvm.qcow2"/>
    </sliver_type>
  </node>
  <node client_id="MonitVMPSNC2"
        component_id="urn:publicid:IDN+ocf:psnc:vtam+node+psnc-blade-3"
        component_manager_id="urn:publicid:IDN+ocf:psnc:vtam+authority+cm"
        exclusive="true">
    <sliver_type name="emulab-xen">
      <location xmlns="http://jfed.iminds.be/rspec/ext/jfed/1" x="202.0" y="209.5"/>
      <emulab:xen cores="1" ram="512" disk="10"/>
      <disk_image name="/mnt/llvm/template/llvm.qcow2"/>
    </sliver_type>
  </node>

  <openflow:sliver email="felix.user@some.mail.net" description="OF-request for i2CAT island">
    <openflow:controller url="tcp:10.21.32.43:6633" type="primary"/>
    <openflow:group name="i2CAT">
      <openflow:datapath component_id="urn:publicid:IDN+openflow:ocf:i2cat:ofam+datapath+00:10:00:00:00:00:00:01"
                         component_manager_id="urn:publicid:IDN+openflow:ocf:i2cat:ofam+authority+cm"
                         dpid="00:10:00:00:00:00:00:01">
        <openflow:port name="GBE0/5" num="5"/>
        <openflow:port name="GBE0/6" num="6"/>
        <openflow:port name="GBE0/12" num="12"/>
      </openflow:datapath>
      <openflow:datapath component_id="urn:publicid:IDN+openflow:ocf:i2cat:ofam+datapath+00:10:00:00:00:00:00:03"
                         component_manager_id="urn:publicid:IDN+openflow:ocf:i2cat:ofam+authority+cm"
                         dpid="00:10:00:00:00:00:00:03">
        <openflow:port name="GBE0/1" num="1"/>
        <openflow:port name="GBE0/6" num="6"/>
        <openflow:port name="GBE0/12" num="12"/>
      </openflow:datapath>
    </openflow:group>
    <openflow:match>
      <openflow:use-group name="i2CAT"/>
      <openflow:packet>
        <openflow:dl_vlan value="2878" />
      </openflow:packet>
```

```
      </openflow:match>
  </openflow:sliver>

<openflow:sliver email="felix.user@some.mail.net" description="OF-request for AIST island">
  <openflow:controller url="tcp:10.21.32.43:6633" type="primary"/>
  <openflow:group name="AIST">
    <openflow:datapath component_id="urn:publicid:IDN+openflow:ocf:aist:ofam+datapath+00:00:00:00:00:00:00:01"
                       component_manager_id="urn:publicid:IDN+openflow:ocf:aist:ofam+authority+cm"
                       dpid="00:00:00:00:00:00:00:01">
      <openflow:port num="3" name="eth14"/>
      <openflow:port num="5" name="eth3"/>
    </openflow:datapath>
  </openflow:group>
  <openflow:match>
    <openflow:use-group name="AIST"/>
    <openflow:packet>
      <openflow:dl_vlan value="1680" />
    </openflow:packet>
  </openflow:match>
</openflow:sliver>

<openflow:sliver email="felix.user@some.mail.net" description="OF-request for PSNC island">
  <openflow:controller url="tcp:10.21.32.43:6633" type="primary"/>
  <openflow:group name="PSNC">
    <openflow:datapath component_id="urn:publicid:IDN+openflow:ocf:psnc:ofam+datapath+00:00:08:81:f4:88:f5:b0"
                       component_manager_id="urn:publicid:IDN+openflow:ocf:psnc:ofam+authority+cm"
                       dpid="00:00:08:81:f4:88:f5:b0">
      <openflow:port name="ge-1/1/4.0" num="14"/>
      <openflow:port name="ge-1/1/6.0" num="16"/>
      <openflow:port name="ge-1/1/7.0" num="17"/>
    </openflow:datapath>
  </openflow:group>
  <openflow:match>
    <openflow:use-group name="PSNC"/>
    <openflow:packet>
      <openflow:dl_vlan value="3156" />
    </openflow:packet>
  </openflow:match>
</openflow:sliver>

