© All Rights Reserved

HORIZON2020 FRAMEWORK PROGRAMME

TOPIC EUK-03-2016

"Federated Cloud resource brokerage for mobile cloud services"

D5.6
Service, security, and privacy quality enforcement: Software prototype

Project acronym: BASMATI

Project full title: *Cloud Brokerage Across Borders for Mobile Users and Applications*

Contract no.: 723131

Workpackage:	5	Service, security and privacy enforcement: Software prototype	
Editor:	Enric Pages		ATOS
Author(s):	Burak Karaboğa		ATOS
	Vinicius Monteiro de Lira		CNR
	Emanuele Carlini		CNR
	Jamie Marshall		AMEN
Authorized by	Young Woo Jung Konstantinos Tserpes		ETRI ICCS
Doc Ref:	5.6		
Reviewer	Konstantinos Tserpes		ICCS
Dissemination Level	Public		

© All Rights Reserved

Document History

Version	Date	Changes	Author/Affiliation
0.1	26-03-2018	Initial ToC	Enric (ATOS)
0.2	09-04-2018	Initial content	Enric (ATOS)
0.3	12-04-2018	Refinement + Screenshots	Enric (ATOS)
0.4	16-04-2018	Fill pending sections	Enric (ATOS)
0.5	20-04-2018	Send to Quality review process	Konstantinos Tserpes (ICCS)
1.0	23-04-2018	Final	Enric (ATOS)

© All Rights Reserved

BASMATI Glossary

Term/Acronym	Definition
Mobile cloud services	Online services offered by cloud resources to support mobile apps. The backend of the mobile apps.
CP	Cloud Provider. The actor that provides the cloud infrastructure/resources, such as VMs
CSP	Cloud Service Provider. The actor that provides cloud services on top of a rent infrastructure from a CP
Cloudlet	Limited capacity infrastructures with virtualization capabilities, often used to support a limited amount of users or perform a limited set of operations on behalf of the central cloud infrastructure that hosts the complete application
Edge resources	Resources aimed to operate specialized functionality, located at the "edge" of the network infrastructure, thus, closer to the end users. Examples are (clusters of) RaspberryPis or cloudlets
BUDaMaF	BASMATI Unified Data Management Framework
KE	Knowledge Extractor
DM	Decision Maker
RB	Resource Broker
MVD	Mobile Virtual Desktop
DASFEST	An 3-day long music festival taking place in Karlsruhe, Germany every July
ACE	Amenesik Cloud Engine. The cloud service deployment tool through which actual federation is achieved
BEAM	BASMATI Enhanced Application Model. An extension of the TOSCA specification
ASP	Application Service Provider. A Federation user that rents resource services in order to provide an Application services to End-users
Brokering	The matchmaking support provided by BASMATI platform to decide about the best cloud resources to exploit for the execution of the back-end of BASMATI applications. This activity regards the placement of the services or data on computational resources and storages belonging to the cloud data centre and the cloudlets within the federation.
End user	A user who benefits the various application and infrastructure services provided by the Cloud. Within BASMATI, the most typical example is exploiting the Cloud federation via a mobile device (possibly a laptop) using specialized apps or a web browser.

© All Rights Reserved

Offloading	The ability of BASMATI platform supporting the runtime placement of the components composing the front-end of BASMATI applications on edge resources available nearby the end user. This activity takes place both when edge and mobiles exchange one each other their own workload or when such devices transfer some workload to the clouds or cloudlets. In BASMATI we often distinguish Front-end offloading, related to the mobile part of application, from Back-end offloading, concerning the server side of applications. The latter roughly translates to the known concept of Cloudbursting.
QoE	Quality of experience. It is a measure of a customer's experiences with a service. It may be related to some aspects of the QoS and QoP, but can also take into account other metrics.
Service handover	Service handover refers to the activity of transferring an active service between two computational resources (e.g. Cloudlets) with minimal or no disruption on the availability of the service. Ideally, service handover is transparent with respect to the user.
Situational Awareness	The ability of the BASMATI platform to recognise the “situation” characterising the actual combined status of users, applications and resources, aimed at achieving an effective and efficient management of applications and resources.

© All Rights Reserved

Executive Summary

The aim of "WP5 Dynamic Brokerage and Federation: Realization" is realising the decisions taken with respect t to the federation, brokerage and offloading. The objectives to target are related with the management of hybrid infrastructures within a federation implementing different techniques for the needs of the different scenarios supported in the project.

