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 D3.3 Inter-Domain Networking Between SDN Slices

Version 1.0

Due date:	31/11/2014
Submission date:	30/01/2015
Deliverable leader:	PSNC
Author list:	Bartosz Belter (PSNC), Krzysztof Dombek (PSNC), Artur Juszczyk (PSNC), Kostas Pentikousis (EICT), Umar Toseef (EICT), Gino Carrozzo (NXW), Roberto Monno (NXW), Carlos Bermudo (i2CAT), Carolina Fernandez (i2CAT), Tomohiro Kudoh (AIST), Atsuko Takefusa (AIST), Jason Haga (AIST), Fumihiro Okazaki (AIST), Jin Tanaka (KDDI), Takatoshi Ikeda (KDDI), Bart Puype (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

List of Tables

Abstract

This document details the new-created software entities and tools used for provisioning the inter-domain networking between SDN slices in the FELIX Federated Framework. To enable communication between SDN networks two modules have been designed: Transit Network Resource Manager (TN-RM) and Stitching Entity Resource Manager (SE-RM). This document gives the clear overview of the TN-RM and SE-RM design and implementation. It explains modules' role in FELIX architecture as well as key functionality and inter-communication workflows for the inter-domain network provisioning.

Excecutive Summary

Deliverable D3.3 aims to explain, from a high-level perspective, the most important concepts of the implementation and design as well as deployment of the modules involved in the inter-domain networking connectivity. This document covers the implementation of Transit Network Resource Manager (TN-RM), defined in FELIX deliverable D2.2 [1], and Stitching Entity Resource Manager (SE-RM). These software components manage the dynamic inter-domain network connectivity and its stitching with the SDN resources. The inter-domain connectivity management bases on the three levels of FELIX management structure: Resource Orchestrators (RO), Resource Managers (RM) and resources themselves. This document is part of a set of deliverables describing the FELIX concept implementation. For better understanding how the SE-RM and TN-RM modules work in FELIX concept, we recommend to look at deliverable D3.1 presenting other Resource Managers and Orchestrator.

The information presented in this document is updated with the latest additions, it is base to future extension or modification derived from the integration tests and the FELIX Use Cases preparation scheduled for Y3. The document introduce firstly the concepts and definitions related to the TN-RM and SE-RM. Afterwards it explains the particularities of the design and the key internal functionality, as well as the workflows intended for each module and its communication with related components. Then lower-level details are presented in the deployment section, including guidelines for the configuration and deployment processes and the common operation usage. This document is addressed to software architects, software and network engineers, system administrators, software developers implementing specific features of the resource managers, especially in inter-domain network connectivity subject and SDN testbeds.

1 Introduction

The main concept of Resource Managers presented in this deliverable is the creation of mechanisms for network provisioning between SDN slices. Presented integrated service prototype supports the FELIX architecture with mechanisms to implement network connectivity in particular islands and between them. Two modules were prototyped: the Transit Network Resource Manager (TN-RM) and Stitching Entity Resource Manager (SE-RM). These modules are controlled by Resource Orchestrator (RO) which orchestrate different types of Resource Managers in the island. Both SE-RM and TN-RM communicate with its RO, in order to receive requests and to notify RO about resource status, success or failure events.

The Transit Network Resource Manager provides an inter-island connectivity using the Network Service Interface provisioning service. This service allows on reserving the point-to-point connectivity between the FELIX islands. Stitching Entity Resource Manager is a component that is in charge of interconnecting the external connection points with the local SDN island. In fact, this is ensured by the switching rules configuration (e.g. adding OpenFlow flows) on the switching device (Stitching Entity). The stitching concept has been introduced by the GENI (Global Environment for Network Innovations) initiative [2], the draft description can be found in the "GENI Network Stitching -- Overview" document [3].

Both modules are based on the concepts of GENI architecture and the OFELIA Aggregate Managers. To make them consistent with other FELIX modules the common part (Delegate, Parsers, Formatters) is based on the RO module implementation. Also, both offer similar northbound APIs as other FELIX manager modules, i.e. the GENIv3. The SE-RM allows an experimenter to request, update and delete stitching resources (in fact the switching rules). This module also can act as a proxy between the RO and stitching device. It receives requests from the RO, checks if they can be reserved, and configures the switching hardware. The SE-RM also checks for expired reservations triggering their teardown. The TN-RM functions in a similar manner, but allows an experimenter to make connections between the FELIX infrastructure and NSI based networks, i.e. it acts as a proxy between the RO and NSI networks.

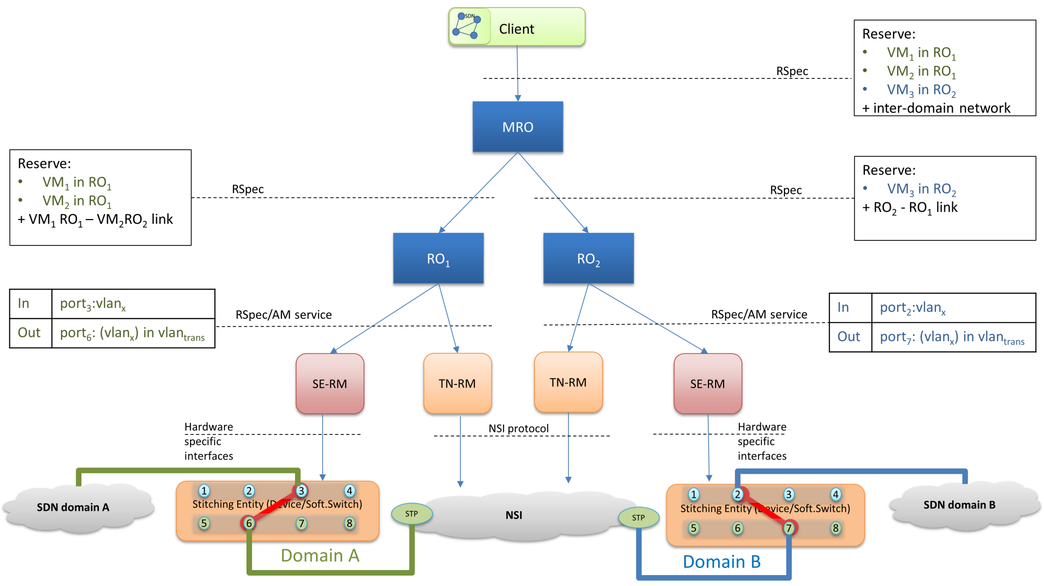

Figure 1.1: Overview of inter-domain networking between SDN slices

A single TN-RM and SE-RM have a many-to-one relationship with the RO and management communication is realized through the GENI interface. The RO itself coordinates communication between the RMs from each island. The SE-RM and TN-RM are key elements for providing inter-domain networking between SDN slices. The overview on Figure 1.1 presents concept of inter-domain networking.

Figure 1.2 and Figure 1.3 illustrate the general architecture of adduced RM's. More details are in each section dedicated for TN-RM and SE-RM.

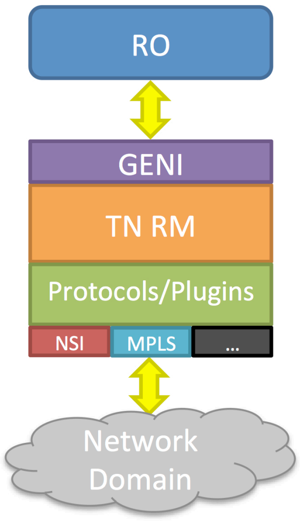

Figure 1.2: General architecture of TN-RM

As previously described in the FELIX Deliverable D2.2 [1], the TN-RM is responsible for :

- integrating with a different L1/L2 technologies in different network domains
- triggering connection operations on south interfaces
- communication from and to RO

