A L I E N

ABSTRACTION LAYER FOR IMPLEMENTATION OF EXTENSIONS IN PROGRAMMABLE NETWORKS

Collaborative project co-funded by the European Commission within the Seventh Framework Programme

Grant agreement no: 317880
Project acronym: ALIEN
Project full title: "Abstraction Layer for Implementation of Extensions in programmable Networks"
Project start date: 01/10/12
Project duration: 24 months

Deliverable D4.3
Final prototype of management software

Due date:	31/07/2014
Submission date:	12/08/2014
Deliverable leader:	Roberto Doriguzzi Corin (CREATE-NET)
Internal Reviewer:	Grzegorz Danilewicz (PUT)
Author list:	Roberto Doriguzzi Corin (CREATE-NET), Michele Santuari (CREATE-NET), Ali Hammad (UnivBris), Krzysztof Dombek (PSNC), Umar Toseef (EICT), Adel Zaalouk (EICT), Jon Matias (EHU/UPV)

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)

Table of Contents

Figure Summary

Table Summary

Acronyms

[AAA]	Authentication, Authorization and Accounting
[ACF]	ALIEN Control Framework
[AM]	Aggregate Manager
[CH]	ClearingHouse
[CLI]	Command Line Interface
[DPID]	DataPath ID
[FOAM]	FlowVisor OpenFlow Aggregate Manager
[GUI]	Graphical User Interface
[HAL]	Hardware Abstraction Layer
[LDAP]	Lightweight Directory Access Protocol
[OCF]	OFELIA Control Framework
[OFGW]	OpenFlow GateWay
[OFELIA]	OpenFlow in Europe: Linking Infrastructure and Applications
[OvS]	Open vSwitch
[RPC]	Remote Procedure Call
[SFA]	Slice Facility Architecture
[TB Plugin]	Time-Based Plugin
[TBAM]	Time-Based Aggregate Manager
[TBAM RM]	TBAM Resource Manager
[UI]	User Interface
[URN]	Uniform Resource Name

Executive summary

T4.3 is the last task of WP4 and its objective is to release the final prototype of the management software for ALIEN devices. During this task we have already released an intermediate document describing a preliminary version of the software. The nature of D4.3 is “Prototype” and this document is meant as a description of each single module composing the management software. In particular, this document extends the previous one (released as Milestone MS13) by updating the description of the modules that have been improved and by adding some sections/chapters for the modules that were not available in the preliminary release.

In brief, this document describes how to install and configure the management software for the ALIEN hardware (also called ALIEN Control Framework) and how to connect it to the OFELIA facility [OFELIA] in order to make the ALIEN resources available to the OFELIA research community.

Additionally, this document provides the description of the efforts towards the implementation of a standardized management interface for ALIEN devices, such as NetConf, as proposed with the Contract Amendment 1.

A demonstration of the implementation results of Task T4.3 will be shown during the Final Review of the project and documented within the Deliverable D5.3.

.

Introduction

In this document we describe the details of the implementation of the ALIEN Control Framework (ACF), a pragmatic approach that allows the integration of the ALIEN devices within the OFELIA Control Framework (OCF) [OCF] in a time-base fashion (only one experiment at a time is allowed). The ACF's primary purpose is to allow the researchers to request for ALIEN network resources along with standard OFELIA resources from a single Graphical User Interface (GUI).

The main motivation behind the ACF is that the current OCF leverages on FlowVisor [FlowVisor] to allow the sharing of network resources. On the other hand, the ALIEN hardware can implement OpenFlow version 1.2 or 1.3 and some particular extensions while FlowVisor only supports version 1.0 of the protocol. In order to allow the experimentation with any version of the protocol (even with extensions outside the standard), the ACF has been designed to avoid the inspection of the OpenFlow protocol by replacing FlowVisor with a TCP proxy that forwards the control messages to the user's controller without any flowspace slicing operation.

With this approach only one experiment at time is allowed and the management of the time-slots booking process is managed by a new aggregate manager called Time-Based Aggregate Manager (TBAM) and configured by the users via the Expedient [Expedient]. Although this approach does not allow multiple simultaneous experiments, on the other hand, the user has exclusive access to the devices, therefore he/she is allowed to reconfigure or re-program them with new features or extension of the control and management protocol.

The TBAM, the core component of the ACF's architecture, has been developed to meet the requirements identified in deliverable [D4.1] and its architecture is described in deliverable [D4.2]. The TBAM integrates a calendar-like aggregate manager (the TBAM Scheduling Plugin), a resource manager which performs the actual provisioning of the slices (the TBAM RM) and a module that translates the user's requests coming from Expedient for the RM (the TBAM Delegate). The TBAM is connected southbound to ALIEN resources via the OpenFlow Gateway (OFGW) and northbound to Expedient via the Time Based plugin (TB Plugin) as depicted on Figure 0.1. The OFGW is in charge of:

- manage and configure the network devices;
- forward the data-plane traffic between the ALIEN resources and the OFELIA facility;
- forward the control-plane traffic between the ALIEN devices and the user's OpenFlow controller.

In addition, the TB Plugin provides an extension of the Expedient GUI for managing the ALIEN reservations and implements a GENI APIv3 [GENI-APIv3] southbound interface. All the modules have been developed within ALIEN Work Package 4.
The aim of this document is to provide an overview of the final implementation with a description of the functionalities provided and of the installation procedures of the prototype. The document is organized as follows: Chapter 1 provides the workflow of an experiment with particular attention to the interaction between the aforementioned modules. The chapters 2, 3, and 4 describe the installation and configuration procedures of the software modules that allow the integration of the ALIEN devices within the OCF, respectively TB Plugin, TBAM and OFGW. Chapter 5 introduces the ClearingHouse, a novel Authentication, Authorization and Accounting (AAA) mechanism for experimental facilities. Finally, Chapter 6 introduces the so-called NetConf Aggregate Manager (NetConf AM), an alternative AM that enables the ALIEN devices with a standardized management interface (NetConf [RFC6241]).

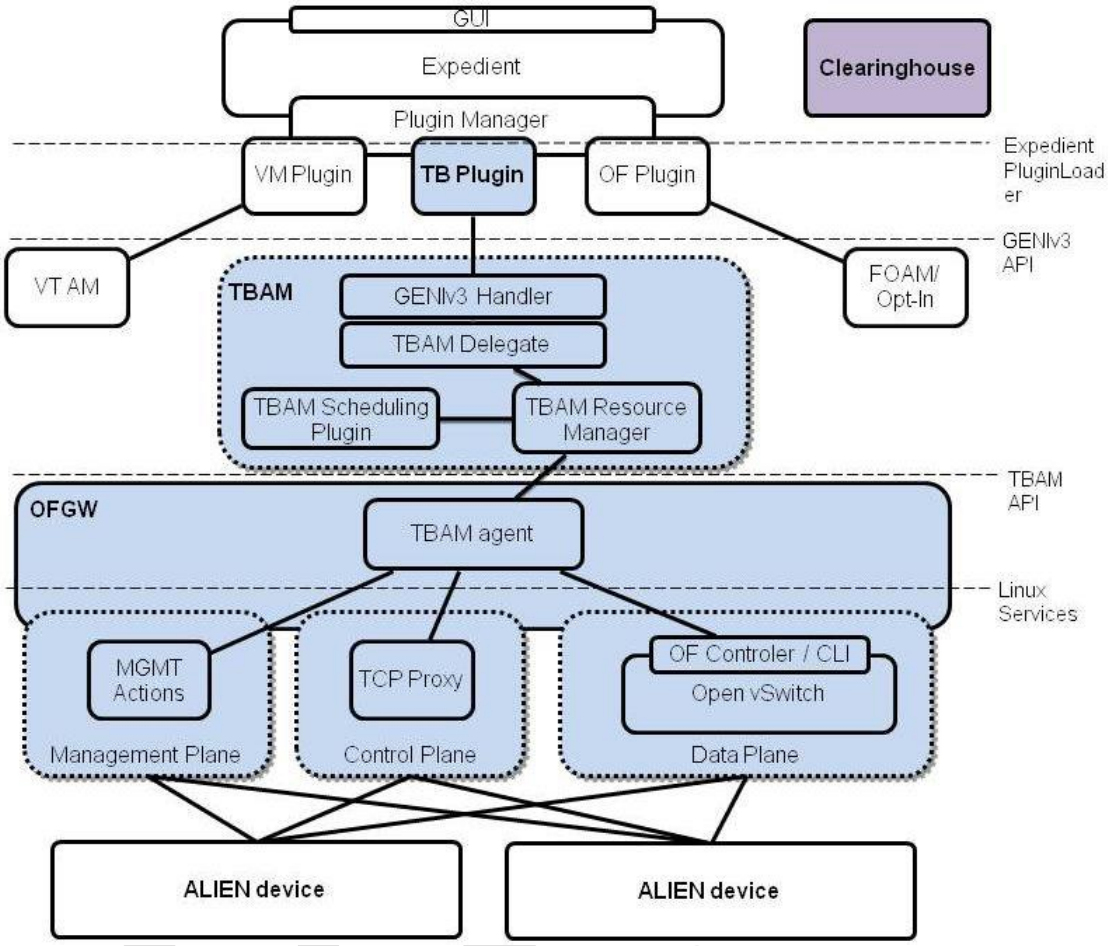


Figure 0.1: The architecture of the ALIEN Control Framework proposed in deliverable [D4.2]

1 The workflow of a new experiment

Each island belonging to the OFELIA facility [OFELIA] must provide a local installation of the OFELIA Control Framework (OCF) which is composed of Expedient, Opt-In (or FOAM), FlowVisor and the VT-AM for the Virtual Machines [OFELIA paper]. Expedient has to be connected to the remote AMs (such as Opt-In/FOAM and VT-AM) instances running on the other islands to allow multi-island experiments to be configured from each Expedient.

In the standard OFELIA experimental environment, when a user wants to run an experiment, he/she has to follow a preliminary procedure that consists of:

1. Registration to the OFELIA facility (information can be found in the OFELIA web-site [OFELIA]). The user's credentials are sent to the central OFELIA Lightweight Directory Access Protocol (LDAP) server;
2. Access one of the Expedients hosted in the OFELIA islands and create a new project;
3. Once the island administrator has approved the project, instantiation of a new slice (experiment) by adding OpenFlow aggregates (OFELIA islands) and computational resources (Virtual Machines (VMs));
4. Request for the flowspace (subset of available switches and subset of available ports), setup of the controller's IP address and TCP port, instantiation of one or more VMs;
5. Allocation of the resources by pressing the "Start slice" button on the Expedient GUI. The resources are not available until the administrators of all islands included in the slice have approved the experiment;
6. The system assigns a VLAN ID that tags all the traffic of the experiment crossing the OFELIA facility and that is used by the FlowVisor instances to perform the slicing operations;
7. Finally, the user can configure the VMs network interfaces with the assigned VLAN and custom IPs and can start generating the network traffic from the VMs.

On the other hand, ALIEN islands have to install the Alien Control Framework (ACF) in order to manage the experiments and to allow the users to access the ALIEN devices. Moreover, the ACF allows the instantiation of experiments that involves both ALIEN and OFELIA resources from a unified GUI provided that the following pre-requisites are met (Figure 1.1 gives an overview of the required setup):

1. The ALIEN island is connected to the OFELIA facility via both data plane and management network;
2. The ALIEN Control Framework has been successfully installed and configured to manage the underlying ALIEN devices;
3. The Expedient's OF-Plugin of the ACF is authorized to allocate OpenFlow resources of standard OFELIA islands through either Opt-In or FOAM aggregate managers (optionally, the TB-Plugin can be installed on standard OFELIA islands to allow ALIEN resources to be reserved also from their Expedient instances);
4. The Expedient's LDAP client is connected to the OFELIA central LDAP server to allow registered users to login into the facility.

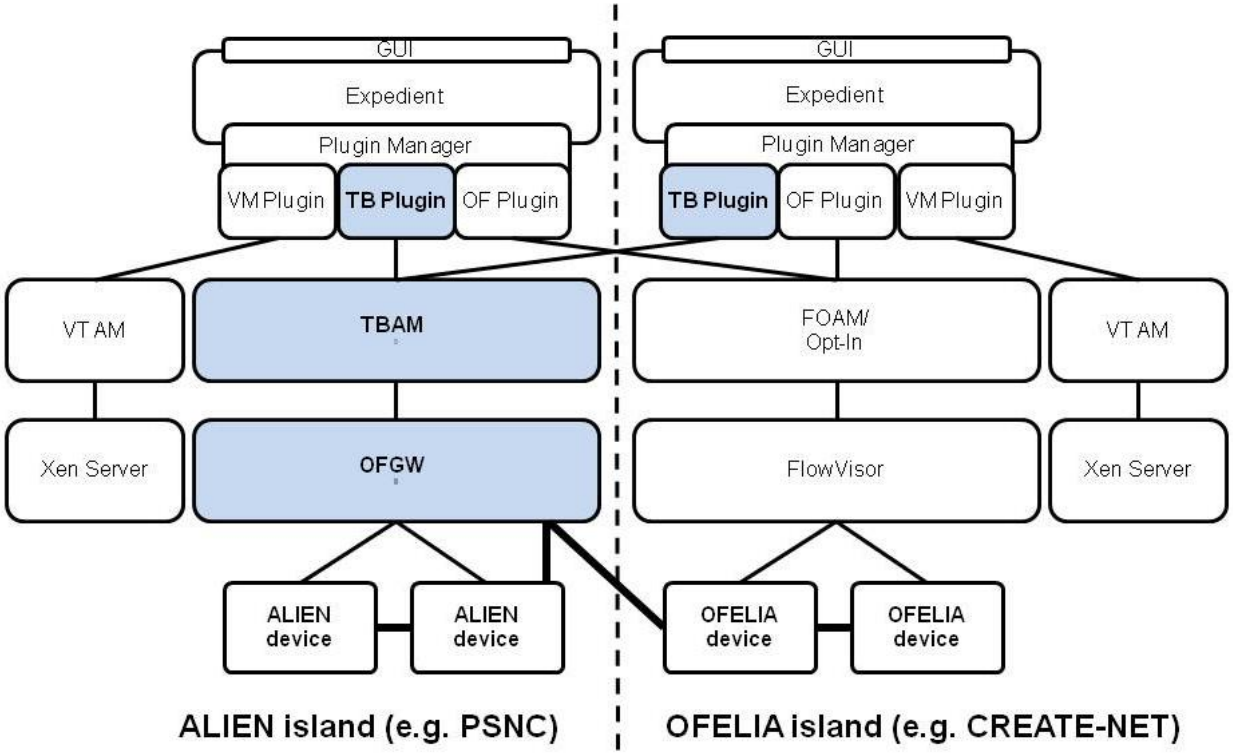


Figure 1.1: : ACF (left) and the classic OCF (right). Each VM Plugin can be connected to the VT AM of other islands. In the figure, the lines for these connections are omitted for the sake of clarity.

