HORIZON2020 FRAMEWORK PROGRAMME

TOPIC EUK-03-2016

“Federated Cloud Resource Brokerage for Mobile Cloud Services”

D4.1

Dynamic Cloud Federation: Design and Specification

Project acronym: BASMATI

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

Contract no.: 723131

<table>
<thead>
<tr>
<th>Work package:</th>
<th>Project category:</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Dynamic Brokerage and Federation: Strategies and Logic</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Work package:</th>
<th>Project category:</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Dynamic Brokerage and Federation: Strategies and Logic</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Editor:</th>
<th>Authors:</th>
<th>Work package:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jörn ALTMANN</td>
<td>Iain James MARSHALL</td>
<td>4</td>
</tr>
<tr>
<td>Jörn ALTMANN</td>
<td>Vaggelis PSOMAKELIS</td>
<td>4</td>
</tr>
<tr>
<td>Vaggelis PSOMAKELIS</td>
<td>Jörn ALTMANN</td>
<td>4</td>
</tr>
<tr>
<td>Enric Pere PAGES MONTANER</td>
<td>Netsanet HAILE</td>
<td>4</td>
</tr>
<tr>
<td>Netsanet HAILE</td>
<td>Ram Govinda ARYAL</td>
<td>4</td>
</tr>
<tr>
<td>Ram Govinda ARYAL</td>
<td>Juan Pablo ROMERO CORONADO</td>
<td>4</td>
</tr>
</tbody>
</table>

Authorized by: K. Tserpes

Reviewer: K. Tserpes

Dissemination Level: Public
## Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
<th>Author/Affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>24/10/2016</td>
<td>ToC</td>
<td>Iain James Marshall / AMENESIK</td>
</tr>
<tr>
<td>03</td>
<td>28/02/2017</td>
<td>ToC and Architecture</td>
<td>Jörn Altmann (SNU)</td>
</tr>
<tr>
<td>06</td>
<td>22/03/2017</td>
<td>Assignment of Chapters to Partners started</td>
<td>Jörn Altmann (SNU)</td>
</tr>
<tr>
<td>18</td>
<td>29/01/2018</td>
<td>Additional Input for Section 3.2</td>
<td>Iain James Marshall</td>
</tr>
<tr>
<td>18</td>
<td>29/01/2018</td>
<td>Additional Input for Section 3.3</td>
<td>Vaggelis Psomakelis</td>
</tr>
<tr>
<td>23</td>
<td>31/01/2018</td>
<td>Integration and Revision of Content</td>
<td>Jörn Altmann (SNU)</td>
</tr>
<tr>
<td>44</td>
<td>06/02/2018</td>
<td>Final revision</td>
<td>Jörn Altmann (SNU)</td>
</tr>
</tbody>
</table>
## BASMATI Glossary

<table>
<thead>
<tr>
<th>Term/Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile cloud services</td>
<td>Online services offered by cloud resources to support mobile apps. The backend of the mobile apps.</td>
</tr>
<tr>
<td>CP</td>
<td>Cloud Provider. The actor that provides the cloud infrastructure/resources, such as VMs.</td>
</tr>
<tr>
<td>CSP</td>
<td>Cloud Service Provider. The actor that provides cloud services on top of a rent infrastructure from a CP.</td>
</tr>
<tr>
<td>Cloudlet</td>
<td>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</td>
</tr>
<tr>
<td>Edge resources</td>
<td>Resources aimed to operate specialized functionality, located at the &quot;edge&quot; of the network infrastructure, thus, closer to the end users. Examples are (clusters of) RaspberryPis or cloudlets</td>
</tr>
<tr>
<td>BUDaMaF</td>
<td>BASMATI Unified Data Management Framework</td>
</tr>
<tr>
<td>KE</td>
<td>Knowledge Extractor</td>
</tr>
<tr>
<td>DM</td>
<td>Decision Maker</td>
</tr>
<tr>
<td>RB</td>
<td>Resource Broker</td>
</tr>
<tr>
<td>MVD</td>
<td>Mobile Virtual Desktop</td>
</tr>
<tr>
<td>DASFEST</td>
<td>An 3-day long music festival taking place in Karlsruhe, Germany every July</td>
</tr>
<tr>
<td>ACE</td>
<td>Amenesik Cloud Engine. The cloud service deployment tool through which actual federation is achieved</td>
</tr>
<tr>
<td>BEAM</td>
<td>BASMATI Enhanced Application Model. An extension of the TOSCA specification</td>
</tr>
<tr>
<td>ASP</td>
<td>Application Service Provider. A Federation user that rents resource services in order to provide an Application services to End-users</td>
</tr>
<tr>
<td>Brokering</td>
<td>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.</td>
</tr>
<tr>
<td>End user</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Offloading</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>QoE</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Service handover</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Situational Awareness</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>
Executive Summary