This report D.5.6, delivers software components developed/adapted based on the design and specifications carried out in "*D.5.5 Quality, security and privacy enforcement design and specification*" which delivers the specification of the mechanisms for ensuring quality, security and privacy in the BASMATI ecosystem.

The document outlines the scope of the software prototypes involved in the quality enforcement process within BASMATI; both at the provider level as well as at federation level, where several providers are working in cooperation to accommodate the placement strategy selected coming from the decision making process within BASMATI.

This document is accompanying report of the software modules developed for the quality enforcement and data management within federation.

© All Rights Reserved

Table of Contents

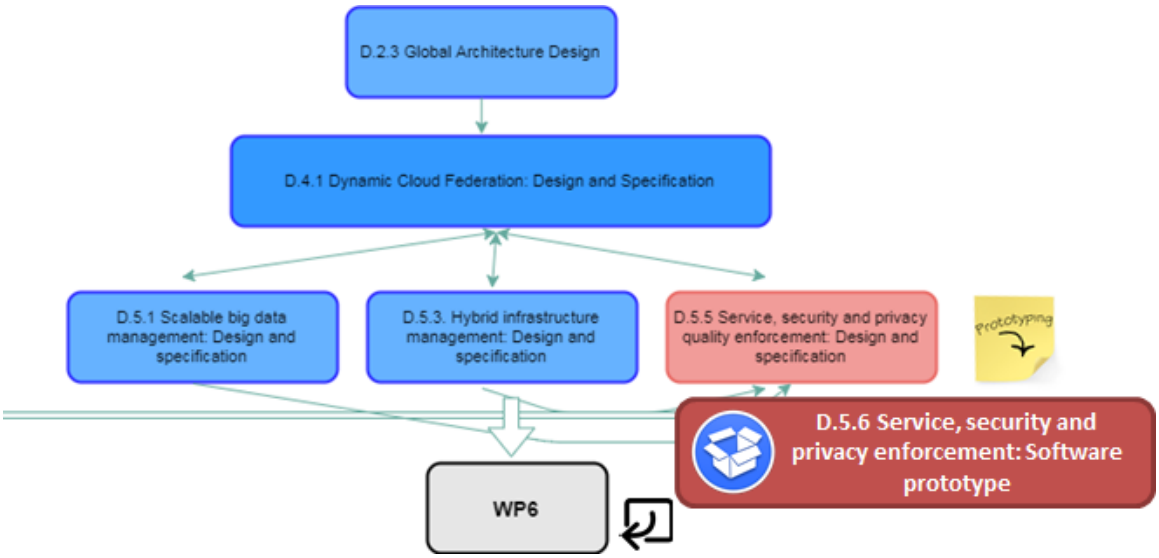
© All Rights Reserved

1 Introduction

BASMATI project aims at the realization of an innovative brokerage platform targeting scalable management of heterogeneous distributed and federated resources. In federated clouds we have to deal with the deployment and management of both internal and external cloud computing services, to satisfy the business needs. In the aforementioned case, a federation is the union of several CSPs that perform a common action binding the management of the inter-connected clouds through a common Quality of Service (QoS) policy.

The document is the accompanying report of the ***BASMATI Federation Manager*** software module within the ***Federation Business Logic*** component group. The aforementioned module works in close cooperation with other components within the same group, as well as with other BASMATI entities in order to manage the lifecycle of Service Level Agreements (SLAs) within the platform, which constitutes the legal bindings between the application plane and the infrastructure plane.

1.1 Relationship to Other Deliverables



Due to the management of the agreements between different entities plays and important role in BASMATI, this report has links with of the software prototypes delivered as part of WP3, WP4, and WP5. In order to describe the role of the module in the platform, the report mentions the relationship with these software modules referencing the reports where the related modules have been released. (Further information can be found on "*D.5.5 Quality, security and privacy enforcement design and specification*".