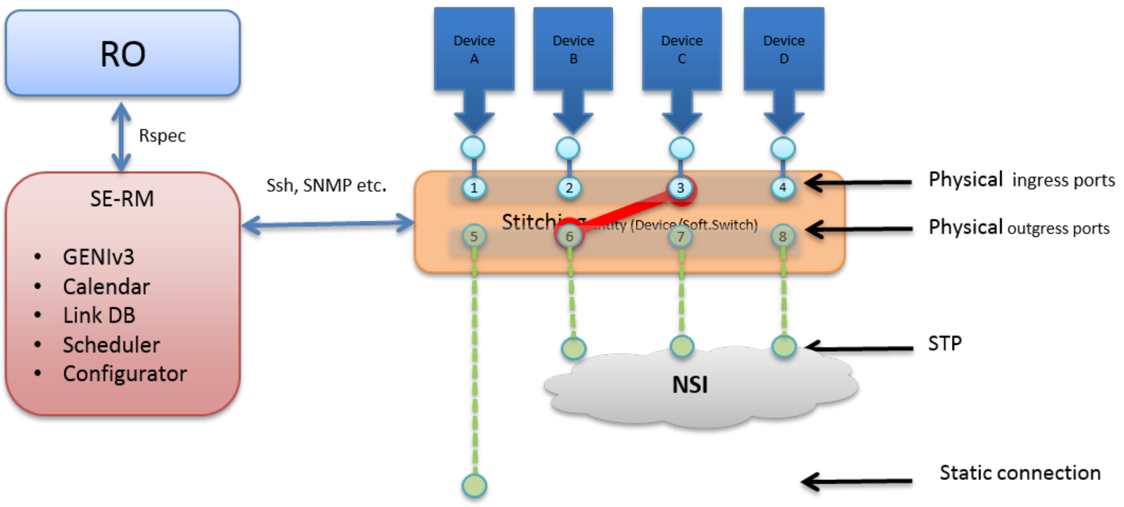

Figure 1.3: General architecture of SE-RM

The SE-RM is responsible for:

- triggering operations over devices based on NSI requests towards the locale domain,
- triggering the outgoing/incoming traffic from/to different regions (other SDN Exchange instances),

- switching the traffic between SDN islands directly connected (if no direct connectivity between them exists).

2 Definitions

Throughout this document we use specific notation and acronyms that are explained here. Please refer to this guide to identify the concept or for a more detailed explanation.

2.1 Abbreviations

- **AG**: Aggregator.
- **AM**: Aggregate Manager.
- **CH**: Clearinghouse.
- **ERO**: Explicit Routing Object.
- **MPLS**: Multi-Protocol Label Switching.
- **NMS**: Network Management System.
- **NRM**: Network Resource Manager.
- **NSI**: Network Services Interface.
- **PA**: Provider Agent.
- **RA**: Requester Agent.
- **RM**: Resource Manager.
- **RO**: Resource Orchestrator.
- **SE-RM**: Stitching Entity Resource Manager.
- **TN-RM**: Transit Network Resource Manager.
- **URN**: Uniform Resource Name.
- **VLAN**: Virtual Local Area Network.

2.2 Definitions

- **Island**: Physical domain under particular management. It provides infrastructure and resources to the end user.
- **OMNI**: CLI tool which is part of the GENI Control Framework, could be client for Resource Managers.
- **RSpec**: Resource specification describing a components in XML document. Following agreed schemas to it represent resources that are understood by Resource and Aggregate Managers.
- **Slice**: Set of slivers spanning a set of network components, plus an associated set of users that are allowed to access those slivers for the purpose of running an experiment on the substrate
- **Stitching**: Interconnecting multiple substrates which are under the control of Aggregate Managers.
- **Stitching Resource Element**: The purpose of this component is to interconnect the external connection points for the local AM.
- **URN**: Public identifiers given to resources in the network in order to uniquely identify and exhaustively describe the properties of the resource. For that, the *urn* scheme is followed.

3 Implementation Details

This section provides the implementation details related to the inter-island connectivity provisioning software. It explains the developed modules, presents the software design and each module's functionality. There are the two modules delivering the Inter-Islands Connectivity tools in FELIX project. The Transit Network Resource Manager is the NSI project dependent module. It provides a functionality to connect island for example using NSI protocol as required by FELIX. The other module is the Stitching Entity Resource Manager, which provides a set of functionality to interact with the network environment established by TN-RM, to allow access to the inter island network for the island resources. The following sections provide the implementation details of TN-RM and SE-RM.

3.1 Transit Network Resource Manager

During this stage, the main effort has been in development of the prototype and fixing bugs encountered. In keeping with previous guidance about reusing existing tools as much as possible from FELIX Deliverable D2.2[1], Both NSI and eiSoil[4] were leveraged for this effort. In FELIX the Java based NSI plugin is implemented to manage NSI STPs. General overview of the NSI framework has been presented in [5]. Details on the NSIv2 Requester and Provider Agents implementation are described in the NSI working group document at OGF[6].

The Python based eiSoil is used to communicate with the NSI domains. Many of the details of eiSoil are covered at the eiSoil Wiki. In brief, it is a lightweight, plugin-based, framework to create aggregation managers (AM) for network test-beds. AMs manage the allocation and provisioning of resources in a network test-bed. eiSoil provides the necessary "glue" between communication handlers and management logic as well as facilitates common tasks in AM development, which reduces duplication of work. It is important to recognize that in order to communicate between the NSI Java calls and the eiSoil Python calls, a third language Jython was implemented.

3.1.1 Design

There are 5 general processes that are executed between the Client and NSI domain as shown in Figure 3.1.

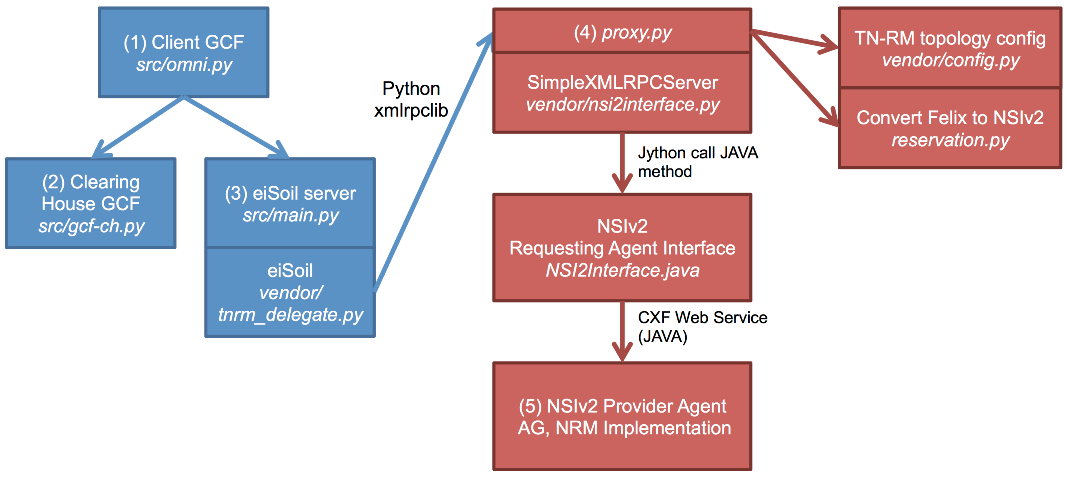

Figure 3.1: Diagram of the 5 processes needed to communicate between the client and the NSI domain, via the TN-RM.

In Step 1, the user makes a request for resources through a client. The resources are described with RSpec XML documents: Advertisement (presents what resources are available), Request (specifies the resources to be used by the client), and Manifest (shows the status of resources). In Step 2, the client obtains certificates/credentials from the Clearinghouse (CH) granting permission for the user request. In Step 3, eiSoil calls tnrm_delegate.py to implement the GENIv3DelegateBase. The tnrm_delegate.py in turn calls proxy.py, which is the front end of the SimpleXMLRPCServer to communicate with the NSIv2 NRM or AG. This is done by calling nsi2interface.py to invoke a Jython call to NSI2Interface.java. This Jython step is necessary in order to translate the Python calls of eiSoil to the Java calls in the NSI agents. Lastly, in Step 5, NSI2Interface.java calls the NSIv2 Requester to connect to NSIv2 provider (AG, NRM, etc.) by NSIv2 protocol using a CXF web service. It is important to note that Step 4 also requires network topology information. This is obtained by having proxy.py call a config.py script that reads a configuration file (config.xml) defined at a TN terminal point. The config.xml file contains node, interface, and NSIv2 STP information, as shown below.

```
<resources>
      <node component_id="urn:publicid:tn-network1:"
            component_manager_id="urn:publicid:IDN+NSI+authority+tnrm"
            exclusive="false">

        <interface felix_domain_id="urn:publicid:tn:aist:network1"
                   felix_stp_id="urn:ogf:network:aist:network1:stp1"
                   nsi_stp_id="urn:ogf:network:aist.go.jp:2013:bi-ps"
                   nsi_stp_id_in="urn:ogf:network:aist.go.jp:2013:in-ps"
                   nsi_stp_id_out="urn:ogf:network:aist.go.jp:2013:out-ps"
                   vlan="1779-1799"
                   capacity="10000" />
      <interface felix_domain_id="urn:publicid:tn:aist:network1" …. />
    </node>
    <node … …> … </node>
</resources>
```

After reading this file, config.py can create advertisement RSpec. Additionally, it must be able to convert FELIX nodes/interfaces to NSIv2 STPs and does so by calling reservation.py .