Figure 1.1 shows how an ALIEN island running the ACF (on the left side) is connected to the OFELIA facility. In the Figure the ALIEN island is connected to just one OFELIA island, but the diagram can be generalized by connecting the OF Plugin of the ACF to FOAM/Opt-In (the Aggregate Managers used in OFELIA) of other islands.

It is worth recalling that, once the user accesses the Expedient of an island (either OFELIA or ALIEN), he/she can include in the experiment only the aggregates that are connected to that Expedient instance through the management network. For this reason, the TB Plugin has to be added to the each Expedient (e.g. also in the CREATE-NET's OCF in Figure 1.1) to allow OFELIA users to add ALIEN resources to their experiments.

Regarding the data plane connections (bold lines between devices in Figure 1.1), we recall that the traffic exchanged between ALIEN and OFELIA island nodes must go through a tagging/untagging/retagging process (see Deliverables [D4.1, D4.2] and Section 4.3 of this document for more details). This operation is performed by the OFGW component of the ACF.

In the following Sections we will describe how to setup an experiment with the ACF (involving both ALIEN and OFELIA resources) and the role that each ACF's components play in the workflow. The description assumes that the user that creates the experiments is already registered to the OFELIA facility, as explained at the beginning of this Chapter.

The reservation of ALIEN resources is handled through TB Plugin, TBAM and OFGW as sketched in Figure 1.2 and summarized in the rest of this Chapter.

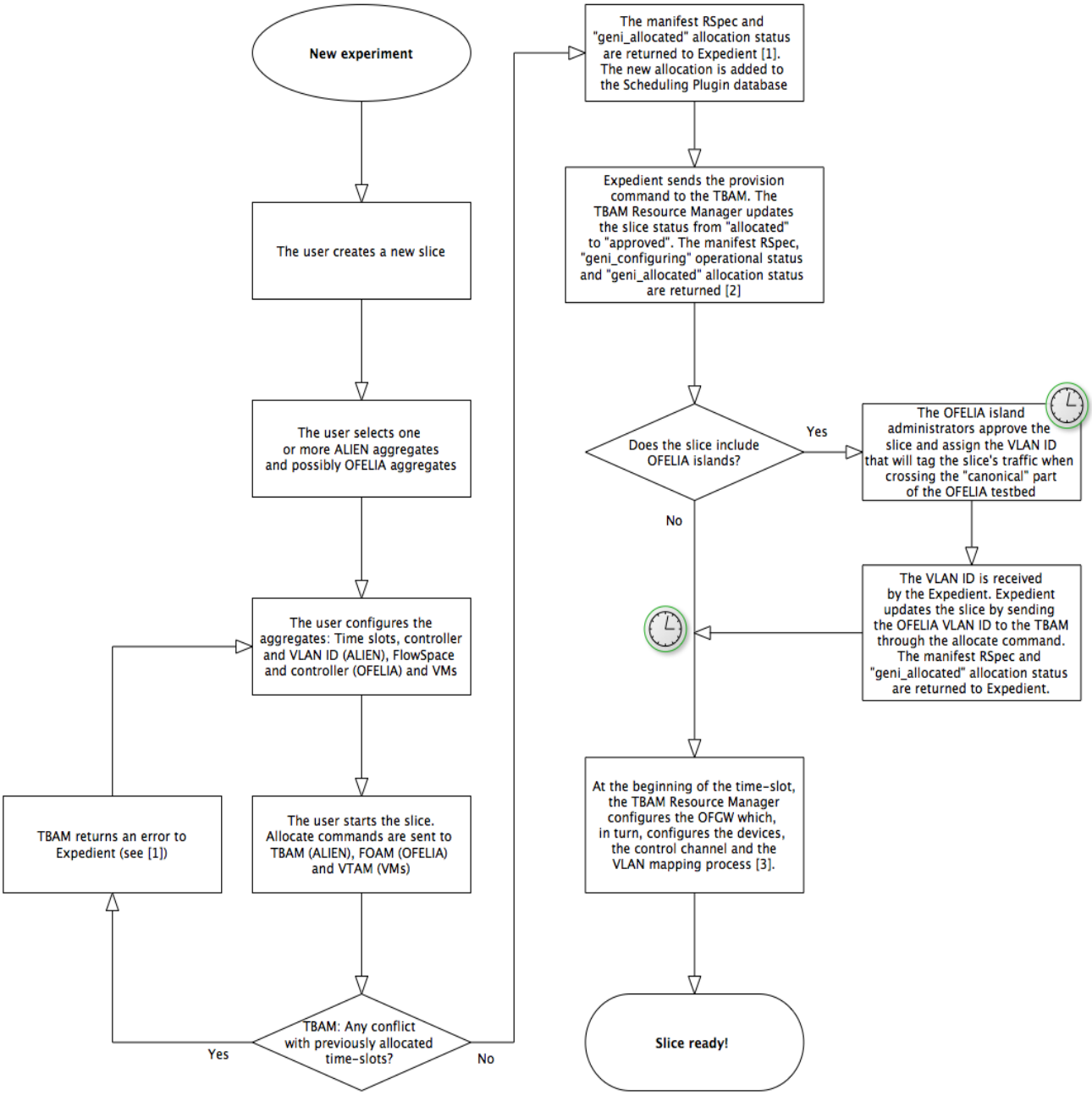


Figure 1.2: Instantiation workflow of a slice on an ALIEN island
[1] http://groups.geni.net/geni/wiki/GAPI_AM_API_V3#Allocate
[2] http://groups.geni.net/geni/wiki/GAPI_AM_API_V3#Provision
[3] If the OFELIA part of the slice has been not approved at the beginning of the time slot, the user's traffic will not be able to leave the ALIEN islands. Once the TBAM Resource Manager receives the OFELIA VLAN ID from Expedient, the VLAN mapping process will start.

1.1 A new slice setup

As a first step of the slice instantiation procedure, the user adds the aggregates to the slice as shown in Figure 1.3. The slice can include both OFELIA and ALIEN Aggregates plus one or more Virtualization Aggregates.

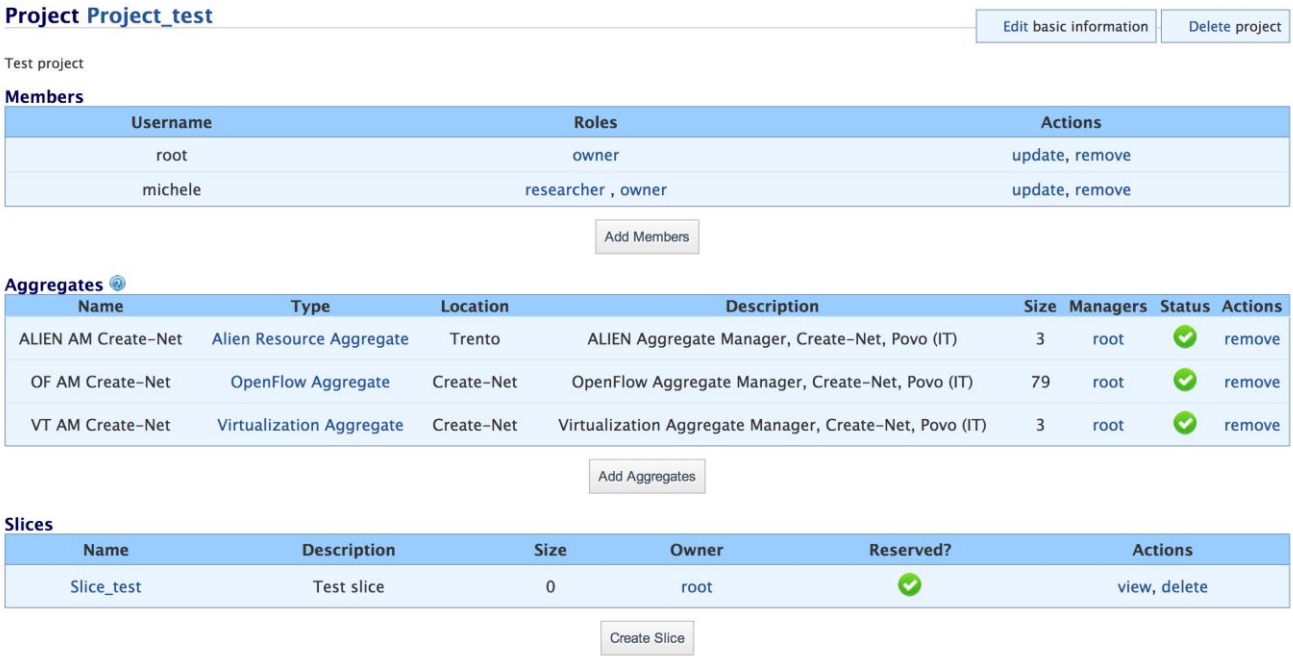
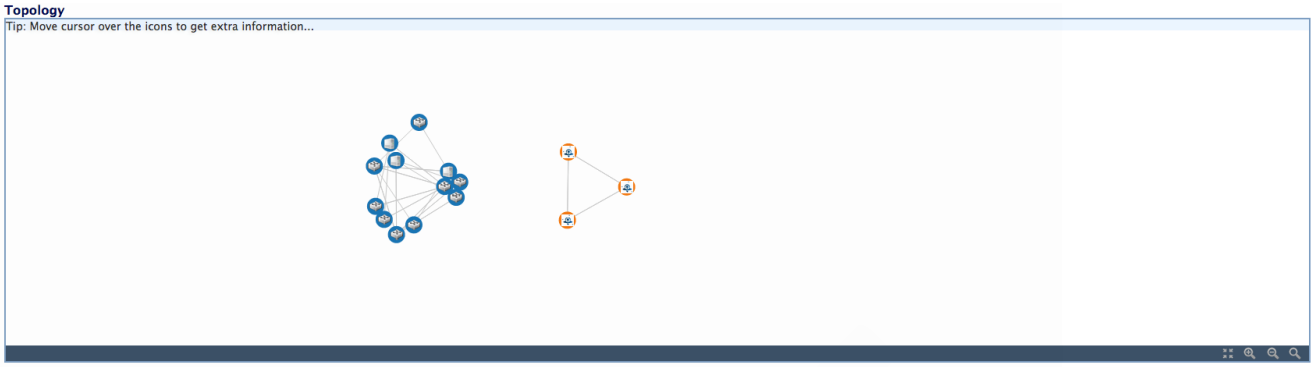


Figure 1.3: ALIEN Aggregate among OpenFlow and Virtualization aggregates

Once the slice is created, the network topology and the configuration parameters are shown on a dedicated section of Expedient (Figure 1.4). In particular, for the ALIEN aggregate reservation, we can find:

- start and stop time of the experiment on a calendar (see Figure 1.5);
- the controller IP Address and Port;
- the VLAN ID used to communicate with OFELIA. In particular this value refers to the ALIEN VLAN that should be mapped with the OFELIA one. The OFELIA VLAN is automatically assigned by Expedient (see deliverables [D4.1, D4.2] or Sections 4.3 of this document for further details about the VLAN remapping problem).