© All Rights Reserved

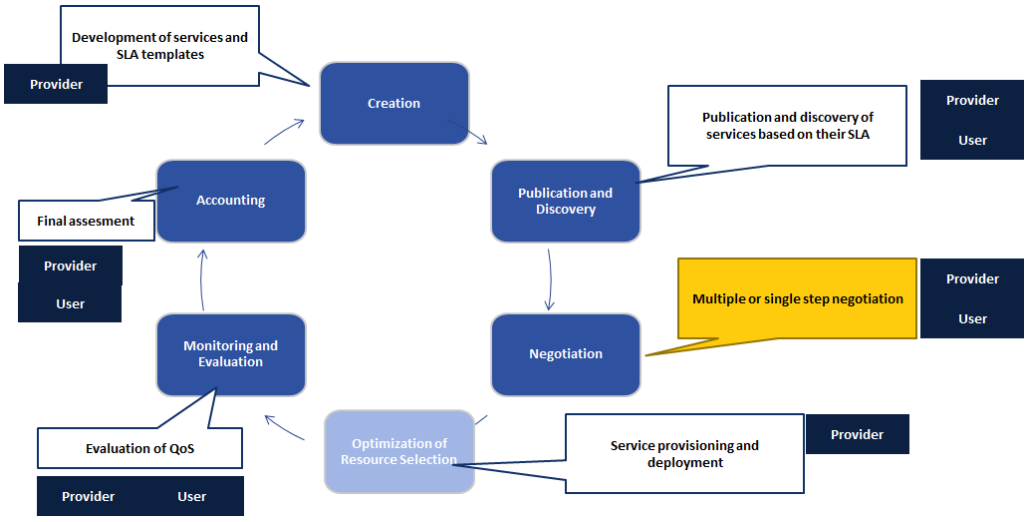
1.2 Scope of the prototype

This report delivers 4 components: **Federation SLA Manager (FSLAM)**, **Provider SLA Manager** (packed together), **Agreement Repository** as well as the **SLA Dashboard** module.

Additionally, the design and specification report D.5.5 describes interactions with other components in order to illustrate the behavior of the FSLAM within the platform. As a consequence, other modules could be mentioned: the Amenesik Cloud Engine (ACE) and the Application Controller (AC) which constitutes the entities bound by the SLA; being the formal contract which defines acceptable service levels to be provided in form of negotiated terms. Those terms, *known as Guarantee Terms*, binds the Cloud Service Providers (CSP) with their customers, in case of BASMATI the relation is established between the ACE modules acting as CSPs entry point and the Application Controller which manages the application states at deployment and runtime.

The ACE is the key component for federation among heterogeneous clouds, cloudlets and edges. It keeps all information of federated clouds and cloudlets, and supports OCCI standard interfaces to manage the infrastructures and services. In addition to the management of the deployment, ACE module creates SLAs which enforce that the capacity requested by the application service can be archived.

The interactions of the aforementioned modules cover the creation, publication, negotiation, and optisation of resource selection (see figure below) of the application, prior to the deployment of the service in the target environment.



When the resources are deployed according to the placement strategy selected, the offer (knows as SLA template) stored in the ***BEAM** Repository* becomes an agreement (based on the international standard WS-Agreement), at runtime the contract is assessed and enforced by the Federation SLA Management module which monitors the fulfilment of the *Guarantee Terms*

© All Rights Reserved

thanks to the Federation Monitoring System within BASMATI, in addition to the assessment of the capacity of the target resources, application level metrics can be provided through an incident signalling interface offered by the Federation SLA module, these information is collected and forwarded by monitoring probes instantiated alongside with the application by the ACE module. The process described covers the monitoring, evaluation and assessment stages depicted in the figure above.

2 Prototype Demonstration

In order to help with the dissemination and convey as much information as possible to the reader in a short period of time, this prototype will be demonstrated in the form of a video uploaded to Youtube.

2.1 Location

The demonstration video is located here:

https://youtu.be/zNXNIXYNXgo

The endpoint for the REST API is here:

http://182.252.135.45:8080/api

Source code can be found here:

http://basmati.amenesik.com/code/platform/modules/cloud_sla_manager

2.2 Description

In the first part of the video the agreement templates hosted in the Application Repository and then the agreement based on this template which belongs to a deployed service is shown in the SLA Manager Dashboard.

The second part of the video focuses on the resource operations allowed by the REST API and how they reflect on the SLA Dashboard.

Finally, the video shows the available SLAs in the Amenesik Cloud Engine.

3 Prototype Installation, Configuration and Deployment Guide

The following processes defined below assume you are using Linux-like prompt (tested with Cygwin in Windows environments).

© All Rights Reserved

3.1 Authentications

The source code of the prototype is located in the private Git repository mentioned in section 2.1. The credentials below can be used to access the source code:

username: guest

password: basmati_review7012

3.2 Requirements

Below you have described the procedures that compiles, setup the environment and installs the core and the dashboard into the VM. For additional information about the installation process, check the component documentation (READMEs).