<link client_id="urn:publicid:IDN+fms:i2cat:mapper+link+i2cat_psnc">
  <component_manager name="urn:publicid:IDN+fms:i2cat:mapper+authority+cm"/>
  <link_type name="urn:felix+virtual_link+type+gre"/>
  <interface_ref client_id="urn:publicid:IDN+fms:i2cat:mapper+domain+i2cat"/>
  <interface_ref client_id="urn:publicid:IDN+fms:i2cat:mapper+domain+psnc"/>
</link>

<link client_id="urn:publicid:IDN+fms:i2cat:mapper+link+i2cat_aist">
  <component_manager name="urn:publicid:IDN+fms:i2cat:mapper+authority+cm"/>
  <link_type name="urn:felix+virtual_link+type+gre"/>
  <interface_ref client_id="urn:publicid:IDN+fms:i2cat:mapper+domain+i2cat"/>
```

```
    <interface_ref client_id="urn:publicid:IDN+fms:i2cat:mapper+domain+aist"/>
  </link>

</rspec>
```

## A.2 Generated request

Inside MRO, the SDN flowspaces are analysed in conjunction with the virtual link. The database of MRO is queried first for the list of available transit domain endpoints (*STPs*) and the free VLANs. After MRO identifies and ensures a valid path, it then asks about the list of available ports and VLANs on the stitching device. With this and the information obtained from the mapper component inside (M)RO, it determines the list of SE ports connected to the TN endpoints and sets a path, which is converted to a request. This request operates under the same workflow as any other, being directed to the proper FELIX modules for reservation (*allocation*) and activation (*provisioning*).

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rspec  type="request"
        xs:schemaLocation="http://www.geni.net/resources/rspec/3
                           http://hpn.east.isi.edu/rspec/ext/stitch/0.1/
                           http://hpn.east.isi.edu/rspec/ext/stitch/0.1/stitch-schema.xsd
                           http://www.geni.net/resources/rspec/3/request.xsd"
        xmlns="http://www.geni.net/resources/rspec/3"
        xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:openflow="http://www.geni.net/resources/rspec/ext/openflow/3"
        xmlns:felix="http://ict-felix.eu/serm_request"
        xmlns:sharedvlan="http://www.geni.net/resources/rspec/ext/shared-vlan/1"
        xmlns:stitch="http://hpn.east.isi.edu/rspec/ext/stitch/0.1/"
        xmlns:emulab="http://www.protogeni.net/resources/rspec/ext/emulab/1">

  <node client_id="MonitVMi2CAT1" component_id="urn:publicid:IDN+ocf:i2cat:vtam+node+Verdaguer"
  component_manager_id="urn:publicid:IDN+ocf:i2cat:vtam+authority+cm" exclusive="true">
    <sliver_type name="emulab-xen">
      <emulab:xen cores="1" ram="512" disk="8"/>
    </sliver_type>
  </node>
  <node client_id="MonitVMi2CAT2" component_id="urn:publicid:IDN+ocf:i2cat:vtam+node+Rodoreda"
  component_manager_id="urn:publicid:IDN+ocf:i2cat:vtam+authority+cm" exclusive="true">
    <sliver_type name="emulab-xen">
      <emulab:xen cores="1" ram="512" disk="8"/>
    </sliver_type>
  </node>

  <node client_id="MonitVMAIST1" component_id="urn:publicid:IDN+ocf:aist:vtam+node+dc1-3"
  component_manager_id="urn:publicid:IDN+ocf:aist:vtam+authority+cm" exclusive="true">
    <sliver_type name="emulab-xen">
      <emulab:xen cores="1" ram="512" disk="10"/>
      <disk_image name="/mnt/l1vm/template/l1vm.qcow2"/>
    </sliver_type>
  </node>
```

```