Final prototype of management software



Slice AMs and resource details

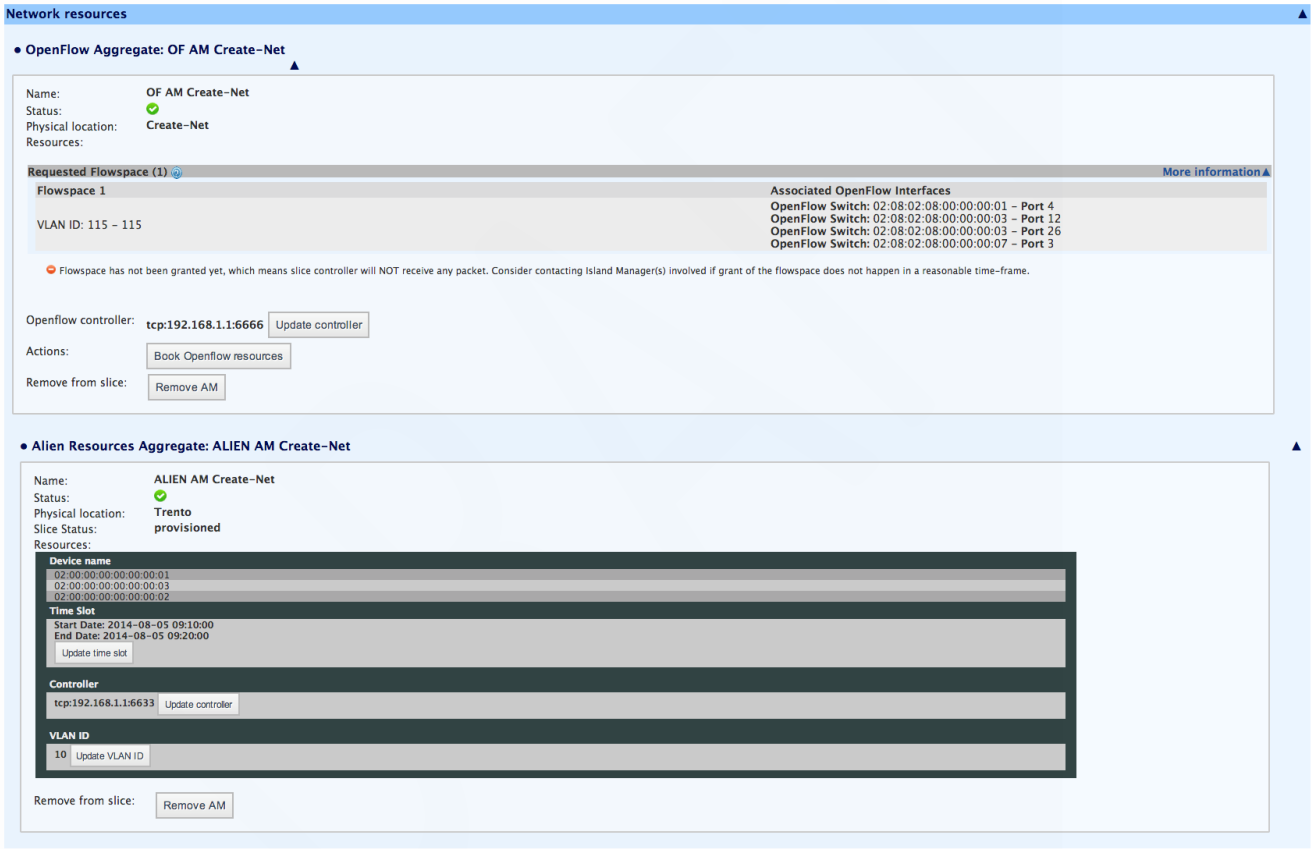


Figure 1.4: ALIEN Aggregate reservation

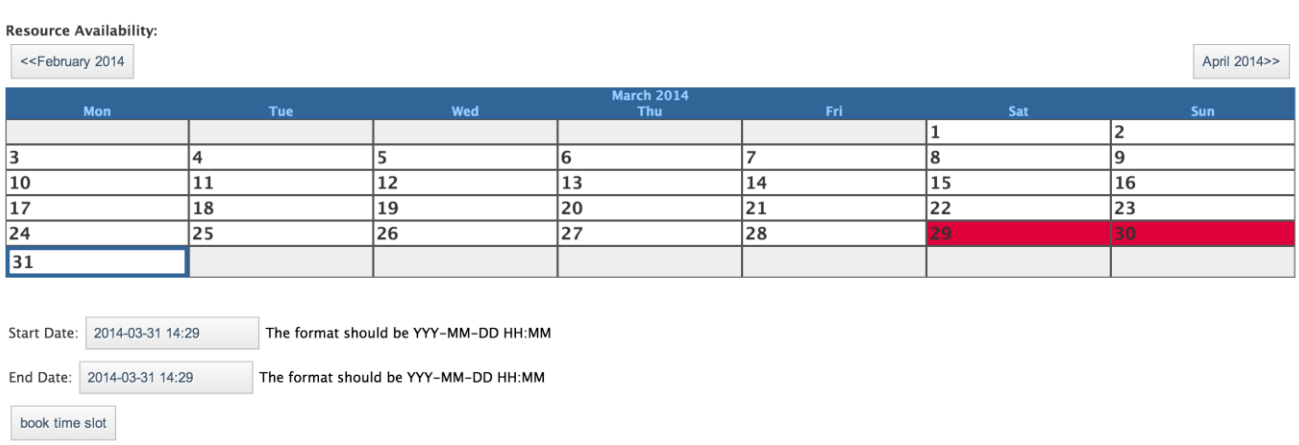


Figure 1.5: The GUI for the time-slot reservation. Already booked time-slots are highlighted in red.

When the user starts the slice, all the configurations are sent to:

- TBAM for ALIEN aggregates,
- FOAM for OFELIA aggregates
- VTAM for the Virtual Machines.

If no conflicts with previous time-slots are found by the TBAM, all the *GENI allocation* [GENIv3 API] information are returned to the TB Plugin as a confirmation message. At this point, the TB Plugin immediately confirms the reservation with a *GENI provision*.
When the OFELIA island administrators approve the slice, a VLAN ID is automatically assigned to the slice by the system (for a more details see section 4.3). Afterwards, Expedient transmits the OFELIA VLAN ID to the OFGW that is in-charge of performing the mapping and re-mapping process of the VLANs between ALIEN and OFELIA.

At the beginning of the time-slot, the TBAM RM sends the slice configuration to TBAM Agent running on the OFGW, which, in turn, configures the other sub-modules such as: Management, Control and Data planes, the OFGW's LDAP client etc.

Thanks to the exclusive access to the devices, the user is allowed to log in to the OFGW with limited privileges (the authentication is performed through the LDAP client which is connected to the central OFELIA LDAP server) and to use the Management Plane application (see Section 4.5) to get information such as: the flowtable content, the running switch configuration, the OpenFlow configuration of the switch etc.

2 Expedient UI extension: TB Plugin

The TB Plugin is a module that extends the Expedient GUI by adding management and control functions for the resources exposed by the TBAM (see Figure 2.1). The web form allows the user to configure an ALIEN aggregate i.e. book a specific subset of ALIEN resources for a fixed time period, modify the amount of booked resources and project's time schedule, set the ALIEN VLAN ID for connection with OFELIA island and configure the user's controller IP Address and Port.

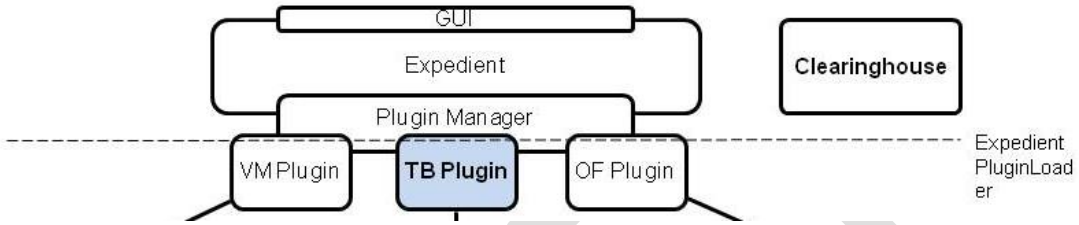


Figure 2.1: The TB Plugin in the ACF's architecture

The TB Plugin's source code is available at [TB Plugin code].

2.1 Installation of the TB Plugin

The procedure starts with the installation and configuration of Expedient followed by the installation of the plugin.

2.1.1 Requirements

```
GNU/Linux Debian-based distribution
Python 2.6 (available in Debian 6 release)
Root access to the system
```

2.1.2 Installation for Expedient

The following operations require root access to the Linux Operating system:

- Install MySQL server:

```
$ apt-get install mysql-server
```

- Create Expedient database:

```
$ mysql -p
$ mysql> CREATE DATABASE expedient;
$ mysql> GRANT ALL ON expedient.* TO user@127.0.0.1 IDENTIFIED BY password;
```

- Clone the OCF repository under folder /opt:

```
cd /opt
git clone https://github.com/fp7-ofelia/ocf.git ofelia
```

- In the file `/opt/ofelia/expedient/bin/versions/default/install/lib/pypelib` change the line:

```
/usr/bin/wget https://github.com/fp7-ofelia/pypelib/raw/deb/pypelib_latest_all.deb
|| error "Could not download pypel$
```

into:

```
/usr/bin/wget --no-check-certificate https://github.com/fp7-
ofelia/pypelib/raw/deb/pypelib_latest_all.deb || error "Could not download pypel
```

- Trigger OFVER installation by performing the following as a root user:

```
$ cd /opt/ofelia/expedient/bin
$ ./ofver install
```

  - When installation starts, ofver will ask if it is an OFELIA project installation or not. Select Yes (Y) for OFELIA testbeds

  - You will need to create the certificates for the Certification Authority (CA) first and for the component (i.e. Expedient) later. Do not use the same Common Name (CN) for both of them, and make sure that the CN you use in the component later certificate (you can use an IP) is the same you then set in the SITE_DOMAIN field in the localsettings.py file

  - Modify the `localsettings.py` or `mySettings.py` depending on the component (i.e. Expedient) being installed:

```
ROOT_USERNAME = "user"
ROOT_PASSWORD = "pass"
DATABASE_NAME = "expedient"
DATABASE_USER = "user"
DATABASE_PASSWORD = "password"
SITE_DOMAIN = "localhost:1234"
LDAP configuration (only needed in case of connection with the OFELIA
central LDAP server):
ENABLE_LDAP_BACKEND = True
LDAP_STORE_PROJECTS = True
ALLOW_LOCAL_REGISTRATION=False
AUTH_LDAP_BIND_PASSWORD = YOUR_LDAP_PASSWORD
AUTH_LDAP_BIND_DN = YOUR_LDAP_BIND_DN
```

- Run:

```
$ ./opt/ofelia/expedient/src/python/expedient/clearinghouse/manage.py
create_default_root
```

- Try locally to open `https://localhost/` and to login into expedient.

2.1.3 Installation for TB Plugin

- Clone the plugin folder:

```
$ git clone https://github.com/fp7-ALIEN/OCF-TBPlugin.git TB-plugin
```

- Copy the folder `TB-plugin/ALIEN_plugin under /opt/ofelia/expedient/src/python/plugins/`

- Copy the sa and ma cert files from the folder `ALIEN_plugin/certs` to the folder `deploy/trusted` of TBAM

- Synchronize database:

```
$ cd /opt/ofelia/expedient/src/python/expedient/clearinghouse
$ python manage.py syncdb
```

- Restart Apache

After that, an aggregate manager with the type "ALIEN Resource Aggregate" can be added and configured from the Administration tab of the Expedient (Figure 2.2 and Figure 2.3).

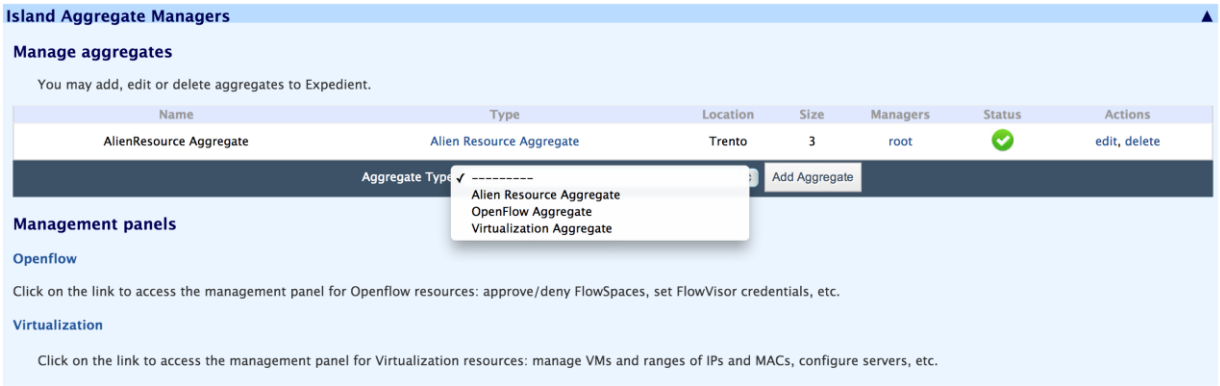


Figure 2.2: Add an ALIEN Resource Aggregate

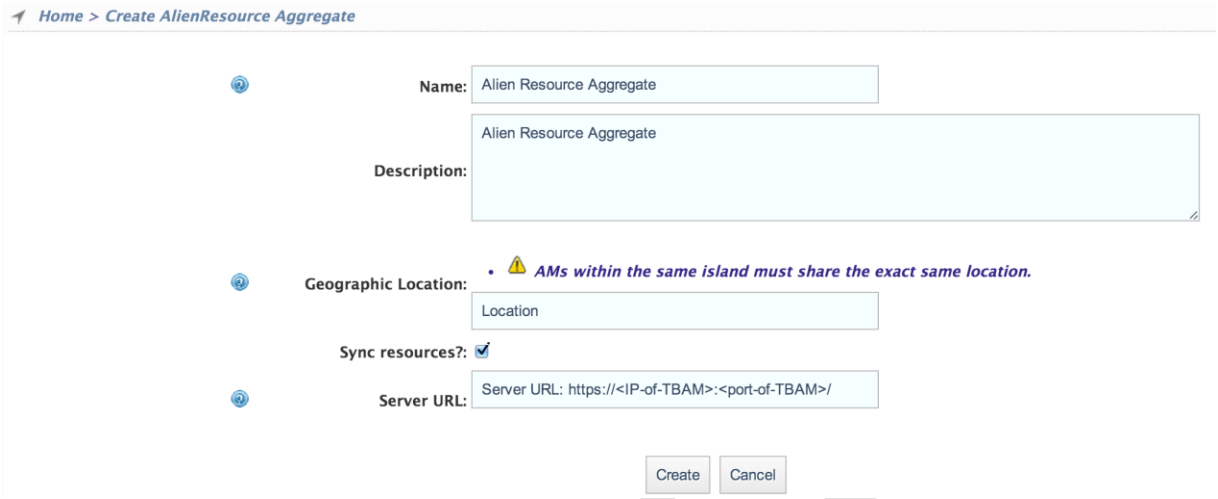


Figure 2.3: Creation of a new ALIEN Resource Aggregate

Then, the new ALIEN aggregate can be added to slices, and user can enter ALIEN slice details and request to allocate the slice in the TBAM as explained in Section 1.1.