This report addresses the design and deployment of components that are related to the federation of clouds. In particular, it presents the requirements of different components related to cloud federation, i.e. the functional requirements of the federation management, the interoperability requirements, and the data management requirements. Based on these requirements, the design and the implementation of the components are presented. This includes the cloud federation management, the federation business logic, and the federation data management. Finally, this concludes with a brief overview about the advancement beyond the state of the art.
# Table of Contents

1. Introduction ................................................................................................................. 1
   1.1 Overview on Cloud Federation .................................................................................. 1
   1.2 Purpose of the Deliverable ....................................................................................... 1
   1.3 Relation to other Deliverables .................................................................................. 2
   1.4 Outline of Deliverable .............................................................................................. 2

2. Requirements of a Dynamic Cloud Federation ............................................................ 3
   2.1 Cloud Provider Management .................................................................................... 3
   2.2 Federation Business Logic ....................................................................................... 3
      2.2.1 Federation Service Level Agreements ................................................................. 3
      2.2.2 Revenue Sharing Mechanisms .......................................................................... 4
      2.2.3 Cross-Cloud Interoperability .......................................................................... 5
   2.3 Federation Data Management ................................................................................... 6
      2.3.1 Cloud Data Governance Requirements ............................................................... 7
      2.3.2 General Data Governance Challenges ............................................................... 7
         2.3.2.1 Data Availability ......................................................................................... 7
         2.3.2.2 Data Validity .............................................................................................. 7
         2.3.2.3 Data Security ............................................................................................. 7

3. Federation Components: Structure and Functionalities ............................................... 8
   3.1 Cloud Management Platform ................................................................................... 8
      3.1.1 Functionality of the Cloud Management Platform ............................................... 8
      3.1.2 Architecture of the Cloud Management Platform ............................................... 9
         3.1.2.1 Application Controller ............................................................................... 9
         3.1.2.2 Federated Cloud Management .................................................................. 10
         3.1.2.3 Edge Provider Management ..................................................................... 11
         3.1.2.4 Cloud Providers ........................................................................................ 11
      3.1.3 Interfaces and Workflow of the Cloud Management Platform .......................... 11
   3.2 Federation Business Logic ....................................................................................... 12
      3.2.1 Architecture of the Federation Business Logic and its Interaction .................... 12
      3.2.2 Federation SLA Manager ................................................................................ 13
         3.2.2.1 Functionality of the Federation SLA Manager .............................................. 13
3.2.2.2 Architecture of the Federation SLA Manager ............................................... 17
3.2.2.3 Interfaces and Workflow of the Federation SLA Manager ........................................ 17
3.2.3 Cost and Revenue Sharing ......................................................................................... 17
3.2.3.1 Functionality of the Cost and Revenue Sharing ......................................................... 17
3.2.3.2 Architecture, Interfaces, and Workflow of the Cost and Revenue Sharing .......... 20
3.2.4 Invoicing and Accounting ......................................................................................... 20
3.2.4.1 Functionality of the Invoicing and Accounting ............................................................... 20
3.2.4.2 Workflow of the Invoicing and Accounting ................................................................. 21
3.2.5 Cross-Cloud Interoperability .................................................................................... 21
3.2.5.1 Functionality of the Cross-Cloud Interoperability ...................................................... 21
3.2.5.2 Architecture of the Cross-Cloud Interoperability Component ................................ 23
3.2.5.3 Interfaces and Workflow of the Cross-Cloud Interoperability Component ........ 26
3.3 Federation Data Management ................................................................................. 27
3.3.1 Functionality of the Federation Data Management .................................................. 27
3.3.2 Architecture of the FDM ......................................................................................... 28
3.3.2.1 Core ............................................................................................................................ 29
3.3.2.2 Offloading APIs ......................................................................................................... 30
3.3.2.3 Analytics Engine ...................................................................................................... 30
3.3.2.4 Security and Anonymization Engine ....................................................................... 30
3.3.3 Workflow .................................................................................................................. 30
4 Advancement beyond the State of the Art ..................................................................... 33
4.1 Further Developments for BASMATI ........................................................................ 33
4.2 Challenges and Highlights ............................................................................................ 33
5 References ...................................................................................................................... 35
1 Introduction