3.1.1.1 RSpec Definitions

In order to pass specific topology information between the RO and TN-RM, FELIX uses the standardized RSpec documents that are written in XML. The current version is GENI v3 and more information is available on the GENI Wiki[7]. There are three different types of RSpecs: advertisement, which describes the resources that the TN-RM has available; request, which the RO sends to a TN-RM to reserve resources; and manifest, which is returned by the TN-RM to the RO describing the resources that were reserved. All three were validated against the RSpec xml schema. Although, specific details of the RSpec are at the GENI Wiki[7], each of the three will be briefly described with respect to the FELIX architecture below.

An example physical network configuration between two islands with an NSI domain is depicted in Figure 3.2.

Based on this, an example of the FELIX advertisement RSpec is shown below. The advertisement RSpec shows the name of available ports using URN identifiers, which is the FELIX notation. Note that for the NSI domain, STPs are used with a range of VLAN IDs that are available. The first four interfaces specified are NSI STPs that belong to FELIX domains. The fifth and sixth interfaces specified are NSI STPs, which can be used in ERO (Explicit Routing Object) designations.

```
<?xml version="1.1" encoding="UTF-8"?>
<rspec type="advertisement"
       xmlns="http://www.geni.net/resources/rspec/3"
       xmlns:sharedvlan="http://www.geni.net/resources/rspec/ext/shared-vlan/1"
       xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
       xs:schemaLocation="http://www.geni.net/resources/rspec/3/ad.xsd
            http://www.geni.net/resources/rspec/ext/shared-vlan/1/ad.xsd">

    <node component_id="urn:publicid:tn-network1:"
          component_manager_id="urn:publicid:IDN+NSI+authority+tnrm" exclusive="false">
        <interface component_id="urn:publicid:tn:aist:network1+urn:ogf:network:aist
                                 :network1:stp1">
            <sharedvlan:available name="urn:publicid:tn:aist:network1+urn:ogf:network
                                 :aist:network1:stp1+vlan" description="1980-1989"/>
        </interface>
```

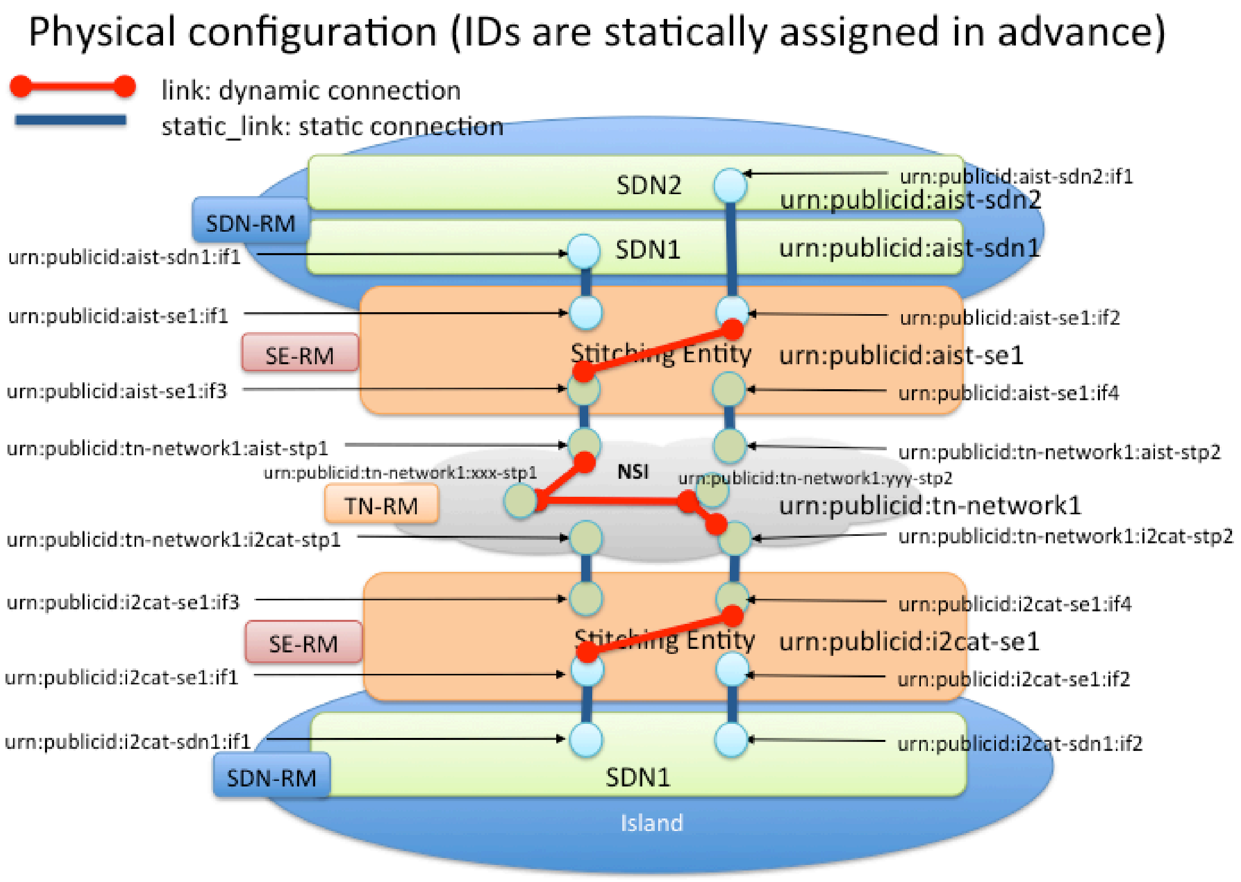
Physical configuration (IDs are statically assigned in advance)
link: dynamic connection
static_link: static connection
SDN2
SDN-RM
SDN1
urn:publicid:aist-sdn2:if1
urn:publicid:aist-sdn2
urn:publicid:aist-sdn1:if1
urn:publicid:aist-sdn1
urn:publicid:aist-se1:if1
urn:publicid:aist-se1:if2
SE-RM
Stitching Entity
urn:publicid:aist-se1
urn:publicid:aist-se1:if3
urn:publicid:aist-se1:if4
urn:publicid:tn-network1:aist-stp1
urn:publicid:tn-network1:aist-stp2
urn:publicid:tn-network1:xxx-stp1
NSI
urn:publicid:tn-network1:yyy-stp2
TN-RM
urn:publicid:tn-network1
urn:publicid:tn-network1:i2cat-stp1
urn:publicid:tn-network1:i2cat-stp2
urn:publicid:i2cat-se1:if3
urn:publicid:i2cat-se1:if4
SE-RM
Stitching Entity
urn:publicid:i2cat-se1
urn:publicid:i2cat-se1:if1
urn:publicid:i2cat-se1:if2
urn:publicid:i2cat-sdn1:if1
urn:publicid:i2cat-sdn1:if2
SDN-RM
SDN1
Island

Figure 3.2: Diagram of example physical configuration between two network islands with a NSI domain between.

```
        <interface component_id="urn:publicid:tn:aist:network1+urn:ogf:network:aist
                                :network1:stp2">
            <sharedvlan:available name="urn:publicid:tn:aist:network1+urn:ogf:network
                                :aist:network1:stp2+vlan" description="1980-1989"/>
        </interface>
        <interface component_id="urn:publicid:tn:i2cat:network1+urn:felix:i2cat-stp1">
            <sharedvlan:available name="urn:publicid:tn:i2cat:network1+urn:felix
                                :i2cat-stp1+vlan" description="1980-1989"/>
        </interface>
        <interface component_id="urn:publicid:tn:i2cat:network1+urn:felix:i2cat-stp2">
            <sharedvlan:available name="urn:publicid:tn:i2cat:network1+urn:felix
                                :i2cat-stp1+vlan" description="1980-1989"/>
        </interface>
        <interface component_id="urn:ogf:network:xxx:stp1">
            <sharedvlan:available name="urn:ogf:network:xxx:stp1+vlan"
                                description="1980-1989"/>
        </interface>
        <interface component_id="urn:ogf:network:yyy:stp2">
            <sharedvlan:available name="urn:ogf:network:xxx:stp2+vlan"
                                description="1980-1989"/>
        </interface>
    </node>
</rspec>
```