After TBAM allocates user slice successfully, user can modify slice details i.e. allocated time slot, controller, ALIEN VLAN through TB Plugin. After any update of slice details, TB Plugin notifies the user that the slice needs to be restarted/updated by pressing the button “update slice” in order for the update to take place (Figure 2.3). By pressing this button, the new configurations will be sent to TBAM, and new time slot can be allocated if required and if there is no conflict between the new slot and other reserved slots (conflicts are highlighted with error messages on the GUI).

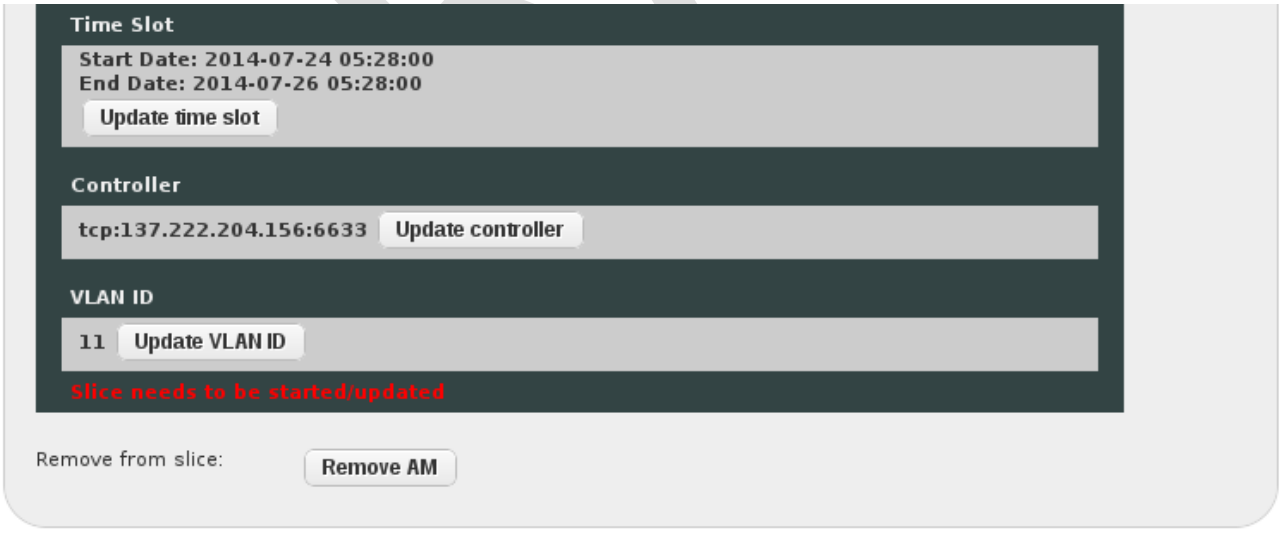


Figure 2.3: User notification of slice update

3 Time-Based Aggregate Manager (TBAM)

The TBAM is developed using the AMsoil framework [AMsoil] and the ALIEN-specific functionalities are provided through three different plug-ins.

In particular, the TBAM integrates a calendar-like aggregate manager (TBAM Scheduling Plugin), a resource manager which performs the actual provisioning of the slices (the TBAM RM) and a module that translates the information coming from Expedient for the TBAM RM and vice-versa (the TBAM Delegate). The TBAM is connected to ALIEN resources via the OpenFlow Gateway (OFGW) and to the Expedient via the Time Based plugin (TB Plugin) (see Figure 3.1).

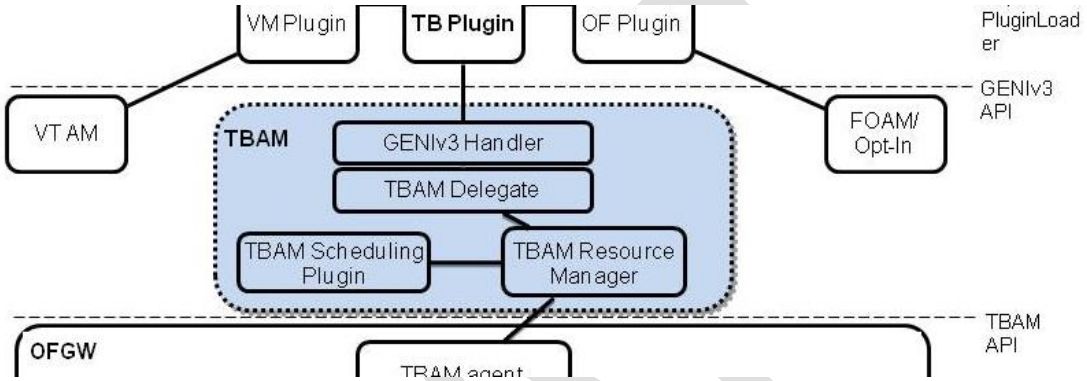


Figure 3.1: The TBAM in the ACF's architecture

The source code of the TBAM (and all the sub-modules described in this Chapter) is available at [TBAM code].

3.1 TBAM Delegate

The Delegate sits between the GENIv3 Handler and the TBAM RM. It translates the GENIv3 call to the domain-specific call of the TBAM RM and, in particular the Resource Specification [RSpec] into TBAM RM values (and vice-versa). Moreover, it handles the TBAM RM's exceptions and re-throws them to Expedient via the the GENIv3 method API.

3.1.1 Interfaces

TBAM Delegate methods corresponding to the GENIv3 API are:

- `list_resources(client_cert, credentials, geni_available)`: retrieves the information of the network devices and the reserved time-slots through the TBAM RM. It returns an advertisement RSpec, e.g.:

```
<rspec aggregate="http://example.com/aggregate" type="advertisement">
   <aggregate:resources xmlns:aggregate="http://example.com/aggregate">
     <aggregate:switch dpid="00:00:00:00:00:00:00:00"/>
     <aggregate:switch dpid="00:00:00:00:00:00:01"/>
     <aggregate:link dpidDst="00:00:00:00:00:00:01" dpidSrc="00:00:00:00:00:00:00"
         portDst="1" portSrc="5"/>
     <aggregate:link dpidDst="00:00:00:00:00:00:01" dpidSrc="00:00:00:00:00:00:00"
         portDst="7" portSrc="6"/>
     <aggregate:reservation end_time="20/02/2014 15:42:30"
         slice_urn="urn:publicid:IDN+geni:gpo:gcf+slice+testing"
start_time="20/02/2014
         15:42:00"/>
   </aggregate:resources>
</rspec>
```

The Delegate can throw a SERVERERROR GENIv3 exception in case of connection problems between the TBAM RM and the TBAM Agent.

- `allocate(slice_urn, client_cert, credentials, rspec, end_time=None)`: translates the RSpec of the allocation to a TBAM RM's `reserve_aggregate` request. An example of the RSpec is:

```
<rspec type="request"
         xmlns="http://www.geni.net/resources/rspec/3"
         xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
         xmlns:aggregate="http://example.com/aggregate"
         xs:schemaLocation="http://www.geni.net/resources/rspec/3
http://www.geni.net/resources/rspec/3/ad.xsd http://example.com/dhcp/req.xsd">
        <aggregate:slice  slice_urn="urn:publicid:IDN+geni:gpo:gcf+slice+testing"/>
        <aggregate:timeslot  start_time="20/2/2014  15:45:0"  end_time="20/2/2014
15:47:00"/>
        <aggregate:VLAN ALIEN="0xffff" OFELIA="10"/>
        <aggregate:VLAN ALIEN="30" OFELIA="40"/>
        <aggregate:controller url="192.168.1.2:6633" />
        <aggregate:project        projectID="612cca94-1a7a-4431-87d0-8572399385aa"
projectName="Project_test" />
</rspec>
```

If the reservation is allocated, the TBAM Delegate returns to the GENIv3 Handler both the `geni_allocated` status and the RSpec containing the information of the allocated resource. If the allocation fails, the Delegate raises one of these GENIv3 standard errors:

- BADARGS: indicates a malformed RSpec;
- ERROR: the database contains at least two entries with the same `slice_urn`. This error should never happen, because the Resource Manager is in-charge of controlling the uniqueness of the `slice_urn`;
- ALREADYEXISTS: the error is triggered by either an already used `slice_urn` or an overbooking of the time-slot;
- SERVERERROR: the communication between TBAM Resource Manager and TBAM Agent has encountered a problem.

- `provision(urns, client_cert, credentials, best_effort, end_time, geni_users)`: translates the provision to the TBAM RM's `approve_aggregate` request. If the request is approved, the Delegate returns the `geni_allocated` status, the `geni_configuring` operational status and the RSpec equal the one of the *allocate* method. In case of errors the method will raise:

  - BADARGS: indicates a malformed RSpec;
  - ERROR: the database contains at least two entries with the same `slice_urn`. This error should never happen, because the Resource Manager is in-charge of controlling the uniqueness of the `slice_urn`;
  - SEARCHFAILED: a previous allocation having the same `slice_urn` is not found;
  - ALREADYEXISTS: another allocation or approval entry exists with same `slice_urn`.

- `delete(self, urns, client_cert, credentials, best_effort)`: translates the *delete* to the Resource Manager `delete_aggregate`. It returns a `geni_unallocated` status or it throws:

  - BADARGS, ERROR, ALREADYEXISTS: equals to the *provision* errors;
  - SEARCHFAILED: a previous allocation or approval having the same `slice_urn` is not found.

3.2 TBAM Resource Manager (TBAM RM)

The Resource Manager is a sub-module of the Time-Based Aggregate Manager and is provided as a plugin for AMsoil [AMsoil]. The Resource Manager is in charge of configuring the ALIEN aggregate and of ensuring that the aggregate is accessed by only users assigned to the ongoing experiment.

The TBAM RM implements domain-specific methods to manage the resources (see also Resource Manager in the AMsoil architecture [AMsoil RM]). In particular the northbound interface of TBAM RM provides a set of methods that the TBAM Delegate uses to interact with the TBAM RM and to perform operations like the allocation and provisioning of the resources. On the other side, the southbound interface of the TBAM RM guarantees the booking of resources (through the Scheduling Plugin [TBAM Scheduling Plugin]) and the configuration of the parameters required by the user (through the OFGW TBAM Agent).

Finally, the TBAM RM checks the start or expiration of the experiments through a mechanism which is based on the AMsoil worker services [AMsoil worker].

3.2.1 Northbound interface: TBAM Delegate <--> TBAM RM

The RM operations are divided into two stages. First, the temporary booking of an ALIEN aggregate for a given time-slot is achieved through an allocate call. Nothing is configured on the ALIEN devices, the TBAM RM just updates the database of the Scheduling Plugin. After that, the booked time-slot is automatically approved by Expedient through a provision call. Also in this stage the only operation performed by the RM is updating the database entry status from "allocated" to "approved". The provision is not approved if the time-slot is different from the one provided during the allocation. The real provisioning of the slice is performed automatically by the TBAM RM by sending the configuration parameters to the OpenFlow Gateway (OFGW) at the beginning of the time-slot. The allocate call is also used to update already approved or provisioned slices (e.g. to change the controller information or to extend/reduce the time-slot). This is achieved thanks to a parameter that identifies the slice (the `slice_urn`).

The Table 3.1 lists the APIs exposed by the Resource Manager on the northbound interface. The table is organized as follows: on the leftmost column are listed the GENIv3 API methods that the TBAM delegate exposes to the TB Plugin for Expedient. On the rightmost column, the corresponding TBAM RM methods are provided.

GENIv3 API	Corresponding TBAM RM methods
`listResources`	`getSws()`: returns a list of DPIDs of the switches `getLinks()`: returns a list of links (e.g. "`DPID:port-DPID:port`") `getAvailability()`: returns all reserved `time-slot (start_time=datetime, end_time=datetime, slice_urn)` `checkAvailability(start_time, end_time)`: returns true if the time-slot is available
`allocate`	`reserve_aggregate(slice_urn, owner_uuid, owner_mail, start_time, end_time, VLANs, Controller, projectInfo)`: controls the availability and processes the temporary booking of the entire ALIEN aggregate in the provided time-slot. Also the provided parameters are saved in the database thanks to the TBAM Scheduling plugin: • `projectInfo`: contains the project information (project id and name) that will allow the user to access the OFGW (detailed in section 0) • `VLANs`: permits the VLAN tag rewriting for the network traffic exchanged between OFELIA and ALIEN islands. It is based on a `python dict`, for instance `{"10" : "0xffff", "30" : "20"}` OFELIA VLAN 10 is set as untagged within ALIEN and OFELIA VLAN 30 is rewritten to the ALIEN VLAN 20 • `controller`: contains the user's OpenFlow controller IP address and port as a Pyhton string `"192.168.100.1:6633"` The allocate method provides also the update of an approved or provisioned entry. In particular when an allocate call with same `slice_urn` is received, all the parameters are updated. If the approved entry is started, the update process will involved also the TBAM Agent for install the new configurations
`provision`	`approve_aggregate(slice_urn)`: The provision is approved if there is previous allocation with same slice_urn.
`delete`	`delete_aggregate(slice_urn)`: releases an allocated or approved slice identified by slice_urn by deleting the entry in the Scheduling Plugin database. If the slice has already been started, the Resource Manager tells the TBAM Agent to restore the resources and its internal modules to the default configuration.