1.1 Overview on Cloud Federation

Besides market places [33] [34], cloud federation has been seen as a possible solution for the volatility in the number of user requests and for the anti-competitive externalities of the economies of scale in the cloud service sector [19] [37]. Horizontally dynamic cloud federation allows small cloud providers to collaborate and gain access to economies of scale by increasing the amount of infrastructure resources available to them [25]. It also helps ensuring the users’ quality of service and minimizing costs [43].

Extensive research has been done on optimizing the performance of certain federations and on dealing with challenges, such as resource sharing and interoperability [17] [21] [36]. Factors hindering providers to adopt cloud federation have also been investigated [18] [31]. Despite the promises of cloud federation, it is important to state that there is no functional federation available in the commercial market.

After a thorough review of the cloud federation literature [23] [40] [41] [43], several factors were identified as important for incentivizing federations and coalitions. Amongst them, the concept of capacity-based revenue sharing is perhaps the most prominent one. It is resource and revenue sharing mechanisms, which determine how cloud providers in a federation share their computational resources, and more importantly, the profits that result from the collaboration. Having an efficient mechanism is of paramount importance since it would encourage cloud providers to participate in a federation [39].

1.2 Purpose of the Deliverable

This deliverable document deals with the design and specifications of the infrastructure required for the formation and management of a dynamic cloud federation. Dynamism in this context entails the ability to automatically form and dissolve federations, to self-adapt to maintain resource compatibility and policies, and to achieve real-time situational data management. This calls for tools that can manage contractual and economic relationships between members, considering the current situation. To this effect the following components of the BASMATI application are selected and discussed in the subsequent sections:

- Cloud Provider Management Platform
- Federation Level Agreements as part of the Federation Business Logic
- Cost and Revenue Sharing schemes as part of the Federation Business Logic
- Cross-Cloud Interoperability as part of the Federation Business Logic
- Federation Level Data Management
1.3 Relation to other Deliverables

This deliverable, deliverable D4.1, is based on two previous deliverables. On the one hand, it relies on the findings of deliverable D2.1, the State-of-the-Art and Requirement Analysis document. It provides details about the current technologies of cloud federations, the available economic models, and the challenges facing the development and acceptance of cloud federations as a viable business model. On the other hand, deliverable D4.1 uses the findings of deliverable D2.3, which provided the overall architecture of the BASMATI platform and the initial concepts and structures of the BASMATI platform components [20] [22].

![Diagram](image)

Figure 1. Deliverable D4.1 and its dependencies with other deliverables

As shown in Figure 1, deliverable D4.1 has two-way relationships with D5.1, D5.3, and D5.5. These are the deliverables, which deal with the specific implementation strategies (algorithms) of the components (i.e., federation data management (D5.1), component management (D5.3), and service level agreements (D5.5)) that are parts of the dynamic cloud federation infrastructure.

1.4 Outline of Deliverable

This deliverable is organized as follows. Section 2 identifies the sub-components of a dynamic cloud federation and presents design objectives for each sub-component. Section 3 deals with the functionalities and detailed structures. Specifically, Section 3.1 presents the Cloud Provider Management, Section 3.2 presents the Federation Business Logic, and the Federation Data Management is detailed in Section 3.3. In particular, Section 3.2 details the design of the Federation Business Logic component. This includes federation SLA management, in terms of various federation topologies, cost and revenue sharing mechanisms, invoicing and accounting, and cross-cloud interoperability. Section 3.3 details the architecture and the workflow of the Federation Data Management. Section 4 highlights the advancements of these components, including further developments and challenges.
2 Requirements of a Dynamic Cloud Federation

2.1 Cloud Provider Management

The requirements for the Cloud Provider Management component, the central component of the BASMATI architecture [20] [22], are derived from general requirements of software engineering. In particular, the Cloud Provider Management component has to be robust towards:

- External erroneous requests,
- Failure of BASMATI components, and
- General system failure.

Since the Cloud Provider Management component is the central component, through which all communication between other BASMATI components take place, reliability and availability are of utmost importance. Even if the Cloud Provider Management component crashes due to external influence, the Cloud Provider should be able to restart from a well-defined state.