For the Request RSpec shown below, the port IDs and point-to-point links that are requested by the client are sent to the TN-RM. The interfaces used in the request are defined in the node section. In this example, the first and fourth interfaces specified are in FELIX domains, which are the endpoints of the requesting path. The second and third interfaces specified are for ERO designation, which is optional. Connections between the interfaces are specified in the link section. Each "property" tag describes a uni-directional path. A link can be uni-directional or bi-directional, however, if a bi-directional link is required, it must be specified as two uni-directional paths. The "stitch:path" tag is used to specify the ERO, which again is optional.

```
<?xml version="1.1" encoding="UTF-8"?>
<rspec type="request"
       xmlns="http://www.geni.net/resources/rspec/3"
       xmlns:sharedvlan="http://www.geni.net/resources/rspec/ext/shared-vlan/1"
       xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:stitch="http://hpn.east.isi.edu/rspec/ext/stitch/0.1/"
       xs:schemaLocation="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
            http://www.geni.net/resources/rspec/ext/shared-vlan/1/request.xsd">

    <node client_id="urn:publicid:tn-network1:" component_manager_id="urn
                       :publicid:IDN+NSI+authority+tnrm">
        <interface client_id="urn:publicid:tn:aist:network1+urn:ogf:network:aist
                       :network1:stp1">
            <sharedvlan:link_shared_vlan name="urn:publicid:tn:aist:network1+urn:ogf
                       :network:aist:network1:stp1+vlan" vlantag="1980-1989"/>
        </interface>
        <interface client_id="urn:ogf:network:xxx:stp1"/>
        <interface client_id="urn:ogf:network:yyy:stp2"/>
        <interface client_id="urn:publicid:tn-network1+urn:felix:i2cat-stp2">
            <sharedvlan:link_shared_vlan name="urn:publicid:tn-network1+urn:felix
                       :i2cat-stp2+vlan" vlantag="1980-1989"/>
        </interface>
     </node>
     <link client_id="urn:publicid:tn-network1:link">
        <component_manager name="urn:publicid:IDN+NSI+authority+tnrm"/>
        <interface_ref client_id="urn:publicid:tn:aist:network1+urn:ogf:network:aist
                       :network1:stp1"/>
        <interface_ref client_id="urn:ogf:network:xxx:stp1"/>
        <interface_ref client_id="urn:ogf:network:yyy:stp2"/>
        <interface_ref client_id="urn:publicid:tn-network1+urn:felix:i2cat-stp2 "/>
        <property source_id="urn:publicid:tn:aist:network1+urn:ogf:network:aist:network1
                      :stp1"dest_id="urn:publicid:tn-network1+urn:felix:i2cat-stp2 "
                      capacity="1000">
            <stitch:path id="urn:publicid:tn:aist:network1+urn:ogf:network:aist:network1:stp1+
                        urn:publicid:tn-network1+urn:felix:i2cat-stp2">
                <stitch:hop id="1">
                     <stitch:link id="urn:publicid:tn-network1:xxx-stp1">
                         <stitch:trafficEngineeringMetric>10</stitch:trafficEngineeringMetric>
                     </stitch:link>
                     <stitch:nextHop>2</stitch:nextHop>
                </stitch:hop>
                <stitch:hop id="2">
                     <stitch:link id="urn:publicid:tn-network1:yyy-stp2">
                         <stitch:trafficEngineeringMetric>10</stitch:trafficEngineeringMetric>
                     </stitch:link>
                </stitch:hop>
            </stitch:path>
        </property>
        <property source_id="urn:publicid:tn-network1+urn:felix:i2cat-stp2 "
                      dest_id="urn:publicid:tn-network1+urn:ogf:network:aist
                                 :network1:stp1" capacity="500">
            <stitch:path id="urn:publicid:tn-network1+urn:felix:i2cat-stp2+
                        urn:publicid:tn:aist:network1+urn:ogf:network:aist:network1:stp1">
                <stitch:hop id="1">
                     <stitch:link id="urn:publicid:tn-network1:yyy-stp2">
                         <stitch:trafficEngineeringMetric>10</stitch:trafficEngineeringMetric>
                     </stitch:link>
                     <stitch:nextHop>2</stitch:nextHop>
                </stitch:hop>
                <stitch:hop id="2">
                     <stitch:link id="urn:publicid:tn-network1:xxx-stp1">
                         <stitch:trafficEngineeringMetric>10</stitch:trafficEngineeringMetric>
                     </stitch:link>
                </stitch:hop>
            </stitch:path>
        </property>
    </link>
```

```
</rspec>
```

In the Manifest RSpec below, the TN-RM returns what connections were established according to the client's request. It is important to note, that the client does not select the VLAN to be used. Rather, the TN-RM selects a VLAN ID to be used and returns it as a VLANtag ID to the client. The VLANtag parameter is mandatory in the Manifest RSpec. Thus, in this example, since the request was fulfilled, all information returned is the same as in the Request RSpec and includes the specific (not a range) VLANtag ID.

```
<?xml version="1.1" encoding="UTF-8"?>
<rspec type="manifest"
       xmlns="http://www.geni.net/resources/rspec/3"
       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:xs="http://www.w3.org/2001/XMLSchema-instance"
       xs:schemaLocation="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/manifest.xsd
            http://www.geni.net/resources/rspec/ext/shared-vlan/1/request.xsd">

    <node client_id="urn:publicid:tn-network1:" component_manager_id="urn:publicid
                              :IDN+NSI+authority+tnrm">
        <interface client_id="urn:publicid:tn:aist:network1+urn:ogf:network:aist
                              :network1:stp1">
            <sharedvlan:link_shared_vlan name="urn:publicid:tn:aist:network1+urn:ogf
                              :network:aist:network1:stp1+vlan" vlantag="1983"/>
        </interface>
        <interface client_id="urn:ogf:network:xxx:stp1"/>
        <interface client_id="urn:ogf:network:yyy:stp2"/>
        <interface client_id="urn:publicid:tn-network1+urn:felix:i2cat-stp2">
            <sharedvlan:link_shared_vlan name="urn:publicid:tn-network1+urn:felix
                              :i2cat-stp2+vlan" vlantag="1983"/>
        </interface>
    </node>

    <link client_id="urn:publicid:tn-network1:link" vlantag="1983">
        <component_manager name="urn:publicid:IDN+AIST+authority+tnrm"/>
        <interface_ref client_id="urn:publicid:tn:aist:network1+urn:ogf:network:aist
                              :network1:stp1"/>
        <interface_ref client_id="urn:ogf:network:xxx:stp1"/>
        <interface_ref client_id="urn:ogf:network:yyy:stp2"/>
        <interface_ref client_id="urn:publicid:tn-network1+urn:felix:i2cat-stp2"/>
        <property source_id="urn:publicid:tn:aist:network1+urn:ogf:network:aist:network1
             :stp1" dest_id="urn:publicid:tn-network1+urn:felix:i2cat-stp2 " capacity="1000">
            <stitch:path id="urn:publicid:tn:aist:network1+urn:ogf:network:aist:network1:stp1+
                          urn:publicid:tn-network1+urn:felix:i2cat-stp2">
                <stitch:hop id="1">
                     <stitch:link id="urn:publicid:tn-network1:xxx-stp1">
                         <stitch:trafficEngineeringMetric>10</stitch:trafficEngineeringMetric>
                     </stitch:link>
                     <stitch:nextHop>2</stitch:nextHop>
                </stitch:hop>
                <stitch:hop id="2">
                     <stitch:link id="urn:publicid:tn-network1:yyy-stp2">
                         <stitch:trafficEngineeringMetric>10</stitch:trafficEngineeringMetric>
                     </stitch:link>
                </stitch:hop>
            </stitch:path>
        </property>
        <property source_id="urn:publicid:tn-network1+urn:felix:i2cat-stp2"
                     dest_id="urn:publicid:tn-network1+urn:felix:i2cat-stp2 " capacity="500">
                        <stitch:path id="urn:publicid:tn-network1+urn:felix:i2cat-stp2+
                             urn:publicid:tn:aist:network1+urn:ogf:network:aist:network1:stp1">
                <stitch:hop id="1">
                     <stitch:link id="urn:publicid:tn-network1:yyy-stp2">
                         <stitch:trafficEngineeringMetric>10</stitch:trafficEngineeringMetric>
                     </stitch:link>
                     <stitch:nextHop>2</stitch:nextHop>
                </stitch:hop>
                <stitch:hop id="2">
                     <stitch:link id="urn:publicid:tn-network1:xxx-stp1">
                         <stitch:trafficEngineeringMetric>10</stitch:trafficEngineeringMetric>
                     </stitch:link>
                </stitch:hop>
            </stitch:path>
        </property>
    </link>
</rspec>
```

3.1.1.2 GENIv3 to NSI call mapping

As mentioned above, the pluggable framework of eiSoil facilitates the communication between the GENIv3 based TN-RM and the NSIv2 domains. In Table 3.1, the GENIv3DelegateBase that was implemented and the corresponding NSIv2 call equivalent is presented. This is the mapping between the two different domains using eiSoil calls. Details on the methods supported by GENIv3 can be found on the GENI Wiki. The "PerformOperationalAction" method was not implemented as it was not needed in the TN-RM.