<node client_id="MonitVMPSNC1" component_id="urn:publicid:IDN+ocf:psnc:vtam+node+psnc-ibm2"
component_manager_id="urn:publicid:IDN+ocf:psnc:vtam+authority+cm" exclusive="true">
  <sliver_type name="emulab-xen">
    <location xmlns="http://jfed.iminds.be/rspec/ext/jfed/1" x="202.0" y="209.5"/>
    <emulab:xen cores="1" ram="512" disk="10"/>
    <disk_image name="/mnt/l1vm/template/l1vm.qcow2"/>
  </sliver_type>
</node>
<node client_id="MonitVMPSNC2" component_id="urn:publicid:IDN+ocf:psnc:vtam+node+psnc-blade-3"
component_manager_id="urn:publicid:IDN+ocf:psnc:vtam+authority+cm" exclusive="true">
  <sliver_type name="emulab-xen">
    <location xmlns="http://jfed.iminds.be/rspec/ext/jfed/1" x="202.0" y="209.5"/>
    <emulab:xen cores="1" ram="512" disk="10"/>
    <disk_image name="/mnt/l1vm/template/l1vm.qcow2"/>
  </sliver_type>
</node>

<openflow:sliver email="felix.user@some.mail.net" description="OF-request for i2CAT island">
  <openflow:controller url="tcp:10.21.32.43:6633" type="primary"/>
  <openflow:group name="i2CAT">
    <openflow:datapath component_id="urn:publicid:IDN+openflow:ocf:i2cat:ofam+datapath+00:10:00:00:00:00:00:01"
        component_manager_id="urn:publicid:IDN+openflow:ocf:i2cat:ofam+authority+cm" dpid="00:10:00:00:00:00:00:01">
      <openflow:port name="GBE0/5" num="5"/>
      <openflow:port name="GBE0/6" num="6"/>
      <openflow:port name="GBE0/12" num="12"/>
    </openflow:datapath>
    <openflow:datapath component_id="urn:publicid:IDN+openflow:ocf:i2cat:ofam+datapath+00:10:00:00:00:00:00:03"
        component_manager_id="urn:publicid:IDN+openflow:ocf:i2cat:ofam+authority+cm" dpid="00:10:00:00:00:00:00:03">
      <openflow:port name="GBE0/1" num="1"/>
      <openflow:port name="GBE0/6" num="6"/>
      <openflow:port name="GBE0/12" num="12"/>
    </openflow:datapath>
  </openflow:group>
  <openflow:match>
    <openflow:use-group name="i2CAT"/>
    <openflow:packet>
      <openflow:dl_vlan value="2878"/>
    </openflow:packet>
  </openflow:match>
</openflow:sliver>

<openflow:sliver email="felix.user@some.mail.net" description="OF-request for AIST island">
  <openflow:controller url="tcp:10.21.32.43:6633" type="primary"/>
  <openflow:group name="AIST">
    <openflow:datapath component_id="urn:publicid:IDN+openflow:ocf:aist:ofam+datapath+00:00:00:00:00:00:00:01"
        component_manager_id="urn:publicid:IDN+openflow:ocf:aist:ofam+authority+cm" dpid="00:00:00:00:00:00:00:01">
      <openflow:port num="3" name="eth14"/>
      <openflow:port num="5" name="eth3"/>
    </openflow:datapath>
  </openflow:group>
  <openflow:match>
```

```
    <openflow:use-group name="AIST"/>
    <openflow:packet>
      <openflow:dl_vlan value="1680"/>
    </openflow:packet>
  </openflow:match>
</openflow:sliver>

<openflow:sliver email="felix.user@some.mail.net" description="OF-request for PSNC island">
  <openflow:controller url="tcp:10.21.32.43:6633" type="primary"/>
  <openflow:group name="PSNC">
    <openflow:datapath component_id="urn:publicid:IDN+openflow:ocf:psnc:ofam+datapath+00:00:08:81:f4:88:f5:b0"
        component_manager_id="urn:publicid:IDN+openflow:ocf:psnc:ofam+authority+cm" dpid="00:00:08:81:f4:88:f5:b0">
      <openflow:port name="ge-1/1/4.0" num="14"/>
      <openflow:port name="ge-1/1/6.0" num="16"/>
      <openflow:port name="ge-1/1/7.0" num="17"/>