The list of prerequisites common for all the procedures is the following:

- Git
- >= Java 7 SDK
- >= Maven 2.0

3.2.1 Manual Installation steps

Normal deployment prerequisites:

- WebServer such as Tomcat, Jetty , others ...
- MySQL-like database

Follow well-documented procedures available on Internet to install a web server as well as a MySQL database, the final artefact is deployed into the web server and fetch information from the database, so-called Agreement Repository.

Manual installation, configuration and deployment

```
>> git clone git@basmati.amenesik.com:code/platform/modules/cloud_sla_manager.git
>> cd sla-framework/sla-core
# Create the database


# Create the database structure
```

```
# For additional scripts related with databases operations.
# Check “sla-framework\sla-core\bin” for further information

# Compiling the artefact

>> cd sla-framework/sla-core

# Configure the web service details in “sla-framework/sla-core”, configuration.properties file.

>> mvn clean package

# Deploy the resultant war file in your preferred container.
# Light-weight Jetty example provided below

>> java -jar sla-service/target/dependency/jetty-runner.jar --port 8080 --path / sla-service/target/sla-service.war

# Check everything is working accessing the REST interface http://<IPaddress>:8080/api/
```

3.2.2 Vagrant deployment

Vagrant deployment prerequisites:

- VirtualBox
- Vagrant >= 1.8.1

Vagrant installation, configuration and deployment

```
>> git clone git@basmati.amenesik.com:code/platform/modules/cloud_sla_manager.git

>> cd sla-framework

>> dist/bin/make-dist.sh vagrant up

# This creates a VM with IP: 10.10.10.100, with core and dashboard running at ports 8080 and 8000, respectively.

# Check everything is working: http://10.10.10.100:8000 (dashboard) or accessing directly the component REST interface http://10.10.10.100:8080/api/
```

3.2.3 Containerisation with Docker-compose

Docker deployment prerequisites:

© All Rights Reserved

- VirtualBox (optional)
- Vagrant >= 1.8.1 (optional)
- Docker and Docker-compose

Docker installation, configuration and deployment

```
# Access your Docker instance

# Download the source code from the repository

>> git clone git@basmati.amenesik.com:code/platform/modules/cloud_sla_manager.git

# From your project directory type (for additional ways to instantiate and access the container check Docker documentation)

>> docker-compose up

# Check everything is working accessing the REST interface http://<IP_address>:8080/api/
```

3.3 Prototype Usage Guide

3.3.1 Federation SLA Manager

Once instantiated we need to populate the manager.

Usage Guide

>> cd sla-core
>> export SLA_MANAGER_URL=http://<IP_adress>:8080/api
 # Load samples (Provider, Template, Agreement samples)
>> bin/load-samples-core.sh

 # Note: The default installation comes with a mock Monitoring adapter that allow you to test the component without the need to connect it to a Monitoring service which is needed for the evaluation/assessment process.

 # The sample agreement is not being evaluated yet, as the usual lifecycle process has not been follow (Creation, publication, negotiation, optimisation) so in order to start the evaluation we can execute

>> bin/start-evaluation.sh

Further information about the usage of the component and how to interact with the REST API offered can be found in "*D.5.5 Quality, security and privacy enforcement design and specification*" where the component interfaces and its usage were documented.

© All Rights Reserved

See bellow some examples on how to interact with the REST API and the type of outputs that you can expect:

GET /providers

```
$ curl  -H "Accept: application/xml" http://basmati:8080/api/providers

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<providers>
  <provider>
    <uuid>provider-a</uuid>
    <name>provider-a</name>
  </provider>
  <provider>
    <uuid>provider-b</uuid>
    <name>provider-b</name>
  </provider>
</providers>
```