GENIv3/eiSoil API Methods	**NSIv2 call equivalent**	**Description**
GetVersion (not call TN-RM)	--	Get info about the AMs
ListResources	--	Return advertisement RSpec
Describe	--	Return manifest RSpec
Allocate	reserve, commit	Reserve a slice/sliver for a short time
Renew	modify, commit	Extend the usage of a slice/sliver
Provision	provision	Provision a reservation for a longer time
PerformOperationalAction	(Not implemented)	Change the operational state of a sliver
Delete	terminate	Remove a slice/sliver
Shutdown	terminate	Emergency stop a slice

Table 3.1: Mapping of GENI/eiSoil methods to equivalent NSI calls.

3.1.2 Workflows

The workflow diagrams are based on the scenario where the RM receives a request for connection from Y to Z, specifically a reservation request in the example network topology presented in Figure 3.3 below where ingress point = a (in Y network) and egress point = d (in Z network). SDP is the NSI Service Demarcation Point that consists of a pair of STPs on adjacent networks that are attached to each other.

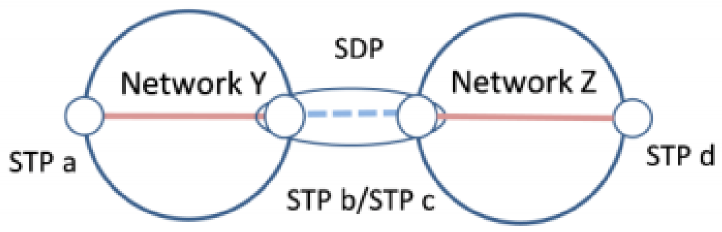

Figure 3.3: Diagram of network topology for reservation request workflow example.

The workflow diagram for this is shown in Figure 3.4 (this sequence diagram and any other appearing in this document are generated using www.websequencediagrams.com). In brief, a reservation request is sent to each network domain. If the requested resources are available, then a subsequent request is sent to commit the resources for the reservation followed by a provision request for those resources.

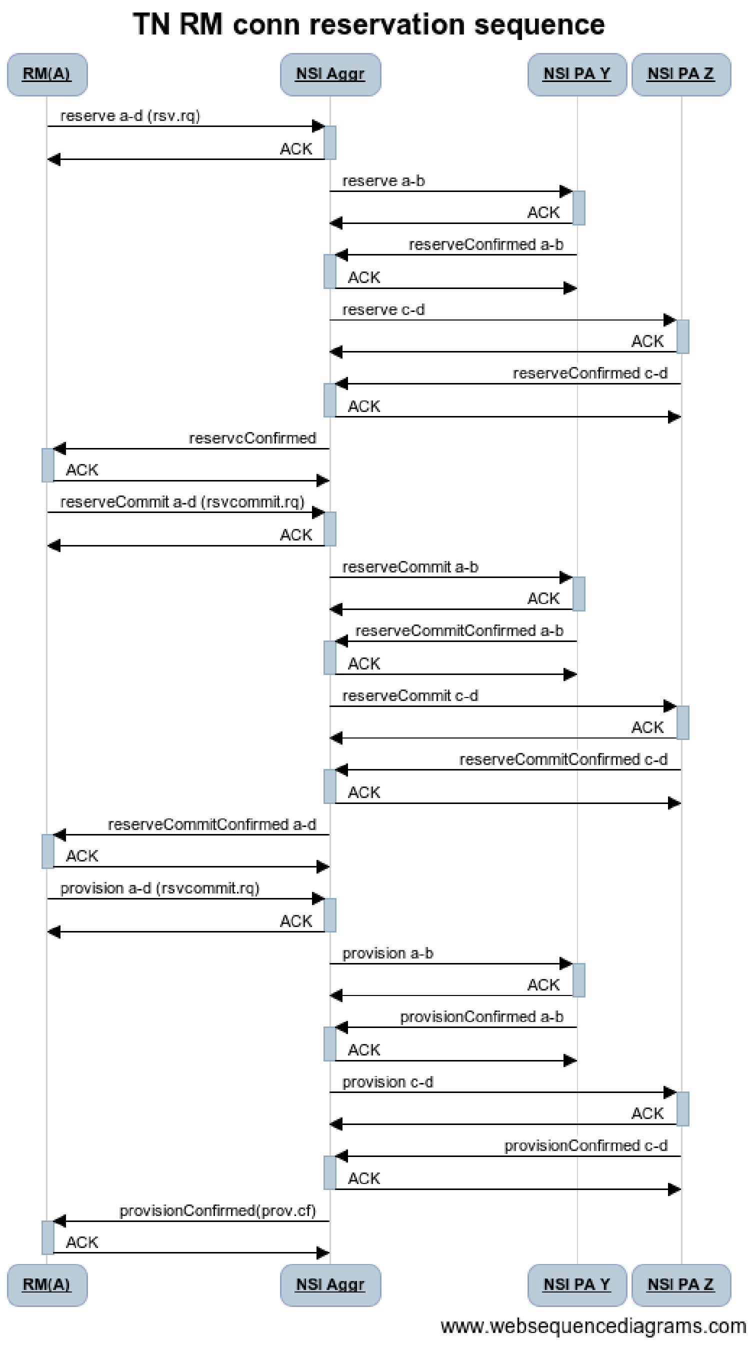

Figure 3.4: Workflow diagram for reservation request.

3.2 Stitching Entity Resource Manager

3.2.1 Design

The SE-RM is composed of a number of internal components each one of which provides a specific feature in the whole manager. The figure below presents the overall SE-RM decomposition with the relations between the internal components.

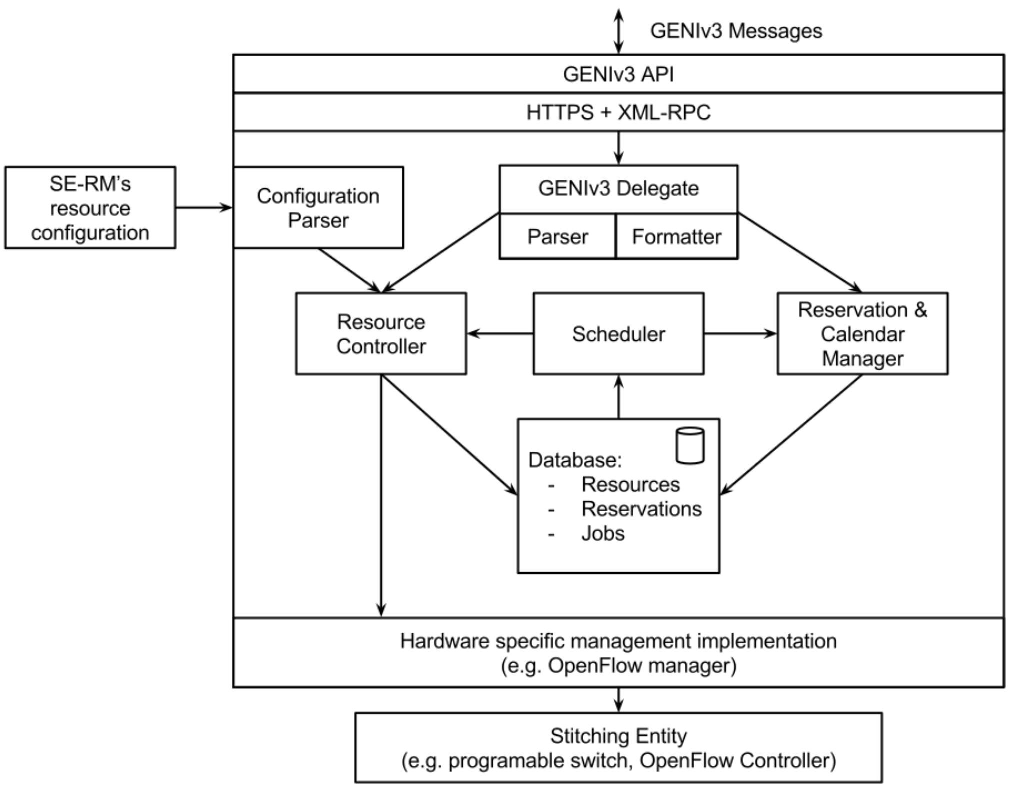
GENIv3 Messages
GENIv3 API
HTTPS + XML-RPC
SE-RM's resource configuration
Configuration Parser
GENIv3 Delegate
Parser
Formatter
Resource Controller
Scheduler
Reservation & Calendar Manager
Database:
- Resources
- Reservations
- Jobs
Hardware specific management implementation (e.g. OpenFlow manager)
Stitching Entity (e.g. programable switch, OpenFlow Controller)