    </openflow:datapath>
  </openflow:group>
  <openflow:match>
    <openflow:use-group name="PSNC"/>
    <openflow:packet>
      <openflow:dl_vlan value="3156"/>
    </openflow:packet>
  </openflow:match>
</openflow:sliver>

<node client_id="urn:publicid:IDN+fms:i2cat:serm+datapath+10:00:78:ac:c0:15:19:c0"
component_manager_id="urn:publicid:IDN+fms:i2cat:serm+authority+cm">
  <interface client_id="urn:publicid:IDN+fms:i2cat:serm+datapath+10:00:78:ac:c0:15:19:c0_1"/>
  <interface client_id="urn:publicid:IDN+fms:i2cat:serm+datapath+10:00:78:ac:c0:15:19:c0_3"/>
  <interface client_id="urn:publicid:IDN+fms:i2cat:serm+datapath+10:00:78:ac:c0:15:19:c0_10"/>
  <interface client_id="urn:publicid:IDN+fms:i2cat:serm+datapath+10:00:78:ac:c0:15:19:c0_10"/>
</node>
<link client_id="urn:publicid:IDN+fms:i2cat:serm+datapath+10:00:78:ac:c0:15:19:c0_1?vlan=2878-10:00:78:ac:c0:15:19:c0_10?vlan=1735">
  <component_manager name="urn:publicid:IDN+fms:i2cat:serm+authority+cm"/>
  <link_type name="urn:felix+static_link"/>
  <interface_ref client_id="urn:publicid:IDN+fms:i2cat:serm+datapath+10:00:78:ac:c0:15:19:c0_1"/>
  <interface_ref client_id="urn:publicid:IDN+fms:i2cat:serm+datapath+10:00:78:ac:c0:15:19:c0_10"/>
</link>
<link client_id="urn:publicid:IDN+fms:i2cat:serm+datapath+10:00:78:ac:c0:15:19:c0_3?vlan=2878-10:00:78:ac:c0:15:19:c0_10?vlan=1795">
  <component_manager name="urn:publicid:IDN+fms:i2cat:serm+authority+cm"/>
  <link_type name="urn:felix+static_link"/>
  <interface_ref client_id="urn:publicid:IDN+fms:i2cat:serm+datapath+10:00:78:ac:c0:15:19:c0_3"/>
  <interface_ref client_id="urn:publicid:IDN+fms:i2cat:serm+datapath+10:00:78:ac:c0:15:19:c0_10"/>
</link>

<node client_id="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0"
component_manager_id="urn:publicid:IDN+fms:psnc:serm+authority+cm">
  <interface client_id="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0_13"/>
  <interface client_id="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0_14"/>
</node>
<link client_id="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0_13?vlan=3156-00:00:54:e0:32:cc:a4:c0_14?vlan=650">
```

```
  <component_manager name="urn:publicid:IDN+fms:psnc:serm+authority+cm"/>
  <link_type name="urn:felix+vlan_trans"/>
  <interface_ref client_id="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0_13"/>
  <interface_ref client_id="urn:publicid:IDN+fms:psnc:serm+datapath+00:00:54:e0:32:cc:a4:c0_14"/>
</link>

<node client_id="urn:publicid:IDN+fms:aist:serm+datapath+00:00:00:00:00:00:00:10"
component_manager_id="urn:publicid:IDN+fms:aist:serm+authority+cm">
  <interface client_id="urn:publicid:IDN+fms:aist:serm+datapath+00:00:00:00:00:00:00:10_3"/>
  <interface client_id="urn:publicid:IDN+fms:aist:serm+datapath+00:00:00:00:00:00:00:10_1"/>
</node>
<link client_id="urn:publicid:IDN+fms:aist:serm+datapath+00:00:00:00:00:00:00:10_3?vlan=1680-00:00:00:00:00:00:00:10_1?vlan=1736">
  <component_manager name="urn:publicid:IDN+fms:aist:serm+authority+cm"/>
  <link_type name="urn:felix+vlan_trans"/>
  <interface_ref client_id="urn:publicid:IDN+fms:aist:serm+datapath+00:00:00:00:00:00:00:10_3"/>
  <interface_ref client_id="urn:publicid:IDN+fms:aist:serm+datapath+00:00:00:00:00:00:00:10_1"/>
</link>