© All Rights Reserved

GET /templates

```
$ curl  -H "Accept: application/xml" http://basmati:8080/api/templates

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<templates>
  <wsag:Template xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement" xmlns:sla="http://sla.atos.eu" wsag:TemplateId="template-a">
    <wsag:Name>TemplateA</wsag:Name>
    <wsag:Context>
      <wsag:AgreementResponder>provider-a</wsag:AgreementResponder>
      <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
      <wsag:ExpirationTime>2014-03-07T13:00:00+01:00</wsag:ExpirationTime>
      <sla:Service>service03</sla:Service>
    </wsag:Context>
    <wsag:Terms>
      <wsag:All>
        <wsag:ServiceDescriptionTerm wsag:Name="SDTName2" wsag:ServiceName="service-a">DSL expression</wsag:ServiceDescriptionTerm>
        <wsag:ServiceProperties wsag:Name="NonFunctional" wsag:ServiceName="service-a"/>
        <wsag:GuaranteeTerm wsag:Name="GT_ResponseTime">
          <wsag:ServiceScope>ServiceName</wsag:ServiceScope>
          <wsag:ServiceLevelObjective>
            <wsag:KPITarget>
              <wsag:KPIName>ResponseTime</wsag:KPIName>
              <wsag:CustomServiceLevel>{"constraint" : "ResponseTime LT qos:ResponseTime"}</wsag:CustomServiceLevel>
            </wsag:KPITarget>
          </wsag:ServiceLevelObjective>
        </wsag:GuaranteeTerm>
        <wsag:GuaranteeTerm wsag:Name="GT_Performance">
          <wsag:ServiceScope>ServiceName</wsag:ServiceScope>
          <wsag:ServiceLevelObjective>
            <wsag:KPITarget>
              <wsag:KPIName>Performance</wsag:KPIName>
              <wsag:CustomServiceLevel>{"constraint" : "Performance GT qos:Performance"}</wsag:CustomServiceLevel>
            </wsag:KPITarget>
          </wsag:ServiceLevelObjective>
          <wsag:BusinessValueList>
            <wsag:CustomBusinessValue count="1">
              <sla:description/>
            </wsag:CustomBusinessValue>
          </wsag:BusinessValueList>
        </wsag:GuaranteeTerm>
      </wsag:All>
    </wsag:Terms>
  </wsag:Template>
</templates>
```

© All Rights Reserved

GET /agreements

```
$ curl  -H "Accept: application/xml" http://basmati:8080/api/agreements

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<agreements>
  <wsag:Agreement xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement" xmlns:sla="http://sla.atos.eu" wsag:AgreementId="agreement-a">
    <wsag:Name>ExampleAgreement</wsag:Name>
    <wsag:Context>
      <wsag:AgreementInitiator>client-prueba</wsag:AgreementInitiator>
      <wsag:AgreementResponder>provider-a</wsag:AgreementResponder>
      <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
      <wsag:ExpirationTime>2019-03-07T13:00:00+01:00</wsag:ExpirationTime>
      <wsag:TemplateId>template-a</wsag:TemplateId>
      <sla:Service>service-a</sla:Service>
    </wsag:Context>
    <wsag:Terms>
      <wsag:All>
        <wsag:ServiceProperties wsag:Name="ServiceProperties" wsag:ServiceName="service-a">
          <wsag:VariableSet>
            <wsag:Variable wsag:Name="ResponseTime" wsag:Metric="xs:double">
              <wsag:Location>service-prueba/ResponseTime</wsag:Location>
            </wsag:Variable>
            <wsag:Variable wsag:Name="Performance" wsag:Metric="xs:double">
              <wsag:Location>service-prueba/Performance</wsag:Location>
            </wsag:Variable>
          </wsag:VariableSet>
        </wsag:ServiceProperties>
        <wsag:GuaranteeTerm wsag:Name="GT_ResponseTime">
          <wsag:ServiceScope wsag:ServiceName="service-a">scope-a</wsag:ServiceScope>
          <wsag:ServiceLevelObjective>
            <wsag:KPITarget>
              <wsag:KPIName>ResponseTime</wsag:KPIName>
              <wsag:CustomServiceLevel>{"constraint" : "ResponseTime LT 0.5"}</wsag:CustomServiceLevel>
            </wsag:KPITarget>
          </wsag:ServiceLevelObjective>
        </wsag:GuaranteeTerm>
        <wsag:GuaranteeTerm wsag:Name="GT_Performance">
          <wsag:ServiceScope wsag:ServiceName="service-a">scope-b</wsag:ServiceScope>
          <wsag:ServiceLevelObjective>
            <wsag:KPITarget>
              <wsag:KPIName>Performance</wsag:KPIName>
              <wsag:CustomServiceLevel>{"constraint" : "Performance BETWEEN (0.3,1)"}</wsag:CustomServiceLevel>
            </wsag:KPITarget>
          </wsag:ServiceLevelObjective>
          <wsag:BusinessValueList>
            <wsag:CustomBusinessValue count="1">
              <sla:Penalty type="discount" expression="35" unit="%" validity="P1D"/>
              <sla:description/>
            </wsag:CustomBusinessValue>
            <wsag:CustomBusinessValue count="5" duration="P1D">
              <sla:Penalty type="service" expression="sms" unit="" validity="P1M"/>
              <sla:description/>
            </wsag:CustomBusinessValue>
          </wsag:BusinessValueList>
        </wsag:GuaranteeTerm>
      </wsag:All>
    </wsag:Terms>
  </wsag:Agreement>
</agreements>
```