Figure 3.5: Stitching Entity Resource Manager components

The SE-RM components are:

- **SE-RM configuration file**: this is a file that allows to specify which ports on the switching hardware take part in the FELIX testbed (NSI, STPs and static links). Some additional parameters can also be specified like VLAN tagging options or FELIX-specified identifiers.
- **Configuration parser**: this component extracts the initial configuration, then checks if change does not interrupt the on-going reservations and transforms all data according to the specific data template that can be stored in the SE-RM Database.
- **GENIv3 Delegate (with Parser and Formatter)**: this component is based on the GENI Aggregate Manager API and was created for FELIX project. It allows aggregates to advertise resources and allocate resources to slices in the form of slivers and is used by other. This component is also responsible for parsing.
- **Resource Controller**: this component gets the requests for deploying a connection between the ports (switching rule), checks their feasibility (e.g., through chceking available port and/or VLAN in configured range) and triggers them by translating real ports values through the hardware-specific management implementation.
- **Reservation & Calendar Manager**: this part of software is responsible for handling RSpec request in reservation context and stores all needed information in dedicated data model. Additionally it also works as calendar for managed resources. Moreover this component takes part in releasing resources process.
- **SE-RM Database**: it stores the status of the stitching resources like ports and associated VLANs. It also stores the on-going reservation with the deployment status and keeps scheduled timeout events. Finally, it stores information about resources to be released when reservation time expires.

- **Hardware specific management implementation**: this pluggable component implements the management features of a specific hardware and management interface. It can be e.g. an SSH client that invokes the CLI commands on the remote console or the OpenFlow client app.

Example SE-RM RSpecs request message

The request RSpec, passed when performing an Allocation operation, must comply with a format similar to the following:

```
<?xml version="1.1" encoding="UTF-8"?>
<rspec type="request"
       xmlns="http://www.geni.net/resources/rspec/3"
       xmlns:sharedvlan="http://www.geni.net/resources/rspec/ext/shared-vlan/1"
       xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
       xs:schemaLocation="http://www.geni.net/resources/rspec/3/request.xsd
            http://www.geni.net/resources/rspec/ext/shared-vlan/1/request.xsd">

    <node client_id="urn:publicid:aist-se1"
          component_manager_id="urn:publicid:IDN+AIST+authority+serm">
        <interface client_id="urn:publicid:aist-se1:eth1">
            <sharedvlan:link_shared_vlan name="urn:publicid:aist-se1:eth1+vlan"
                                         vlantag="1000"/>
        </interface>
        <interface client_id="urn:publicid:aist-se1:eth2">
            <sharedvlan:link_shared_vlan name="urn:publicid:aist-se1:eth2+vlan"
                                         vlantag="1000"/>
        </interface>
    </node>

    <link client_id="urn:publicid:aist-se1:eth1-eth2">
        <component_manager name="urn:publicid:IDN+AIST+authority+serm"/>
        <link_type name="urn:felix+vlan_trans"/>
        <interface_ref client_id="urn:publicid:aist-se1:eth1"/>
        <interface_ref client_id="urn:publicid:aist-se1:eth2"/>
    </link>
</rspec>
```

Its tags and attributes are explained in the following table:

Tag	Attribute	Type	Description	Advertisement	Request	Manifest
rspec	type	Fixed string	RSpec message type. Possible values: advertisement, request, manifest	X	X	X
node	client_id	URN	Unique switching device identifier.		X	X
node	component_id	URN	Unique switching device identifier.	X		
interface	client_id	URN	Unique physical port identifier.		X	X
interface	component_id	URN	Unique physical port identifier.	X		
link_shared_vlan	name	URN	Unique VLAN at specific physical port identifier.		X	X
link_shared_vlan	vlantag	Integer	Number of the VLAN.		X	X
link	client_id	String	Unique switching rule identifier.		X	X

link	link_type	Fixed string	VLAN translation type or static link. Required format: "urn:felix+<params>". Available parameters: vlan_trans -- Change the VLAN TAG QinQ -- Add S-VLAN static_link -- Use already provisioned link	X	X	X
link	interface_ref	String	Physical port identifiers that participate in switch-ing.	X	X	X
link	sliver_id	String	Unique provisioned switching rule identifier.		X	X

Table 3.2: SE-RM RSpec tags and attributes

3.2.2 Workflows

This section describes the workflows that the SE-RM follows to perform its basic operations (GENIv3 interface) on resources, e.g., list, allocate, delete, etc

3.2.2.1 Allocate and provision

Figure 3.6 below shows the allocate/provision workflow. When the GENIv3 message arrives (e.g. from RO) to SE-RM the *Reservation&Calendar Manager* passes it on to the *Resource Controller* that checks the availability of the resources and triggers the adding of new switch entry. *Resource Controller* updates the resources state in the database and sends acknowledgment to the *Reservation&Calendar Manager*. The *R&C Manager* creates then the slice reservation in database. The GENIv3 message (RSpec Manifest) is formatted and sent back to the RO.

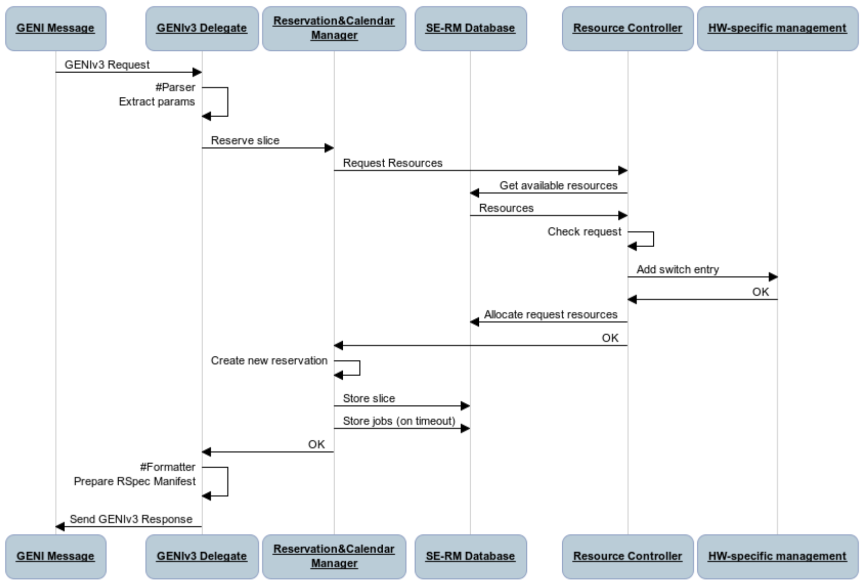

Figure 3.6: Allocate slice and provision the switch entry on Stitching Entity

3.2.2.2 Scheduled delete

Figure 3.7 shows the delete workflow. The SE-RM is configured to use a scheduler for executing jobs at specified points of time in calendar. One of the job types is "release the resources at given time" (trigger the switch entry teardown). The scheduler in *Reservation&Calendar Manager* checks the delegated jobs in a background worker. When it finds the teardown job, it invokes teardown request to *Resource Controller*. The *Resource Controller* translates it into a request for *HW-specific management* and by that removes the proper entries from device. After that the *Resource Controller* releases the resources in database and sends acknowledgment to the *Reservation&Calendar Manager*.

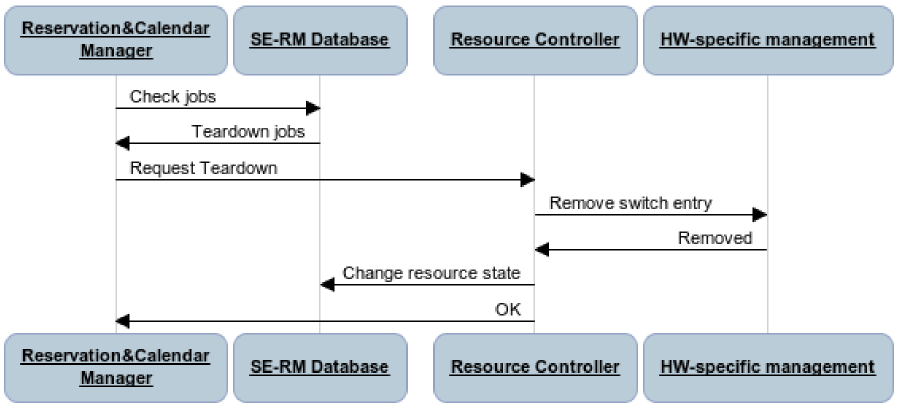

Figure 3.7: SE-RM scheduler triggers the switch entry remove on Stitching Entity

4 Deployment

This section provides some details on how to deploy the Transit Network Resource Manager (TN-RM) and the Stitching Entity Resource Manager (SE-RM). Below, we try to briefly explain how to:

- Get the code from the GIT-based FELIX repository [8].
- Install the dependencies for each component and satisfy its requirements.
- Install and configure the component itself.