Table 3.1: Northbound interface of the TBAM Resource Manager

3.2.1.1 Errors

- `IslandRMRPCError`: the connection from TBAM RM and TBAM Agent returns an error. All aforementioned TBAM RM methods can raise this error;
- `IslandRMMoreSliceWithSameID`: can be raised by `reserve_aggregate` and `approve_aggregate` and the error means that the TBAM Scheduling Plugin contains at least two slices having the same `slice_urn`.
- `IslandRMNotUnivocal:` the TBAM RM have found a conflict with an already existing `slice_urn` during the `reserve_aggregate`;
- `IslandRMNotAllocated: delete_aggregate` and `approve_aggregate` are called for a slice not already allocated or approved;
- `IslandRMAlreadyReserved`: it is used by the `reserve_aggregate` when the TBAM Scheduling Plugin finds an overbooking error.

3.2.2 Southbound interface: TBAM RM <--> OFGW TBAM Agent

The TBAM Agent is an internal OFGW sub-module that is invoked by the TBAM RM to provision/reset the resources. In addition the TBAM Agent permits to retrieve the information of the network devices and to access the forwarding plane through some pre-defined functionalities exposed by the Management plane. The communication between the TBAM Resource Manager and TBAM Agent is achieved through a SecureXMLRPC protocol.

TBAM Agent exposed methods	Functionality
`getSws`	returns the DPIDs of the switches retrieved through MGMT interface
`getLinks`	returns the links between switches retrieved through MGMT interface
`getAvailability`	returns already reserved slices time-slot and urn.
`set/remUserAuth(projectInfo)`	manages the user's credentials needed by the user to access the OFGW
`set/remTCPProxy (controller)`	*configures the TCP Proxy daemon with the user's controllers coordinates*
`set/remTCPProxy (controller)`	*configures the VLAN mapping between Alien and OFELIA forwarding planes*

Table 3.2: The Southbound interface of the TBAM Resource Manager

3.3 TBAM Scheduling Plugin

The TBAM RM leverages on the TBAM Scheduling Plugin [TBAM Scheduling Plugin] to record the allocated/provisioned time-slots and to avoid conflicts among different experiments. The TBAM Scheduling Plugin is an extended version of the AMsoil's Scheduling Plugin [AMsoil Schedule] with the following additional APIs that are used to access the internal database:

- flag `allocate/approved.` Indicates if the slice in the database has been already approved or just allocated. Allocated slices expire after a pre-determined time (set by default to 48 hours).
- flag `started/not started`. Indicates whether an approved slice is currently started or not (in other words if it is provisioned or not).

Other parameters, that do not require any modification in the existing API, exploit the RSpec of the Scheduling Plugin:

- `projectInfo`
- `controller IP and port`
- `VLANs Mapping`

3.4 TBAM installation

AMSoil requires Python version 2.7 [AMsoil installation] and the `web.py` module for Python, therefore a Debian 7 Linux OS is recommended. However, during the testing phase we were able to install both Expedient and the TBAM on the same Debian 6 machine (required by Expedient) by adding the version 2.7 of Python to the system.

The repository [TBAM code] contains the AMsoil code tree with the three modules discussed in this Chapter already installed in the *AMsoil/src/plugins* (see [TBAM Delegate], [TBAM RM] and [TBAM Scheduling Plugin]).

The TBAM can be started from the command line (no *root* privileges are needed) by running `python AMsoil/src/main.py` and then `AMsoil/src/main.py --worker` to start the worker server.

ADDITIONAL NOTE:

The southbound interface of the TBAM Resource Manager allows the communication with the OFGW through the TBAM Agent (described in Chapter 4). However, for testing purposes, this interface can be disabled and Resource Manager will use fake fixed data that will be returned to the TBAM Delegate when requested (e.g. with the `listResources` command). The connection to the TBAM Agent can be enabled/disabled by changing the *CONN_WITH_AGENT* parameter to *True/False* in the `src/plugins/islandRM/islandresourcemanager.py` file. Obviously, the activation of this connection requires the installation and configuration of the OFGW as detailed in Chapter 4.

4 The OpenFlow Gateway (OFGW)

The OFGW is the resource manager for the ALIEN devices and represents the only entry point for accessing the ALIEN Island's network resources. It represents a single access point for the management, control and data planes. It is deployed as a Debian 6 Linux machine (real or virtual) and is configured/controlled by the TBAM Agent daemon.

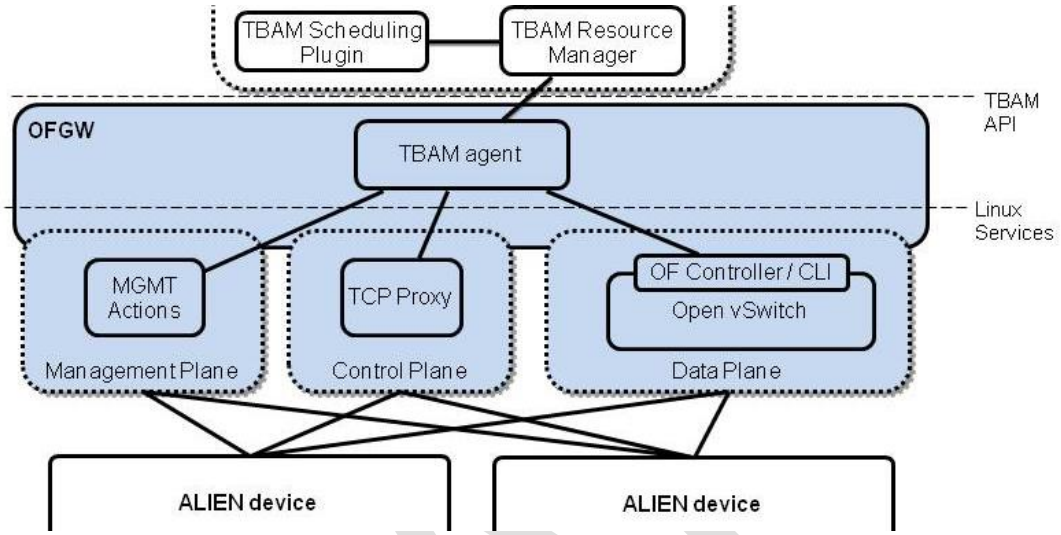


Figure 4.1: The OFGW in the ACF's architecture

As shown in Figure 4.1, the OFGW is composed of four different modules: the TBAM Agent, the Management Plane, the Control Plane and the Data Plane. The installation and configuration procedures of each of these modules are described in the following sections.

4.1 The TBAM Agent

The TBAM Agent is a daemon running on the OFGW and is responsible for the communication with the TBAM RM and for the configuration of Management Plane, Control Plane and Data Plane sub-modules plus the LDAP client of the operating system.

The TBAM Agent's source code is available at [TBAM Agent code].

4.1.1 Installation

The communication with the TBAM RM is obtained with a SecureXMLRPC interface. For testing purposes, the RPC server in the TBAM Agent and the TBAM RM use pre-installed certificates. In real deployments, the ClearingHouse (see Section 5) can be used to generate the certificates. The TBAM Agent uses three certificates for the SecureXMLRPC authentication; moreover, the references contained in the code must be compliant with your local path. In particular, the parameters of the certificates used in the TBAM Agent are presented in Table 4.1.

Parameter	Reference
SERVER_KEY_PATH	it refers to the server key.
SERVER_CERT_PATH	it refers to the server cert.
TRUSTED_CERT_PATH	it refers to trusted certificate issued by Clearinghouse.

Table 4.1: The parameters linked to the certificates

On the other hand, in order to communicate with the TBAM Agent, the TBAM RM must use the correct client certificates defined by parameters `CLIENT_KEY_PATH` and `CLIENT_CERT_PATH` within the TBAM RM's code [TBAM RM code].

The TBAM Agent can be started by entering the folder `OCF-OFGW/TBAM-Agent` and by `running python src/main.py`. The TBAM Agent has been tested with Python version 2.7 and with Debian-based operating system.

To connect the Resource Manager to the TBAM Agent, the two variables `OFGW_ADDRESS` and `OFGW_PORT` in the TBAM Resource Manager's `islandrecurcemaneger.py` must be updated with the IP address and TCP port of the SecureXMLRPC server of the TBAM Agent.

By default, the configuration considers that both TBAM Agent and Resource Manager reside on the same machine `(OFGW_ADDRESS = "127.0.0.1")` and the TCP port used for the communication is 8234 `(OFGW_PORT = 8234).`

ADDITIONAL NOTE:

The TBAM-Agent uses the Management Plane module to retrieve network details such as DataPath IDs (DPIDs) of the devices and links. For testing purposes, this connection is disabled by default and the TBAM-Agent is configured to return fake information. The connection to the Management Plane module can be enabled/disabled by changing the `CONN_WITH_MGMT` parameter to `True/False` in the `src/main.py`. Obviously, the activation of this connection requires the Management Plane to be installed, configured and started (see Section 4.5).

4.2 User's authentication

On the OFGW, both users/experimenters and administrators can use the Management interface of the devices to perform custom configurations or to retrieve the current status. The access is guaranteed by the local LDAP client which is connected to the central OFELIA LDAP server to allow the users SSH access using the OFELIA credentials.

While the island administrator has always access to the OFGW, normal users are allowed to login only during their experiment time-slot. The LDAP client is configured by the TBAM Agent which, in turn, receives the experiments details (e.g. project id, time-slot, etc.) from the TBAM RM. With these details, the TBAM Agent re-configures the LDAP client at the beginning of each time-slot so that only the authorized users are enabled to access the OFGW.

4.2.1 LDAP client configuration

To connect the OFGW's LDAP client to the central OFELIA LDAP server, the following steps are required:

1. DNS bind between LDAP url and its IP address (add the line `LDAP_IP_address LDAP_url` to `/etc/hosts`);

2. Install package `ibpam-ldapd` (e.g, `apt-get install ibpam-ldapd`) and follow the procedure on the UI to configure the package. When required:

    a. Insert the LDAP url (`ldap://ldap.ibbt.fp7-ofelia.eu`)
    b. Set the correct DN (`dc=fp7-ofelia,dc=eu`)
    c. Select not to root management
    d. Authorize the modification of `nsswitch.conf`: check that `passwd,group,netgroup` are selected.

3. Add auth required `pam_access.so` in `/etc/pam.d/common-auth`

4. Add session required `pam_mkhomedir.so skel=/etc/skel umask=0022` in `/etc/pam.d/common-session`

5. Create file `/etc/security/access.conf` (e.g., `touch /etc/security/access.conf`) and add this three line:

```
+ : ALL : LOCAL
+ : @proj_<project_UUID>_<project_name> : ALL
EXCEPT root login:ALL EXCEPT LOCAL
```

After the configuration, the root user is able to login through a SSH connection. Other users can authenticate only if both TBAM Agent and TBAM RM are up and running and the users are associated to the current experiment.

4.2.2 Configuration interface

The TBAM Agent receives the LDAP configuration parameters through the following interface:

`Set/remUserAuth(projectInfo)`: respectively write or clear the `projectInfo` in the `/etc/access.conf` file. The `projectInfo` containts `project_UUID` and `project_name` and allows to query the LDAP server to authenticate only the permitted users.

4.3 Data Plane

The aim of the Data Plane module is to guarantee the connection between ALIEN and OFELIA. In particular the Data Plane provides the translation between the ALIEN VLANs and the OFELIA VLANs.

While in the OFELIA islands VLAN tags are used to isolate the traffic among the different slices, in the ALIEN islands the user can freely tag the traffic with one or more VLAN tags (the VLAN tags are passed to the TBAM Agent through the

Expedient UI). As a consequence, an additional process (provided by the Data Plane module) is needed to swap of the ALIEN VLAN id to the OFELIA VLAN id and vice versa.

The re-tagging mechanism is achieved through an Open vSwitch [OvS] instance configured by the TBAM Agent. In particular, static flow entries are configured in the OvS to achieve correct VLANs tagging and untagging process.

The Data plane requires OvS version higher than 2.0.0. The installation procedure is as follows:

```
sudo apt-get install build-essential fakeroot
wget http://openvswitch.org/releases/openvswitch-2.1.2.tar.gz
tar -zvxf openvswitch-2.1.2.tar.gz && cd openvswitch-2.1.2
fakeroot debian/rules binary
cd ../ && dpkg –i openvswitch-common openvswitch-datapath-dkms openvswitch-
datapath-source openvswitch-pki openvswitch-switch
```

The initial OvS configuration process requires running the following commands as root:

```
ovs-vsctl add-br switch -- set Bridge switch fail-mode=secure
ovs-ofctl add-flow switch "priority=0, actions=drop"
```

The port configuration considers two ports: one connected to OFELIA (OFELIAPort) and the other to an ALIEN device (ALIENPort). The port configuration requires these three commands:

```
ovs-vsctl add-port switch ALIENPort -- set Interface ALIENPort ofport_request=2
ovs-ofctl mod-port switch 2 up
ovs-vsctl add-port switch OFELIAPort -- set Interface OFELIAPort ofport_request=3
ovs-ofctl mod-port switch 3 up
```

4.3.1 Configuration interface

The Data Plane module receives the configuration parameters through the following interface:

`setOvs(VLANs)` and `remOvs(VLANs)` respectively add or remove the flows entry in the OpenvSwitch for the VLANs mapping. The VLANs parameter is a `python dict: {OFELIA-VLAN : ALIEN-VLAN ...}`; for instance `{"10" : "0xffff", "30" : "20"}` means that OFELIA VLAN 10 is set as untagged within ALIEN and OFELIA VLAN 30 is rewritten to the ALIEN VLAN 20.

4.4 Control Plane

The OFGW acts as TCP Proxy and forwards the control traffic from the devices to the OpenFlow user's controller and vice-versa. The TCP Proxy is implemented as an `iptables` [iptables] configuration handled by the TBAM Agent.