<node client_id="urn:publicid:IDN+fms:aist:tnrm+stp" component_manager_id="urn:publicid:IDN+fms:aist:tnrm+authority+cm"
exclusive="false">
  <interface client_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:i2cat.net:2015:gre:felix">
    <sharedvlan:link_shared_vlan vlantag="1735" name="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:i2cat.net:2015:gre:felix+vlan"/>
  </interface>
  <interface client_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:pionier.net.pl:2015:gre:bi-ps">
    <sharedvlan:link_shared_vlan vlantag="650" name="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:pionier.net.pl:2015:gre:bi-ps+vlan"/>
  </interface>
</node>
<node client_id="urn:publicid:IDN+fms:aist:tnrm+stp" component_manager_id="urn:publicid:IDN+fms:aist:tnrm+authority+cm" exclusive="false">
  <interface client_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:i2cat.net:2015:gre:felix">
    <sharedvlan:link_shared_vlan vlantag="1795" name="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:i2cat.net:2015:gre:felix+vlan"/>
  </interface>
  <interface client_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:aist.go.jp:2015:gre:bi-se1">
    <sharedvlan:link_shared_vlan vlantag="1736" name="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:aist.go.jp:2015:gre:bi-se1+vlan"/>
  </interface>
</node>
<link client_id="urn:publicid:IDN+fms:aist:tnrm+link+urn:ogf:network:i2cat.net:2015:gre:felix?vlan=1735-urn:ogf:network:pionier.net.pl:2015:gre:bi-ps?vlan=650">
  <component_manager name="urn:publicid:IDN+fms:aist:tnrm+authority+cm"/>
  <interface_ref client_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:i2cat.net:2015:gre:felix"/>
  <interface_ref client_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:pionier.net.pl:2015:gre:bi-ps"/>
  <property source_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:i2cat.net:2015:gre:felix"
      dest_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:pionier.net.pl:2015:gre:bi-ps" capacity="100"/>
  <property source_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:pionier.net.pl:2015:gre:bi-ps"
      dest_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:i2cat.net:2015:gre:felix" capacity="100"/>
</link>
<link client_id="urn:publicid:IDN+fms:aist:tnrm+link+urn:ogf:network:i2cat.net:2015:gre:felix?vlan=1795-urn:ogf:network:aist.go.jp:2015:gre:bi-se1?vlan=1736">
  <component_manager name="urn:publicid:IDN+fms:aist:tnrm+authority+cm"/>
  <interface_ref client_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:i2cat.net:2015:gre:felix"/>
  <interface_ref client_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:aist.go.jp:2015:gre:bi-se1"/>
  <property source_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:i2cat.net:2015:gre:felix"
      dest_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:aist.go.jp:2015:gre:bi-se1" capacity="100"/>
  <property source_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:aist.go.jp:2015:gre:bi-se1"
```

```
    dest_id="urn:publicid:IDN+fms:aist:tnrm+stp+urn:ogf:network:i2cat.net:2015:gre:felix" capacity="100"/>
  </link>
</rspec>
```

The scientific/academic work is financed from financial resources for science in the years 2013 - 2016 granted for the realization of the international project co-financed by Polish Ministry of Science and Higher Education.