Additionally, some useful information are given on the operations that the users can do when the modules are correctly up and running on the system.

4.1 Transit Network Resource Manager

4.1.1 Requirements and Dependencies

There are no dependencies for eiSoil, however, you will need jar of NSIv2.

4.1.2 Configuration and Installation

There is no separate installation except for the java library. The source code is available from the git repository:

```
$ git clone https://github.com/dana-i2cat/felix/tree/aist
```

The library of the required java is the following.

```
$ ls -l ${CXF_HOME}
lrwxrwxrwx 1 root root 17 Jun 12 2014 /opt/cxf -> apache-cxf-2.7.11

LIBS=$LIBS:${NSI_HOME}/clientapi/build/jar/nsi2_ client.jar
LIBS=$LIBS:${NSI_HOME}/nrm/lib/commons-logging-1.1.1.jar
LIBS=$LIBS:${NSI_HOME}/nrm/lib/log4j-1.2.13.jar

for f in ${CXF_HOME}/lib /*.jar
do
 LIBS = $LIBS:$f
done
```

Please note that to access the TN-RM you need Provider Agent or Aggregator of NSIv2 running. Now this is configured in nsi2interface.py and the following change must be made in order to make the connection.

```
AIST-NRM: pURL = 'https://127.0.0.1:22311/aist_upa/Services/ConnectionProvider'
AIST-DUMMY-NRM: pURL = 'http://127.0.0.1:28080/provider/services/ConnectionProvider'
```

proxy:

```
$ cd ${eiSoil_HOME}/src/vendor /tnrm; ./proxy.sh
```

CH:

```
$ cd ${GCF_HOME}; python src/gcf-ch.py --host=163.220.30.xxx --port 8000
```

eiSoil:

```
$ cd  ${eiSoil_HOME}/src; python main.py
```

4.1.3 Operation

The TN-RM can be tested as a standalone application without RO due to its compatibility with a 3rd party command line tool Omni [9]. This tool is shipped with GENI Control Framework (GCF) software package and allows communication with all FELIX RMs to list resources or make the reservations. To use the Omni tool the Clearing House (CH) should be installed and deployed. In addition, any certificate keys should be exchanged between the Clearing House, Omni and TN-RM.

4.1.3.1 List Resources

To list resources from the TN-RM enter the Omni directory and run command:

```
$ python src/omni.py -o --project tnrm -a https://163.220.30.145:8001 -V 3 listresources
```

As a successful result the RSpec Advertisement will be saved in the same directory and a summary will appear on console output:

```
......
Wrote rspecs from 1 aggregate(s) to 1 file(s)
Saved listresources RSpec from 'unspecified_AM_URN'
https://163.220.30.145:8001
to file rspec-163-220-30-145-8001.xml;
```

4.1.3.2 Allocate Slice

To allocate resources from the TN-RM enter the Omni directory and run command:

```
$ git clone https://github.com/dana-i2cat/felix/tree/aist
```

As a successful result the RSpec Manifest will be saved in the same directory and a summary will again appear on console output:

```
$ ls -l ${CXF_HOME}
lrwxrwxrwx 1 root root 17 Jun 12 2014 /opt/cxf -> apache-cxf-2.7.11

LIBS=$LIBS:${NSI_HOME}/clientapi/build/jar/nsi2_ client.jar
LIBS=$LIBS:${NSI_HOME}/nrm/lib/commons-logging-1.1.1.jar
LIBS=$LIBS:${NSI_HOME}/nrm/lib/log4j-1.2.13.jar

for f in ${CXF_HOME}/lib /*.jar
do
 LIBS = $LIBS:$f
done
```

TN-RM will send to NSIv2 NRM the following reservation request.

```
AIST-NRM: pURL = 'https://127.0.0.1:22311/aist_upa/Services/ConnectionProvider'
AIST-DUMMY-NRM: pURL = 'http://127.0.0.1:28080/provider/services/ConnectionProvider'
```

4.1.3.3 Renew Slice

To renew expired time for a slice from the TN-RM enter the Omni directory and run command:

```
$ cd ${eiSoil_HOME}/src/vendor /tnrm; ./proxy.sh
```

As a successful result the RSpec Manifest will again be saved in the same directory and a summary will appear in the console showing the updated expiration time:

```
$ cd ${GCF_HOME}; python src/gcf-ch.py --host=163.220.30.xxx --port 8000
```

TN-RM will send to NSIv2 NRM the following modification request.

```
$ cd  ${eiSoil_HOME}/src; python main.py
```

4.1.3.4 Provision Slice

To provision a slice from the TN-RM enter the Omni directory and run command:

```
$ python src/omni.py -o --project tnrm -a https://163.220.30.145:8001 -V 3 listresources
```

As a successful result the RSpec Manifest will be saved in the same directory and a summary will appear on console output:

```
......
Wrote rspecs from 1 aggregate(s) to 1 file(s)
Saved listresources RSpec from 'unspecified_AM_URN'
https://163.220.30.145:8001
to file rspec-163-220-30-145-8001.xml;
```

TN-RM will send to NSIv2 NRM the following provision request and the NSI dataplane is activated now.

```
libssl-dev python-m2crypto build-essential python-all-dev libssl-dev
swig python-setuptools openssl mongodb-server
```

4.1.3.5 Delete Slice

To delete a slice from the TN-RM enter the Omni directory and run command:

```
python-dateutil pyyaml pyopenssl m2crypto lxml pymongo apscheduler
requests flask flask-pymongo blinker flup argparse Flask-XML-RPC
networkx
```

As a successful result the RSpec Manifest will be saved in the same directory and a summary will appear on console output with notification of deletion:

```
$ git clone https://github.com/dana-i2cat/felix.git
$ git checkout stitching-element
```

TN-RM will send NSIv2 NRM the following terminate request and the NSI dataplane is in teardown, terminating the reservation.

```
$ cd felix/modules/resource/manager/stitching-entity/deploy/
$ ./install.sh
```

4.1.3.6 Parameters for operations

example_slice_name -- urn name for allocated slice

request.xml -- example Rspec request slivers

```
$ cd src/
$ python main.py
```

4.2 Stitching Entity Resource Manager

4.2.1 Requirements and Dependencies

This is the list of requirements and dependencies to install SE-RM. Basically, it needs a Linux-based operating system with installed Python runtime environment extended with Python libraries listed in this section. Moreover, it requires locally installed MongoDB database.

4.2.1.1 Requirements

The tested working environment ia a Debian-based distribution. Specifically, the module was tested both on *Debian 7 (Wheezy)* and *Ubuntu Ubuntu 14.04.1 LTS (Trusty Tahr)*. Althought, it should run on other Linux-based operation systems if all needed dependencies can be satisfied. SE-RM module needs the MongoDB database instance to be installed and configured to be available from the module by localhost interface and default port that is *27017*. The module connects to the MongoDB server and assumes a default database name of *felix_se*. A helper install and configuration scripts can be find in deploy directory (see section "Configuration and Installation").

4.2.1.2 Dependencies

There is a number of packages required for SE-RM to work. These are retrieved & installed by Debian's Advanced Packaging Tool (apt-get) or the Python version system (easy_install, pip).