2.2 Federation Business Logic

2.2.1 Federation Service Level Agreements

The following requirements are to be respected, when composing XML Federation Agreement documents in the absence of an SLA document preparation assistant:

- The value of the initiator attribute of the parenting agreement element MUST be the name of the platform operator that will load the agreement to define an offer of the federated service.
- The value of the responder attribute of the parenting agreement element MUST be the name of the platform operator that issued the agreement defining the offer of service.
- The value of the service provider attribute of the parenting agreement element MUST specify the value of “responder”.
- The federation element MUST specify values for the user, password, publisher and security attributes.
- The agreement attribute of the federation element MUST be the same as the value of the name attribute of the parenting agreement element.
- The value of the operator attribute of the provider and price elements MUST be the same as the value of the responder attribute of the parenting agreement element.
- The provider element MUST specify values for the category and zone attributes.
- The value of the zone attribute of the provider element MAY be “any”, indicating that the underlying provisioning interface will be capable of, and responsible for, multi-zone placement.
• Each quota element of all provider elements MUST specify values for the property, value and condition attributes.
• Price elements MAY be defined for the quota elements of the provider elements of the federation element.
• Condition and guarantee term elements MAY be specified for the agreement but no semantic is currently defined.
• Contract management is performed on the requesting platform of the federation member.
• Provider quota management is performed on the responding platform of the federation member.

2.2.2 Revenue Sharing Mechanisms
Revenue sharing is the distribution of profits and costs between stakeholders of a business or an organization. Although it is an existing concept, it has been transformed and popularized due to the introduction of platform-based content provisioning over the Internet [24] [26]. The content can comprise, for example, applications, advertisements, music, and videos.

These revenue sharing mechanisms (i.e., business models) need to be investigated to become the basis for the implementation of cloud federations [13] [14]. A successful revenue sharing mechanism is the requirement for cloud providers to cooperate. That means:

• The revenue sharing mechanism must ensure that they are able to perceive a benefit from their participation in a federation.
• Moreover, a fair system is needed, ensuring that all cloud providers are properly recompensed for the amount of resources that they invest into the federation.

The objectives of these mechanisms are:

• To maximize profit for the entire federation, every member of the federation obtains a profit higher than it would have achieved without joining the federation. Policies of fairness can be incorporated to ensure this [13].
• To share risk (cost) among the federation partners

The revenue sharing part of the federation agreement (contract) needs to transparent and include information about:

• Name of sharing parties
• Amount of resources provided per period
• Sharing scheme with functions for calculating cost and revenue
• Sharing period, such as the expiration of the federation agreement
In addition to this, for specific revenue sharing schemes, there are conditions that need to be fulfilled with respect to pricing and distributions of shares [28] [29] [30] [32]. These conditions are:

- Percentage of revenue to be allocated to each member of the federation
- Price of resources to be set.

2.2.3 Cross-Cloud Interoperability

There are many examples that show the lack of interoperability between cloud providers. For example, Microsoft .NET application containers and Microsoft Azure database services cannot be integrated with Google App Engine [1] [2]. The technologies with potential issues of interoperability can be classified into programming frameworks, application programming interfaces, and data formats [3] [17].

Programming Frameworks: Since service developers are attached to their software development tools, programming languages, and runtime environments, they tend to avoid changing their programming frameworks. Thus, cloud providers need to support the multitude of different programming frameworks that are in use today. Any change of programming framework would come with the cost of learning a new technology for service developers [4] [5]. Cloud providers would also face a cost, if they would have to support different software development tools.

Application Programming Interface: Many cloud platform providers have their own application programming interface (API) [27], complicating the process of orchestration of applications across multiple service platforms of other providers. The incompatibilities between the APIs for uploading, downloading, inspecting, and configuring are significant issues for interoperability. For example, the Amazon EC2 API is different from the GoGrid (Datapipe) API [6] [7], although both offer similar IaaS services. In addition to this, cloud providers offer their own proprietary services. An example of valuable but proprietary services is Amazon Elastic Load Balancing [8]. This service only exists on a specific platform. Standardization of APIs would allow applications to be moved to different providers without any additional integration or switching costs. Another way would be the provisioning of programming toolsets, which enable their services to be deployable on multiple platforms [9] [10]. Although this kind of middleware could break the dependencies on specific APIs, it is only useful for new application developments [11]. Similarly, the format of virtual machines (VM) is one of the major challenges for creating seamless portability among cloud platforms. As platform customers have already different virtualization environments (e.g., VMware, Hyper-V, KVM, and Xen) in their in-house data centres, cloud platform providers should make sure that these customers can use the one that they are familiar with [11].