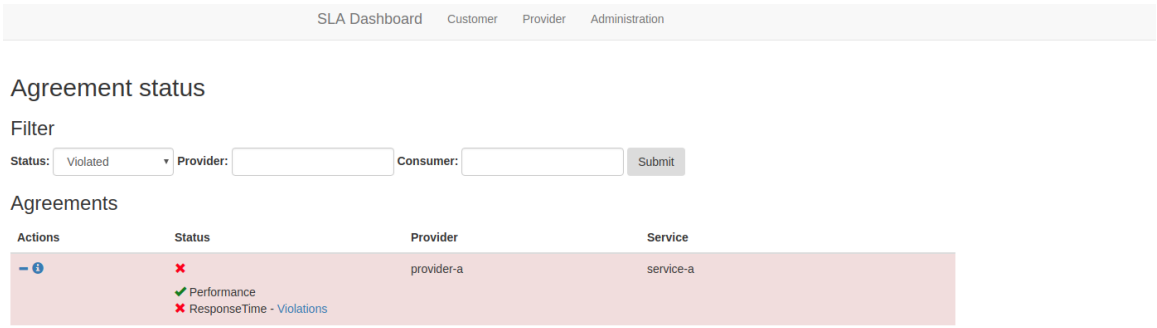
GET /providers

```
$ curl  -H "Accept: application/xml" http://basmati:8080/api/violations

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
 <violations>
   <violation>
     <uuid>42374f63-6553-4346-863d-11501ecc1b6a</uuid>
     <agreement_id>agreement-a</agreement_id>
     <service_name>service-a</service_name>
     <service_scope>scope-b</service_scope>
     <kpi_name>Performance</kpi_name>
     <datetime>2018-04-11T15:39:01+02:00</datetime>
     <expected_value>0.5</expected_value>
     <actual_value>2</actual_value>
  </violation>
  <violation>
    <uuid>680504c6-e5ed-4f68-b30d-92c0eb0d3549</uuid>
    <agreement_id>agreement-a</agreement_id>
    <service_name>service-a</service_name>
    <service_scope>scope-b</service_scope>
    <kpi_name>Performance</kpi_name>
    <datetime>2018-04-11T15:39:02+02:00</datetime>
    <expected_value>0.5</expected_value>
    <actual_value>3</actual_value>
  </violation>
</violations>
```