4.2.1.3 Advanced Packaging Tool (APT)

The following Debian packages must be installed on the system for SE-RM to properly work:

```
libssl-dev python-m2crypto build-essential python-all-dev libssl-dev
swig python-setuptools openssl mongodb-server
```

4.2.1.4 Python Package Index (PIP)

The following python packages must be installed on the system for SE-RM to properly work:

```
python-dateutil pyyaml pyopenssl m2crypto lxml pymongo apscheduler
requests flask flask-pymongo blinker flup argparse Flask-XML-RPC
networkx
```

4.2.2 Configuration and Installation

4.2.2.1 Installation

Get the source code from git repository and switch to *stitching-element* branch:

```
$ git clone https://github.com/dana-i2cat/felix.git
$ git checkout stitching-element
```

Now, enter the SE-RM's deploy directory and run the installation script (root account or sudo is needed):

```
$ cd felix/modules/resource/manager/stitching-entity/deploy/
$ ./install.sh
```

All the required Debian packages and Python libraries should be installed automatically. Enter the module's root directory. The default SE-RM installation is supplied with a template configuration file. The module can be now run with this example configuration but will not be operating on real resources. Please make sure you proceed with the next step "Configuration" before running the final instance. To run the SE-RM, go to *src* directory and run the *main.py* script:

```
$ cd src/
$ python main.py
```

Logs are collected in **log/stitching-entity.log**

4.2.2.2 Configuration

The SE-RM configuration can be changed in set of files in *conf/* directory. Below the description of each configuration file is given.

4.2.2.3 Stitching Element RM Resources

- **se-config.yaml**

This configuration file contains a list of ports on Stitching Element that are under SE-RM management. For each port the VLANs that take part in translating can be specified or alternatively, the remote endpoint label of static link. Besides, the transport VLAN setting can be chosen (QinQ, VLAN translation). The configuration also contains SE-RM Rspec specific parameters like, URN identifiers or link capacity. Please note that this configuration is a YAML file and suggested syntax for YAML files is to use two spaces for indentation however YAML follows whatever indentation system is employed. In the following an example configuration file is presented.

```
component_id: urn:publicid:aist-se1
component_manager_id: urn:publicid:IDN+AIST+authority+serm

vlan_trans: true
qinq: false
capacity: 1G

interfaces:
    eth1:
        1000 : true

    eth2:
        1000 : true
        2000 : true
        3000 : true

    eth3:
        dp2 : urn:publicid:aist-sdn1:if2

    eth4:
        dp1 : urn:publicid:aist-sdn1:if1
```

4.2.2.4 XML-RPC Server

- **flask.conf**

This file contains the properties of the FLASK XML-RPC Server [10] like IP address, port or debug mode.

```
[general]
host = 0.0.0.0
port = 8447
debug = True

[fcgi]
# Development server
enabled = False
port = 8450

[certificates]
force_client_certificate = True
```

4.2.2.5 GENIv3

- **geniv3.conf**

This file contains parameters of the RSpec validation mode and directory to the certificates.

```
[general]
rspec_validation = True

[certificates]
cert_root = cert/trusted
```

4.2.2.6 Logging

- **log.conf**

This file contains the Logger parameters like logging format, level or output filename.

```
[general]
name = stitching-entity
level = logging.DEBUG
format = \%(asctime)s [\%(levelname)s] - \%(message)s
file = stitching-entity.log
```

4.2.3 Operation

The SE-RM can be tested as a standalone application without RO thanks to compatibilty with a 3rd party command line tool Omni [9]. This tool is shipped with GENI Control Framework (GCF) software package and allows to communicate with all FELIX RMs to make list the resources or make the reservations. To use the Omni tool the Clearing House (CH) should be installed and deployed. In addition, the certificate keys should be exchanged between the Clearing House, Omni and SE-RM.

4.2.3.1 List Resources

To list resources from running SE-RM enter the Omni directory and run command:

```
$ python src/omni.py -o -a https://127.0.0.1:8447/xmlrpc/geni/3/ -V 3 --debug \
 -c ~/.gcf/omni_config --no-compress --available listresources
```

As a successful result the RSpec Manifest will be saved in the same directory and the summary will appear on on console output:

```
......
Wrote rspecs from 1 aggregate(s) to 1 file(s)
Saved listresources RSpec from 'unspecified_AM_URN'
(url 'https://127.0.0.1:8447/xmlrpc/geni/3/')
to file rspec-127-0-0-1-8447-xmlrpc-geni-3.xml;
```

4.2.3.2 Allocate Slice

To allocate resources from running SE-RM enter the Omni directory and run command:

```
$ python src/omni.py -o -a https://127.0.0.1:8447/xmlrpc/geni/3/ -V 3 -c ~/.gcf/omni_config \
 --no-compress --available allocate example_slice_name \
 request_rspec_example.xml --end-time 201502312200
```

4.2.3.3 Delete Slice

```
$ python src/omni.py -o -a https://127.0.0.1:8447/xmlrpc/geni/3/ -V 3 -c ~/.gcf/omni_config \
 --no-compress --available delete example_slice_name
```

4.2.3.4 Parameters for operations

example_slice_name -- urn name for allocated slice

request_rspec_example.xml -- example Rspec request slivers

```
<?xml version="1.1" encoding="UTF-8"?>
<rspec type="request"
       xmlns="http://www.geni.net/resources/rspec/3"
       xmlns:sharedvlan="http://www.geni.net/resources/rspec/ext/shared-vlan/1"
       xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
       xs:schemaLocation="http://www.geni.net/resources/rspec/3/request.xsd
            http://www.geni.net/resources/rspec/ext/shared-vlan/1/request.xsd">

    <node client_id="urn:publicid:aist-se1"
          component_manager_id="urn:publicid:IDN+AIST+authority+serm">
        <interface client_id="urn:publicid:aist-se1:if2">
            <sharedvlan:link_shared_vlan name="urn:publicid:aist-se1:if2+vlan"
                                         vlantag="25"/>
        </interface>
        <interface client_id="urn:publicid:aist-se1:if3">
            <sharedvlan:link_shared_vlan name="urn:publicid:aist-se1:if3+vlan"
                                         vlantag="1983"/>
        </interface>

        <interface client_id="urn:publicid:aist-se1:if5">
            <sharedvlan:link_shared_vlan name="urn:publicid:aist-se1:if5+vlan"
                                         vlantag="55"/>
        </interface>
        <interface client_id="urn:publicid:aist-se1:if6">
            <sharedvlan:link_shared_vlan name="urn:publicid:aist-se1:if6+vlan"
                                         vlantag="1986"/>
        </interface>

    </node>

    <link client_id="urn:publicid:aist-se1:if2-if3">
        <component_manager name="urn:publicid:IDN+AIST+authority+serm"/>
        <link_type name="urn:felix+vlan_trans"/>
        <interface_ref client_id="urn:publicid:aist-se1:if2"/>
        <interface_ref client_id="urn:publicid:aist-se1:if3"/>
    </link>

    <link client_id="urn:publicid:aist-se1:if5-if6">
        <component_manager name="urn:publicid:IDN+AIST+authority+serm"/>
        <link_type name="urn:felix+vlan_trans"/>
        <interface_ref client_id="urn:publicid:aist-se1:if5"/>
        <interface_ref client_id="urn:publicid:aist-se1:if6"/>
    </link>

</rspec>
```

5 Conclusions and Summary

For both the TN-RM and SE-RM, after the core functionality is ready and stable, the next steps will focus on finishing the SE and TN hardware specific part implementation and integration with other components in FELIX architecture. The maintenance of code will also be necessary. In addition to this the implementation efforts could be modified during the test phase. Moreover, possible additional work on both implementations will be needed to address the FELIX Use Cases needs and any issues that arise during the Use Case execution.

References

[1] ``Deliverable D2.2: General Architecture and Functional Blocks.'' http://www.ict-felix.eu/wp-content/uploads/2014/03/FELIX_D2.2_General_Architecture_and_Functional_Blocks.pdf, 2014.

[2] ``GENI (Global Environment for Network Innovations) - website.'' http://www.geni.net.

[3] ``GENI Network Stitching - Overview.'' https://wiki.maxgigapop.net/twiki/pub/GENI/NetworkStitching/geni-network-stitching-architecture-overview.pdf.

[4] ``eiSoil Wiki.'' https://github.com/EICT/eiSoil/wiki.

[5] T. Kudoh, et al., ``Network services interface: An interface for requesting dynamic inter-datacenter networks,'' *Optical Fiber Communication Conference (OFC)*, Mar. 2013.

[6] G. Roberts, et al., ``Network service framework v2.0,'' Jun. 2014.

[7] ``GENI v3 RSpec Schema.'' http://groups.geni.net/geni/wiki/RSpecSchema3.

[8] ``Felix repository website.'' https://github.com/dana-i2cat/felix.

[9] ``GENI Omni installation guide.'' http://trac.gpolab.bbn.com/gcf/wiki/QuickStart.

[10] ``Flask xmlrpc website.'' https://pythonhosted.org/Flask-XML-RPC/.

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.

Project:	FELIX (Grant Agr. No. 608638)
Deliverable Number:	D3.3
Date of Issue:	30/01/2015