Additionally, the TBAM Agent takes care of the interruption of the connections after the expiration of a time-slot. This functionality is implemented through the iptables's `conntrack` [conntrack] module so that only the TCP proxy is reset without affecting the rest of the OFGW's firewall operations.

The Control Plane requires the `iptables` version equal or higher to 1.4.14 and `conntrack` packages. The installation procedure is the following (root previleges are needed):

- conntrack:

```
apt-get install conntrack
```

- iptables

```
git clone git://git.netfilter.org/iptables.git
git checkout -b 1.4.20 remotes/origin/stable-1.4.20
./autogen.sh && ./configure && make
make install
cd /sbin
ln -s /usr/local/sbin/iptables iptables
sysctl net.ipv4.ip_forward=1
```

4.4.1 Configuration interface

The Control Plane module is configured by the TBAM Agent through the following interface:

`setTCPProxy(controller)` and `remTCPProxy(controller)` respectively activate and disable the forwarding of the control traffic to and from the user's controller. The controller parameter is a `python string: "ip:port"`; for instance `"192.168.1.1:6633"`.

4.5 Management Plane

The Management Plane module provides an access to a device for the island administrators and users. It is used to perform some initial configuration of a device (e.g. VLANs, port configuration, etc.) as well as the configuration of the OpenFlow protocol. The TBAM agent leverages the Management Plane module to return island's topology details to the Expedient UI (e.g. datapath_ids, available ports, links, etc.).

In particular OFGW Management Plane module is responsible for provisioning the management tasks on hardware. It allows for secure control the device by the OFELIA experiment user. It also provides a set of tools for the OFGW administrator and the RESTful interface for the TBAM Agent.

The OFGW Management Plane consists of a Command Line Interface (CLI) for both OFELIA users and OFGW administrator, an inventory database, plugins for each hardware type and a RESTful interface.

The MGMT Actions's source code is available at [MGMT Actions code].

4.5.1 Command Line Interface

The OFGW Management CLI allows the remote control of the devices as well as getting status reports. The access to the specific commands is restricted by the user role: OFGW administrator or OFELIA experiment user. The table below presents OFGW CLI commands.

Command	Description
`list`	*list all devices*
`list --status`	*list all devices and gets the device status (UP/DOWN/PING time)*
`reboot [DEVICE_ID]`	*reboots the device*
`fact-reset`	*resets the device and restore the factory settings*
`show`	*shows the device configuration*
`show --ports`	*shows port status*
`show --of`	*shows OpenFlow configuration*
`show --tables`	*shows OpenFlow tables dump*
`show --neighbours`	*shows device's neighbours*

Table 4.2: The Command Line Interface for the user

Command	Description
`list`	*list all devices*
`list --status`	*list all devices and gets the device status (UP/DOWN/PING time)*
`reboot [DEVICE_ID]`	*reboots the device*
`show`	*shows the device configuration*

`show --ports`	*shows port status*
`show --neighbours`	*shows device's neighbours*
`fact-reset`	*resets the device and restore the factory settings*
`admin config`	*shows the OFGW Management configuration*
`admin users`	*shows OFELIA users and status*

Table 4.3: The Command Line Interface for the OFGW administrator

4.5.2 Inventory database

The inventory database contains configuration files that specify the hardware groups and specific device management interface parameters. The parameters are used by the hardware group plugin to control and force the user commands. The below snippets present the example configuration files.

Hardware groups

The groups configuration file specifies common parameters for inventory configuration file that meets the requirements for managing a specific hardware. These parameters are used by the hardware plugin and scopes on values like: SSH IP address/port, username, etc. Below an example of `groups.yaml` configuration file is presented.

```
groups.yaml
groups:
  - name: group1
    required_params:
      - id
      - host
      - login
    optional_params:
      - ofctrl

  - name: group2
    required_params:
      - id
      - host
      - login
    optional_params:
      - ofctrl
```

Inventory

The inventory configuration file contains entries of the hardware inventory under the OFGW management. Each hardware element should be placed in a new row and under proper group. Comments are allowed (started with # character). Below an example of `inventory.yaml` configuration file is presented.

```
inventory.conf
```

```
[group1]
id=ID1 host=127.0.0.1 ofctrl=127.0.0.1:5000 login=alice
id=ID2 host=127.0.0.2 login=bob

[group2]
id=ID3 host=127.0.0.3 login=alice
```

4.5.3 RESTful interface

The RESTful interface allows the OCF to inform on the current device status through the TBAM Agent. The RESTful interface specification is presented below.

Neighbours configuration file

The neighbours configuration file contains a static database that describes the connections between the ALIEN devices. This file is needed for feeding upper layers by the TBAM Agent. Below an example of `neighbours.yam`/ configuration file is presented.

```
neighbours.conf

ID1:
  dpid: 02-00-00-00-00-00-00-01
  ports:
    1: 02-00-00-00-00-00-00-02
    3: 02-00-00-00-00-00-00-03
    7: 02-00-00-00-00-00-00-04

ID2:
  dpid: 02-00-00-00-00-00-00-02
  ports:
    1: 02-00-00-00-00-00-00-01
    3: 02-00-00-00-00-00-00-03
    7: 02-00-00-00-00-00-00-04
```

Available REST methods

Method	Url	Response	Description
GET	/api/help	json	Available REST methods
GET	/of-table	json	The OpenFlow table (all devices)

GET	/of-table/{DPID}	json	The OpenFlow table (of specified DPID)
GET	/of-table-raw	json	The OpenFLow table staus in raw POX controller format
GET	*/hosts*	*json*	*List of devices under OCF control*
GET	*/port-status/id/{ID}*	*json*	*Device's port status (of specified ID)*
GET	*/port-status/dpid/{DPID}*	*json*	*Device's port status (of specified DPID)*
GET	*/neighbours*	*json*	*Get all devices' neighbours*
GET	*/neighbours/dpid/{DPID}*	*json*	*Device's neighbours (of specified DPID)*
GET	*/neighbours/id/{ID}*	*json*	*Device's neighbours (of specified ID)*

Table 4.4: The RESTful interface for the TBAM Agent

4.5.4 Installation of the Management Plane software

4.5.4.1 Requirements

The following Python modules must be installed:

- `argcomplete`
- `flask`
- `pyyaml`
- `fabric`
- `python-daemon`
- `texttable`

The Python dependencies can be installed by the command:

```
$ pip install [name]
```

The project includes *PYTHON_REQUIREMENTS* file that contains a list of tested and working versions of the required Python modules. Command to install the suggested versions of the Python requirements is as follows:

```
$ pip install -r PYTHON_REQUIREMENTS
```

4.5.4.2 Configuration

The CLI needs the configuration files (`groups.yaml, inventory.conf`) in the program's directory to be placed. The source code of the Management Plane includes the example set of configuration files. Please copy the `*.example` files with the needed filenames as below.

```
$ cp inventory.conf.example inventory.conf
$ cp groups.yaml.example groups.yaml
```

CLI autocomplete feature installation (optional)

Usage of the autocomplete feature in CLI needs the `argcomplete` installation.

`Argcomplete` *Basic installation*

```
$ pip install argcomplete
$ activate-global-python-argcomplete
```

In case of errors insert a line like the one below into the *.bash.rc* file in your home directory.

```
eval "$(register-python-argcomplete my-awesome-script.py)"
```

Refresh your bash environment (start a new shell or *source* `/etc/profile`).

The REST service daemon needs the static configuration of the hardware's neighbours. An example configuration file is placed in `neighbours.yaml.example` file in the root directory of the source code. To start working on this file please make a copy and rename its name.

```
$ cp neighbours.yaml.example neighbours.yaml
```

4.5.4.3 Running

Command Line Interface usage:

```
./ofgw_main.py --help,
```

RESTful service daemon usage:

```
python ./daemon_ofgw_mngt.py [start|stop|restart]
```

Default HTTP service port: 5000

5 ClearingHouse

The control framework adopted by SDN experimental facilities such GENI and OFELIA suffers from drawbacks such as, tight-coupling of AAA mechanisms within the implementation of the system architecture; little or no regard for reusability (i.e., one AAA architecture cannot be reused by a different SDN experimental facilities); and no support for a standard access interface between the experimental network and the AAA architecture. To address these drawbacks, a certificate-based AAA (C-BAS) framework has been developed, which provides loose-coupling between the AAA architecture and the experimental network, it is reusable, and delivers AAA services through a well-defined interface, which ensures consistency and compliance between a experimental networks and the AAA architecture. Further details about C-BAS can be accessed from [C-BAS].

The ClearingHouse's source code is available at [ClearingHouse code].

5.1 Overview

Figure 5.1 illustrates the proposed AAA framework where the ClearingHouse (CH) manages user and slice certificates and credentials and a registration app which communicates user registration data both to the CH. The CH supports the Slice Facility Architecture (SFA) credential format as mandatory but also supports the alternative ABAC format. It is required that all entities must establish an SSL connection to interact with the CH. For this purpose, these network entities are provided with the certificates issued by the CH during the bootstraping process. The process of issuing a certificate to Structure the network entities and user-agents is not dynamic and must be performed by the CH administrator. In order to facilitate this process the CH provides an admin console which can be used by the administrators to generate SSL certificates signed by the root certificate of the CH.

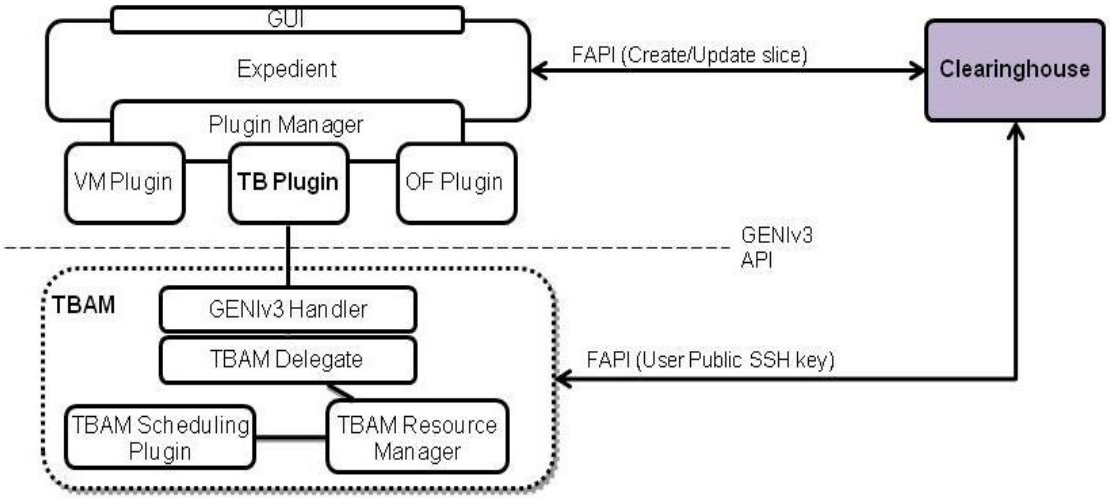


Figure 5.1: Overview of ALIEN Control Framework using the novel AAA mechanism

The user registration is performed in two steps as shown in Figure 5.2. In first step, the user is registered with the CH using a *registration app* provided by the CH. During this step of registration process, the user is asked if he wants to provide his public SSH key or should the CH generate the SSH key pair for her/him. If the SSH key is provided by the user, then the user information along with this SSH key is sent to the CH. The CH creates the member certificate and credentials, stores them in its database, and sends the created credential back to the registration application to make it available for user download. If the user wants the CH to create the SSH key pair for her/him, then the registration app creates the SSH key pair and sends the public key along with the registration info to the CH and also makes SSH key pair available for user download.

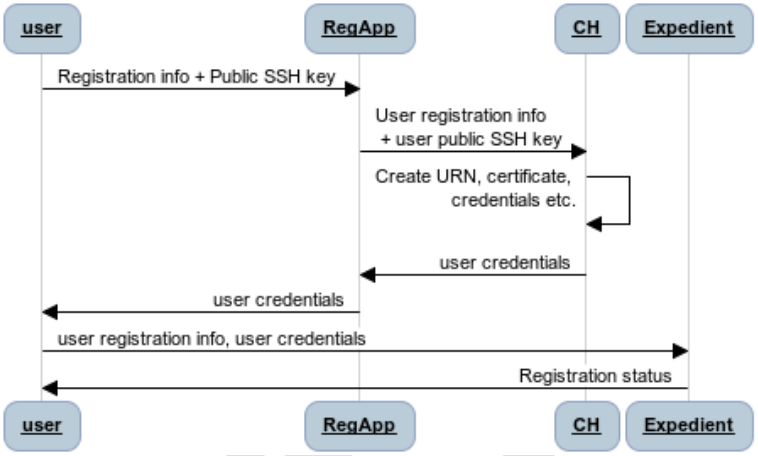


Figure 5.2: User registration process

In the second step, the user must register himself with the Expedient and set a password for login purposes. In addition, as a part of registration process the user must also provide his credentials issued by the CH and offered to him through registration app to allow Expedient communicate with the CH on her/his behalf.

The registered member now can login to the Expedient and perform the desired experiments. The Figure 5.3 shows how the ALIEN user interacts with the aggregate managers using Expedient. For example when creating a slice, Expedient fetches the user credentials from the local database and passes them to the CH for requesting a slice creation operation. The CH authorizes the request based on the provided member credentials, creates the slice and also the associated slice certificate and credentials. This information is stored in the CH database and the slice credentials are provided back to the Expedient. These slice credentials are then provided at the aggregate managers to authorize the user operation on this slice, for example, adding slivers to the slice. A slice has an expiration time before which it must be renewed or it will be deleted by the CH. In addition to creating or updating a slice, the CH also provides the lookup function. For this purpose, the user-agent must provide the user credentials to lookup slices associated with that user. Similarly, when an AM demands the public SSH key of the user to setup her/his login, for example, in one instantiated virtual machine, the user-agent requests its user's public SSH key from the CH by providing the user credentials or the Uniform Resource Name (URN).