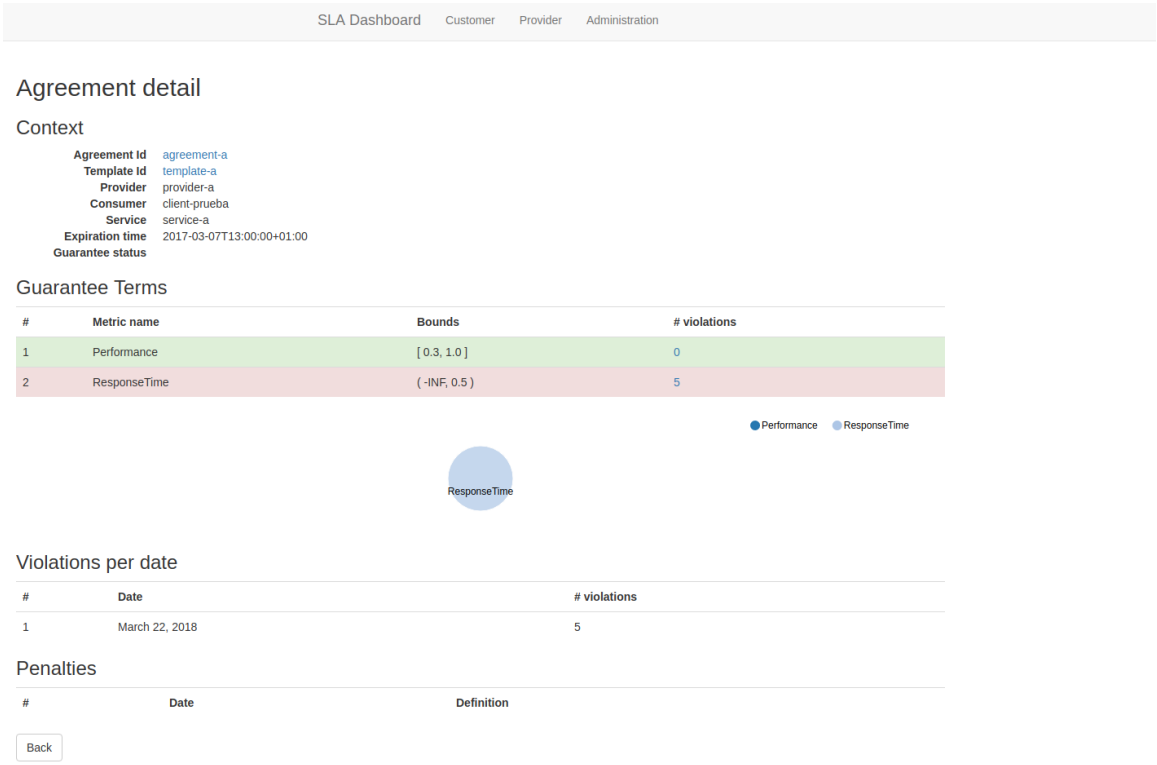
GET /providers

```
$ curl  -H "Accept: application/xml" http://basmati:8080/api/penalties

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<penalties>
  <penalty xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement" xmlns:sla="http://sla.atos.eu">
    <uuid>83e19690-bf2a-405b-a101-72793f8e027f</uuid>
    <agreement_id>agreement-a</agreement_id>
    <datetime>2018-04-11T15:39:01+02:00</datetime>
    <definition type="discount" expression="35" unit="%" validity="P1D"/>
  </penalty>
  <penalty xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement" xmlns:sla="http://sla.atos.eu">
    <uuid>5e262394-da1b-4d26-8314-66684ff7cd23</uuid>
    <agreement_id>agreement-a</agreement_id>
    <datetime>2018-04-11T15:39:02+02:00</datetime>
    <definition type="discount" expression="35" unit="%" validity="P1D"/>
  </penalty>
</penalties>
```

© All Rights Reserved

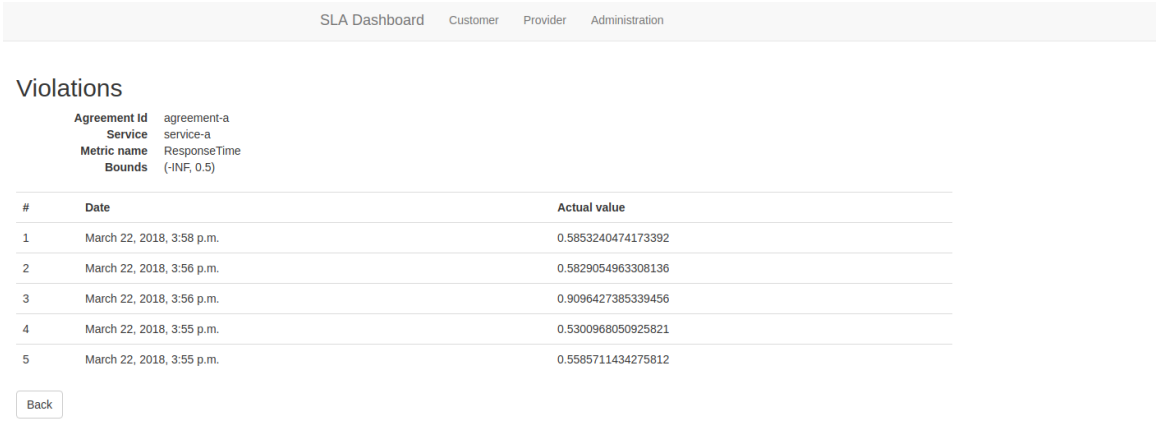


SLA Dashboard – Agreement status page



SLA Dashboard – Agreement detail page

© All Rights Reserved



SLA Dashboard – Violations page

4 Conclusions

The module has been adapted to interact with other BASMATI modules in order to:Receive customer requirements from the BASMATI platform.

- Work in conjunction with the BASMATI Service Description topology based on TOSCA specs.
- Support federated scenarios thanks to the specialization of the standard specifications used.
- Be able to inject incident signalling (instead of monitoring information) from both the infrastructure and the application layers.