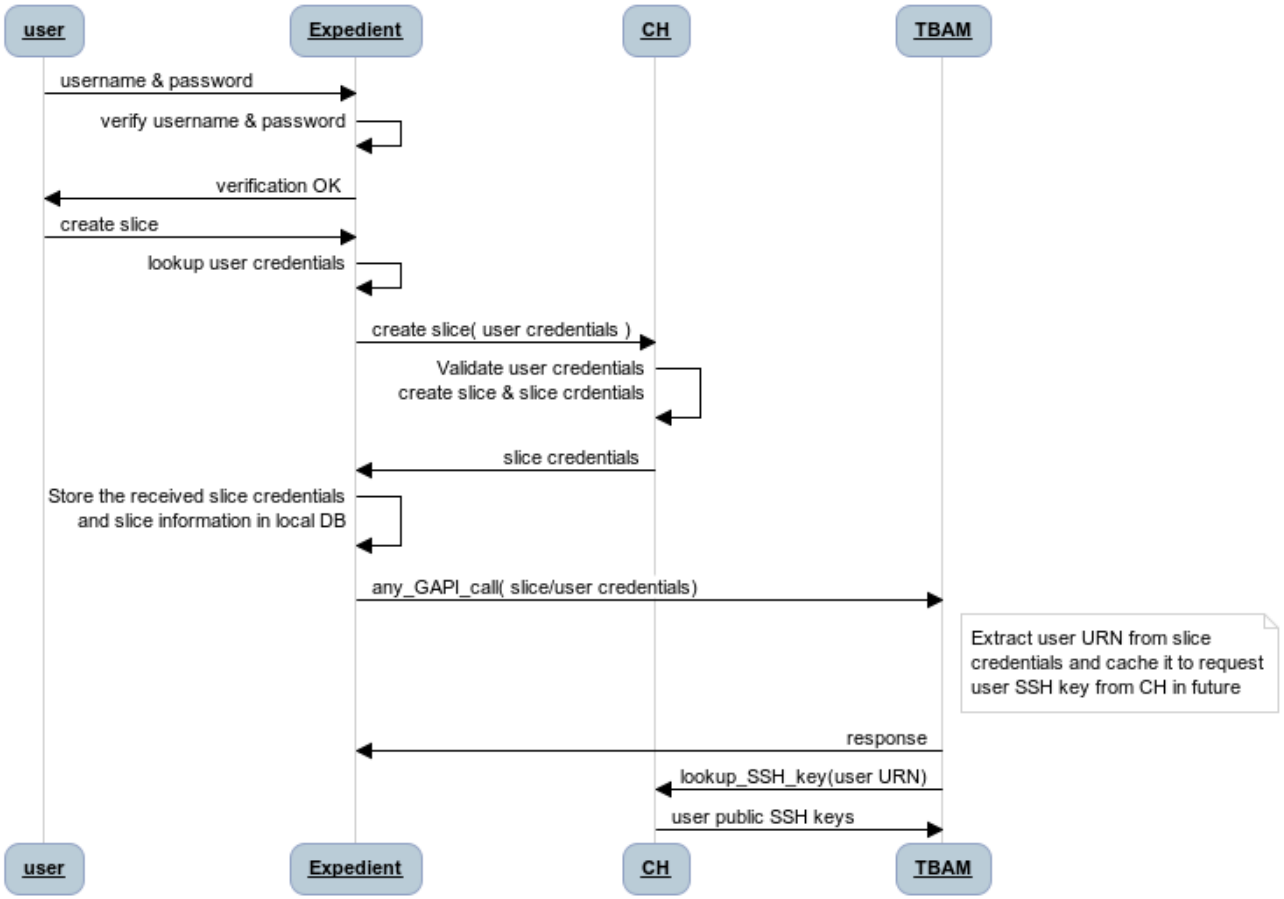


Figure 5.3: User interaction with TBAM using Expedient

5.2 Control Framework Components

This section provides necessary guidelines for the installation and configuration of ALIEN Control Framework components.

5.2.1 ClearingHouse

5.2.1.1 Dependencies

The ClearingHouse software installation on Debian-based system requires the following steps

- Check out the code from github available at `http://www.eict.de/c-bas`
- Install the following system dependencies:

```
sudo apt-get install python-setuptools python-dev swig libxmlsec1 xmlsec1 nginx
```

- Install the following Python dependencies using `requirements.txt` with pip package manager:

```
pip install -r requirements.txt
```

- Install the following dependencies for GENIv3RPC:

```
sudo apt-get install libxml2-dev libxslt-dev.
```

- Install additional dependencies

```
sudo apt-get install python-dateutil libxmlsec1 xmlsec1 libxmlsec1-openssl
libxmlsec1-dev
```

- Please install Swig from http://www.swig.org which is needed for the M2Crypto Python package.
- The CH currently relies on a MongoDB database running on the local host (and default port). This can be obtained from https://github.com/motine/Ohouse/blob/development/mongodb.org

5.2.1.2 Running

- Copy the following files and adjust the entries as required

- The `deploy/config.json.example` to `deploy/config.json`
- The `deploy/registry.json.example` to `deploy/registry.json`
- The `deploy/supplementary_fields.json.example` to `deploy/supplimentary_fields.json`

- Create test certificates and credentials and copy them to the respective places (see `test/creds/TODO.md`).
- Run the server with:

```
python src/main.py
```

- In a new console run the config client `python admin/config_client.py --interactive` and make change according to your setup

5.2.2 User Registration App

The user registration app follows server-client model where a server is run at the CH with the following command: python `src/main_registration.py`. The client side can be run with the command: `python Ohouse_registeration_app/registration_client.py`. However before running the registration app client the configuration file must be made available, for example, by running the command:

```
cp Ohouse_registeration_app/configuration/registration_app_config.json.example
Ohouse_registeration_app/configuration/registration_app_config.json
```

Figure 5.4 shows the graphical user interface of the client registration app.

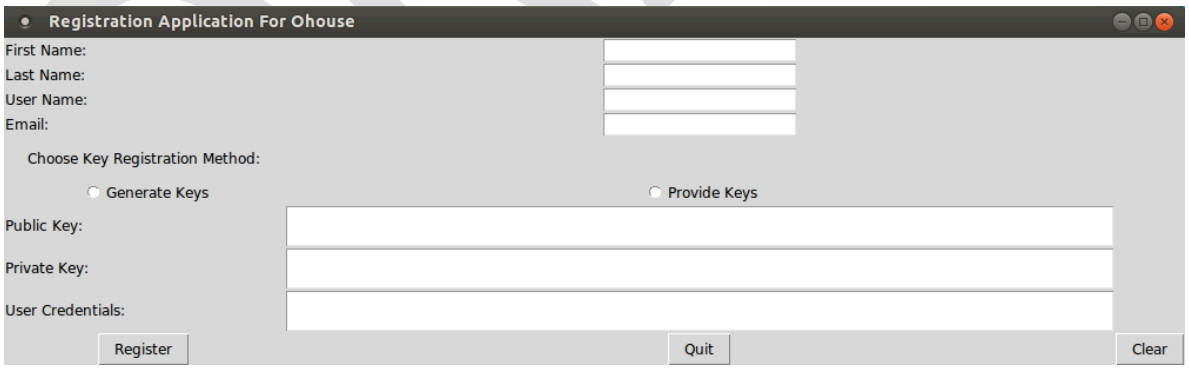


Figure 5.4: The graphical user interface of registration app

5.2.3 Expedient

Expedient user agent is a part of OFELIA Control Framework (OCF). The installation of OCF involves following steps:

5.2.3.1 Requirements

GNU/Linux Debian-based distribution (tested under Debian 6.0) with the following packages:

- `Python 2.6`
- `Django 1.2.3 (automatically installed)`
- `MySQL server (automatically installed)`

5.2.3.2 Installation

- Copy the `ofelia` folder under folder `/opt`
- Run the install script as a root user:

```
cd /opt/ofelia/deploy
python install.py
```

The install script triggers following actions:

- Installation of dependencies
- Certificate generation
- Configuration of Apache server
- Setting the file permissions
- Modification of the `localsettings.py` or `mySettings.py` depending on the component being installed
- Database population

5.2.3.3 Running

Start the Apache server with the command: `/etc/init.d/apache2 start`. In order to check if Expdient is up and running access its user interface using the local machine web browser under the following link: https://127.0.0.1. Figure 5.5 shows user registration page of Expedient.

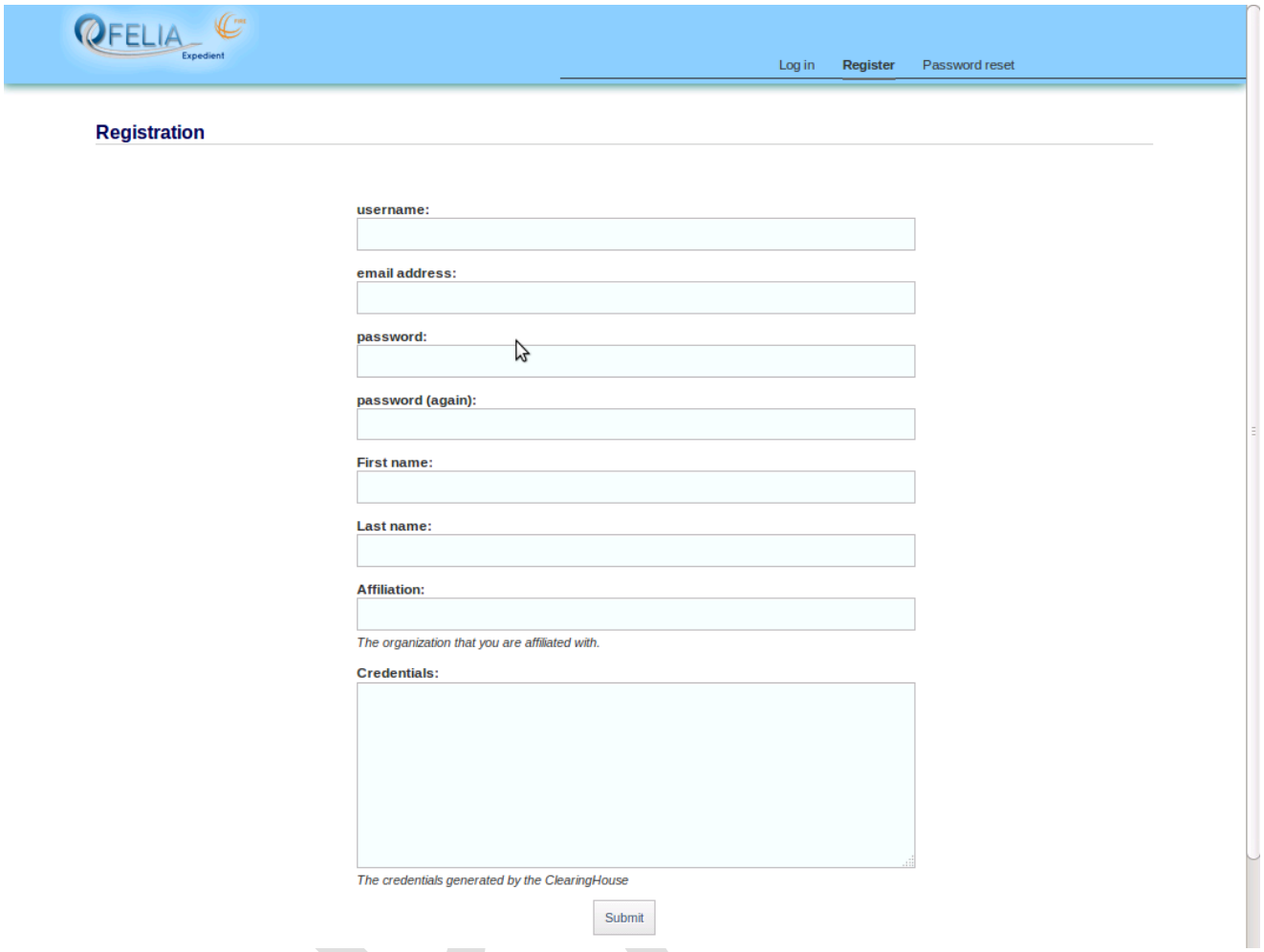


Figure 5.5: User registration page of Expedient

6 NETCONF AM

Existing management protocols such as SNMP [SNMP] suffer from several draw-backs such as:

- Un-reliable transport of management data (e.g., UDP)
- No clear distinction between operational & configuration data
- No support for roll-backs in case of errors / disasters
- Lack of configuration concurrency support (i.e., N:1 device configuration)
- No support for distinct transaction models (e.g., running, startup, candidate)

In 2003 a working group was formed in the Operations and Management area of the IETF to produce a protocol supporting network configuration. The outcome of this group is the development of the NETCONF protocol.

6.1 Overview

NETCONF is a configuration management protocol defined in RFC 6241 [RFC6241]. It provides multiple operations for interacting with configuration and operation data in the underlying managed device. The operation provided by NETCONF include, `get-config` for getting the configuration data, `edit-config` for modifying configuration data, `delet-config` for deleting configuration data, `lock` for concurrency support (i.e., to avoid race conditions in modifying the same configuration at the same time by different entities).

In addition to remote configuration management, NETCONF supports multiple configuration data stores such as `candidate, running and startup`. The `candidate` data store is there to prevent committing configuration changes instantly, instead only after a commit the configuration changed are submitted and deployed on the managed device. Furthermore, the running and startup data stores are those available when the device boots up or when the device is running.

The NETCONF protocol stack is comprised of four layers as shown in the following figure.

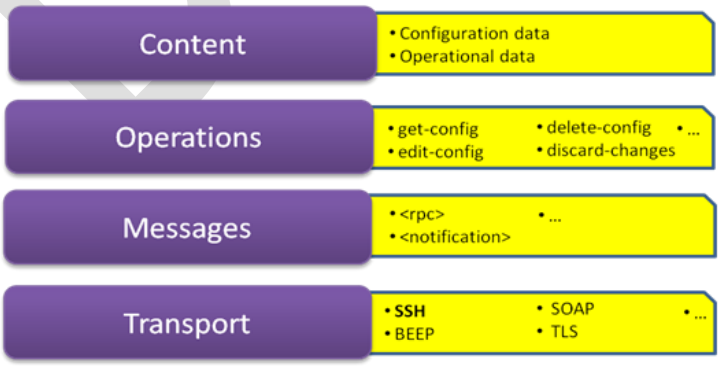


Figure 6.1: NETCONF protocol Stack

The Content layer is where the configuration and operation data resides. Within the Operations layer, XML-encoded parameters provide the base operations which can be performed in the configuration of the network device. The RPC layer provides a simple, transport independent, framing mechanisms for encoding RPCs. Finally, the Transport layer provides a path for communication between the client and the underlying managed devices, the NETCONF RFC recommends the use of three common security transport SSH, BEEP, and SOAP.

6.2 NETCONF in Hardware Abstraction Layer (HAL)

The NETCONF protocol is introduced in the HAL architecture [D2.2] as shown in the following figure.

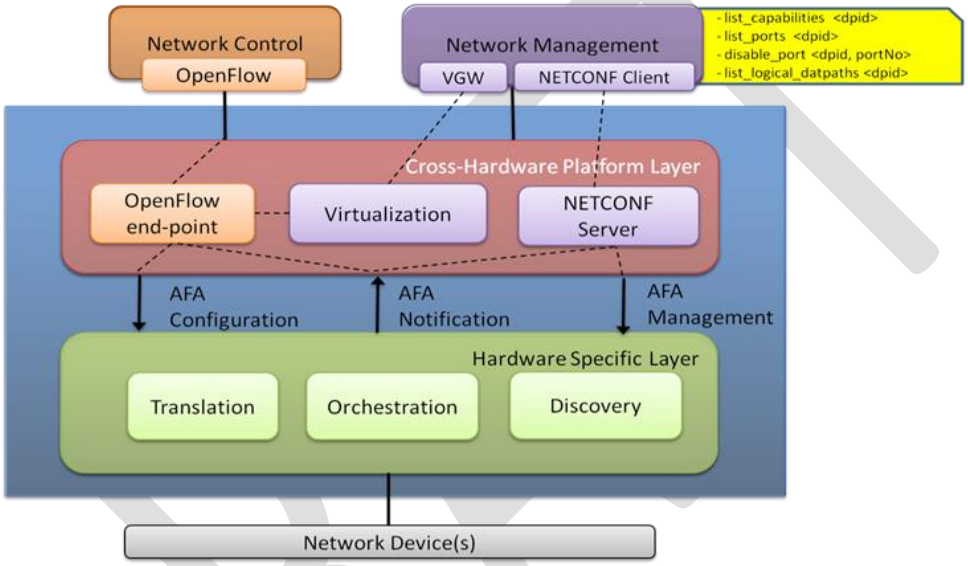


Figure 6.2: NETCONF Plugin in the HAL architecture

The NETCONF client is used by network administrators to modify and retrieve configuration parameters from the underlying ALIEN devices. To hide NETCONF specific details from the administrators and to facilitate the configuration ALIEN devices using NETCONF without having to understand the specifics of NETCONF commands, an abstraction layer is implemented on-top of the NETCONF client. This abstraction layer provides the administrator with a CLI from which she/he can execute management commands such as:

Command	Description
list_capabilities	*to see what are the available data models on the device for which can be used to configure*
list_ports	*to list the available ports on the device*
disable_port	*to disable a switch port of choice*
list_logical_datapths	*to see available OpenFlow datapths*

Table 6.1: List of management commands

6.3 NETCONF and AMSoil

In this section, the AMSoil framework is briefly introduced. Furthermore, the implementation of the NETCONF aggregate manager using the framework is described.

6.3.1 AMSoil Overview

AMSoil is a light framework that provides plugin-driven development for building aggregate managers in test-beds. It also provides the necessary glue between RPC handlers and resource managers.

Some of the advantages of AMSoil are:

- Exchangeability: replacing implementation without affecting functionality
- Encapsulation: protect implementation from other developers
- Clarity: abstraction of un-necessary details
- Encapsulation: Protect implementation from tampering

A service in AMSoil can be divided into three parts:

- Handler: receives high-level RPC requests
- Delegate: translates high-level requests for resource managers
- Resource Manger: Handles the actual allocation of the resources

6.3.2 NETCONF implementation in AMSoil

The NETCONF protocol defines the functions of two entities:

- **The Agent**: resides in the underlying (to-be managed) device.
- **The Client**: allows a network administrator to issue configuration & management commands

The client implementation is divided into three parts as shown in Figure 6.3.

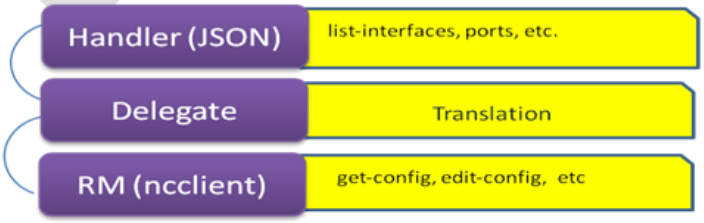


Figure 6.3: The structure of the NECONF client in AMSoil

To understand how the management client works, please depict the following figure:

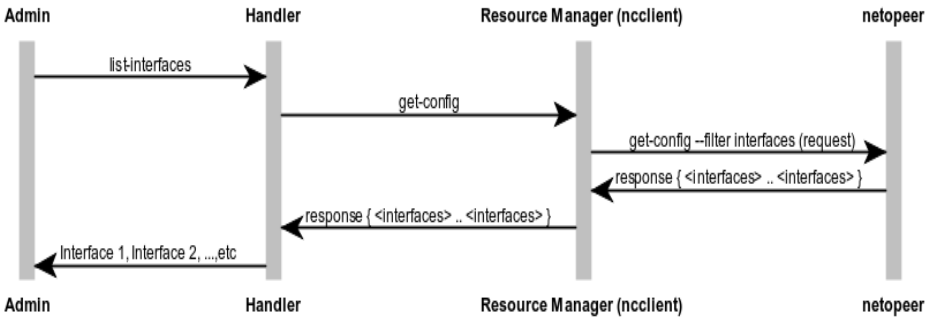


Figure 6.4: The NETCONF plugin workflow

The handler part of the client receives RPC calls such as `list_interfaces , list_ports, etc`. and then forwards the requests to the client delegate. The delegate maps high-level RPC calls to low level NETCONF commands. For example, when the `list_interfaces` method is called, a NETCONF `get-config` with a filter set to *interfaces* is called. In the *resource manager* , the output of the call would look like this:

```
<interfaces xmlns="http://zanetworker.com/ns/my-box">
    <interface>
      <name> eth0 </name>
    </interface>
  </interfaces>
```

The resources manger makes use of the `ncclient` library [ncclient] which is an open source client side implementation of NETCONF to make the calls.

6.4 Installation of the NETCONF plugin

In this section a brief description of the installation steps required to have the NETCONF plugin up and running. The code for the NETCONF plugin is available at [NETCONF code]

6.4.1 Installation Requirements

To install the NETCONF client, the following requirements should be fulfilled:

- Flask==0.10.1
- Flask-XML-RPC==0.1.2
- Jinja2==2.7.2
- M2Crypto==0.22.3
- MarkupSafe==0.18
- SQLAlchemy==0.9.3
- Werkzeug==0.9.4
- blinker==1.3
- cffi==0.8.1
- cryptography==0.2.2
- flup==1.0.2
- itsdangerous==0.23
- lxml==3.3.3
- pika==0.9.13
- pyOpenSSL==0.14
- pycparser==2.10
- python-dateutil==2.2
- pytz==2014.1.1
- six==1.5.2
- wsgiref==0.1.2

To install these requirements either install each package independently or use the requirements file (which comes with the repository)

```
pip install -r requirements.txt
```

Furthermore, to configure the server the configuration parameters for the RPC server and the NETCONF server should be set. The configuration file can be found in `deploy/config.json.example`. This file needs to be copied into another file called `config.json`. The content of the file could be as follows:

```
"NETCONF_RPC_SERVER": {
       "description"  : "Configuration of the RPC NETCONF server",
       "server" : "localhost",
       "port" : "2222"
},

"NETCONF_SERVER": {
       "server" : "localhost",
       "port" : "830",
       "password" : "password",
       "yang_namespace" : "http://zanetworker.com/ns/my-box"
}
```

the password and the server hostname/IP need to be changed accordingly.

6.4.2 Running

To run the NETCONF plugin within AMSoil, two steps are involved:

- First the AMSoil main file needs to be running using `python src/main.py`
- Then the main file which runs the NETCONF server should run in parallel `python src/main_netconf.py`

Conclusions

This document is the last deliverable of Work Package 4 and reports the implementation results achieves in Task T4.3. In particular this document describes how the architecture defined in the Deliverable [D4.2] has been implemented in order to meet the requirements identified in Deliverable [D4.1].
In order to achieve the goal of the Work Package, i.e. “Alien Hardware Integration within OFELIA Control Framework”, three main modules have been implemented: the TB Plugin for Expedient , the TBAM aggregate manager and the OFGW resource manager.
Beside the integration within the OFELIA Control Framework, an alternative aggregate manager has been proposed. Like the TBAM (which was developed to achieve the integration), this aggregate manager is build on top of AMsoil and is based on the NetConf protocol.
Additionally, an AAA architecture for experimental facilities have been designed and implemented with the aim to propose to the research community a new mechanism to manage identities, credentials and certificates and, at the same time, decoupled from the experimental facility.
At the time of writing this document, all the software modules have been released in a stable and tested version. Tough in the last two months of the project the testing process will continue within the activities of Task T5.3 (“Experimental-driven research: test, validation and reporting activities”) in order to improve the code and to fix any possible bug.

Source code repositories

OCF-TB Plugin	https://github.com/fp7-alien/OCF-TBPlugin
OCF-TBAM	https://github.com/fp7-alien/OCF-TBAM
OCF-OFGW	https://github.com/fp7-alien/OCF-OFGW/
ClearingHouse	https://github.com/zanetworker/C-BAS
NETCONF	https://github.com/zanetworker/NETCONF_AM

References

[D4.1]	https://wiki.man.poznan.pl/alien/index.php/D4.1
[D4.2]	https://wiki.man.poznan.pl/alien/index.php/D4.2
[Expedient]	http://yuba.stanford.edu/~jnaous/expedient/
[FlowVisor]	https://github.com/OPENNETWORKINGLAB/flowvisor/wiki
[GENIv3 API]	http://groups.geni.net/geni/wiki/GeniApi
[AMsoil]	https://github.com/motine/AMsoil
[AMsoil plugin]	https://github.com/motine/AMsoil/wiki/Plugin
[AMsoil installation]	https://github.com/motine/AMsoil/wiki/Installation
[AMsoil RM]	https://github.com/motine/AMsoil/wiki/GENI#wiki-resource-manager
[AMsoil worker]	https://github.com/motine/AMsoil/wiki/Worker
[AMsoil schedule]	https://github.com/motine/AMsoil/wiki/Schedule
[MGMT Actions code]	https://github.com/fp7-alien/OCF-OFGW/tree/master/ofgw_mngt
[OCF]	https://github.com/fp7-ofelia/ocf/wiki/Development
[OFELIA]	http://www.fp7-ofelia.eu/
[OvS]	http://openvswitch.org/
[RFC2641]	http://tools.ietf.org/html/rfc6241
[RSpec]	http://groups.geni.net/geni/wiki/GeniRspec
[SNMP]	https://www.ietf.org/rfc/rfc1157.txt
[D2.2]	https://wiki.man.poznan.pl/alien/index.php/D2.2
[TBAM code]	https://github.com/fp7-alien/OCF-TBAM
[TBAM Delegate]	https://github.com/fp7-alien/OCF-TBAM/tree/master/AMsoil/src/plugins/islandgeni3
[TBAM RM]	https://github.com/fp7-alien/OCF-TBAM/tree/master/AMsoil/src/plugins/islandRM
[TBAM scheduling plugin]	https://github.com/fp7-alien/OCF-TBAM/tree/master/AMsoil/src/plugins/schedule
[TBAM Agent code]	https://github.com/fp7-alien/OCF-OFGW/tree/master/TBAM-Agent
[TB Plugin code]	https://github.com/fp7-alien/OCF-TBPlugin
[ClearingHouse code]	https://github.com/zanetworker/C-BAS
[NETCONF code]	https://github.com/zanetworker/NETCONF_AM
[conntrack]	http://www.netfilter.org/projects/conntrack-tools/index.html
[iptables]	http://www.netfilter.org/projects/iptables/index.html
[C-BAS]	U. Toseef, et al., "C-BAS: Certificate-based AAA for SDN Experimental Facilities", in Proc. EWSDN 2014, Sept. 2014 (Accepted)
[OFELIA paper]	M. Sune et al., “Design and Implementation of the OFELIA FP7 Facility: “The European OpenFlow Testbed,” Computer Networks, vol. 61, 2014.