In this accompanying report, we presented the main installation/usage guides of the BASMATI Federation SLA Manager module, which offers the mechanisms for ensuring quality in BASMATI, as well as the Quality of Services (QoS) policies approach for the management of federated cloud resources.

© All Rights Reserved

5 Annex

The component offers a REST API; other modules can request information to the module via GET method or create new entities within the module via POST method. This version of the deliverable adds the POST /violations REST method to the interface.
Find below a more detailed description of the interfaces provided by the module:

AGREEMENTS
GET /agreements{?providerId,consumerId,active} **Examples:** curl http://localhost:8080/sla-service/agreements curl http://localhost:8080/sla-service/agreements?consumerId=user-12
GET /agreemets/{id} **Example:** curl http://localhost:8080/sla-service/agreements/agreement04
GET /agreements/{agreementId}/guaranteestatus **Example:** curl -H "Content-type: application/xml" http://localhost:8080/sla-service/agreements/{agreementId}/guaranteestatus
POST /agreements **Example:** curl -H "Content-type: application/xml" -d@agreement02.xml http://localhost:8080/sla-service/agreements -X POST
DELETE /agreements/{agreement_id} **Example:** curl -X DELETE http://localhost:8080/sla-service/agreements/agreement04
ENFORCEMENTS
GET /enforcements **Example:** curl http://localhost:8080/sla-service/enforcements
GET /enforcements/{agreementId} Example: curl http://localhost:8080/sla-service/enforcements/agreement04
PUT /enforcements/{agreementId}/start **Example:** curl -X PUT http://localhost:8080/sla-service/enforcements/e3bc4f6a-5f58-453b-9f59-ac3eeaee45b2/start
PUT /enforcements/{agreementId}/stop **Example:** curl -X PUT http://localhost:8080/sla-service/enforcements/e3bc4f6a-5f58-453b-9f59-ac3eeaee45b2/stop
POST /enforcements **Example:** curl -H "Content-type: application/xml" -X POST -d @enforcement.xml

© All Rights Reserved

http://localhost:8080/sla-service/enforcements

PROVIDERS

GET /providers

Example:

curl http://localhost:8080/sla-service/providers

GET /providers/{uuid}

Example:

curl http://localhost:8080/sla-service/providers/fc923960-03fe-41eb-8a21-a56709f9370f

POST /providers

Example:

curl -H "Content-type: application/xml" -X POST –d @provider.xml http://localhost:8080/sla-service/providers

DELETE /providers/{provider_id}

Example:

curl -X DELETE http://localhost:8080/sla-service/provider/fc923960-03fe-41eb-8a21-a56709f9370f

TEMPLATES

GET /templates

Example:

curl http://localhost:8080/sla-service/templates

curl http://localhost:8080/sla-service/templates?serviceIds=service02

curl http://localhost:8080/sla-service/templates?serviceIds=service02,service03

GET /templates/{template_id}

Example:

curl http://localhost:8080/sla-service/templates/contract-template-2007-12-04

PUT /templates

Example:

curl -H "Content-type: application/xml" -X PUT -d @template01.xml http://localhost:8080/sla-service/templates

POST /templates

Example:

curl -H "Content-type: application/xml" -X POST -d @template01.xml http://localhost:8080/sla-service/templates

DELETE /templates/{template_id}

Example:

Curl –X DELETE http:/localhost:8080/sla-service/templates/contract-template-2007-12-04

VIOLATIONS

GET /violations

Example:

curl http://localhost:8080/sla-service/violations

GET /violations/{violation_id}

© All Rights Reserved

Example:

curl http://localhost:8080/sla-service/violations/violation94

GET /violations/{?agreementId,guaranteeTerm,providerId,begin,end}

Example:

curl -H "Accept: application/xml" http://localhost:8080/sla-service/violations/?agrementId=agreement04&guaranteeTerm=gt_uptime

curl "Accept: application/json" http://localhost:8080/sla-service/violations/?providerId=agreement04&begin=2014-03-18T15:23:00

PENALTIES

GET /penalties/{?agreementId,guaranteeTerm,providerId,begin,end}

Example:

curl -H "Accept: application/xml" http://localhost:8080/sla-service/penalties/?agrementId=agreement04&guaranteeTerm=gt_uptime

curl "Accept: application/json" http://localhost:8080/sla-service/penalties/?providerId=agreement04&begin=2014-03-18T15:23:00