Data Format: Many application services provide their own semantics and format for storing data [4], making it costly to access the data from a different application. If no standard for the data format exists, conversion tools are needed to translate the data format of one application
service into the data format of another application service [12]. This would increase the cost of

![Image of Interoperability Diagram]

the cloud provider or the data owner.

Figure 2. Interoperability aspects between cloud platform providers

On the one hand, interoperability between technologies provides a reduction in switching cost, learning new technologies, and integration. On the other hand, interoperability reduces the innovation potential in the market.

As major innovations and technological developments result in an increase in revenue and reduction in interoperability, the challenge is to find a compromise between cost for making legacy systems interoperable and the overall revenue that can be made. Due to this situation, cloud providers need to decide on the investments in interoperability with other cloud providers, enabling them to join a federation of cloud providers.

Joining a cloud federation requires that the cost for achieving interoperability with other providers is smaller than the potential additional revenue from joining the federation. For this, a cloud provider needs to have access to a tool, as part of the Federation Business Logic component, which allows the provider to make an informed decision.

2.3 Federation Data Management

BASMATI will be using a plethora of IoT and traditional devices in the form of a multi-cloud architecture. Each type of device gathers or creates data in a format that is not always compatible with each one of the other devices. Of course, in order to form a cloud, or a cloud of devices, a common data structure and a common data storage must be defined, so that the devices that form this cloud can exchange information freely. BASMATI takes that one step
further by forming a federation of clouds. This federation of clouds requires a common data management solution, in order to function in a unified way, as one federated cloud.

The requirements that will be described shortly can be separated into two main categories, the cloud data governance requirements and the general data governance requirements. The first category consists of requirements that arise mainly in big, distributed data stores or cloud data stores. The second category is formed by requirements, which arise in every data store, regardless of size and location.

2.3.1 Cloud Data Governance Requirements
Every system must have a place for information exchange. Even in a classical federation, that of human beings, there is a senate, in which the members of the federation gather to exchange information and derive conclusions. In our case, this senate will not be a building. It has to be software that unifies all the individual databases for communicating in a unified way, transparent to the BASMATI services. In our case, this has to be the form of a middle-ware that handles the inter-component communications, translating all requests to a common language, using Web services and publicized APIs.

2.3.2 General Data Governance Challenges

2.3.2.1 Data Availability
The vision of BASMATI is that end users positively experience all provided services with no knowledge of the infrastructure limitations that their application encountered and overcame. In order to achieve this, we have to ensure data availability, either on the local cloud that is owned and managed by each application or in the cloud federation. This exposes the requirement of scaling and data redundancy mechanisms, in order to achieve high data availability, even under extreme cases.

2.3.2.2 Data Validity
Data validation is another important part of working with such a vast amount of data. Due to the heterogeneity of the data and the databases, a unified way of data validation would be hard to implement. It would be more efficient for each local database, both during the development stage and during the usage stage, to validate its own data, especially if the data modifications are only done by local devices.

2.3.2.3 Data Security
The BASMATI use cases are targeting end users and devices that contain a number of personal data. This data should not be available to anyone except the authorized applications and services, which are to be granted the permission, by the end user, to access and use them. In order to prevent both accidental and malevolent data leaks, we need to consider a data security scheme that will protect the data during their storage, their retrieval and their modification.
3 Federation Components: Structure and Functionalities

3.1 Cloud Management Platform
This section of this document describes the way, in which cloud providers, whether public, private, edge or federated, are managed within the Amenesik Cloud Engine component of the BASMATI platform.

3.1.1 Functionality of the Cloud Management Platform
The Cloud Management Platform is responsible for providing a cloud application description and deployment abstraction layer for the realization of resource deployment not only on existing public cloud providers but also through the Edge Provider Management component, that is nowadays referred to as Fog Computing, and through the BASMATI Federated Cloud Management.

Customer’s cloud applications and their requirements are introduced to the platform as BEAM/TOSCA documents, describing not only the technical characteristics and topology of the required configuration but also the service level objectives and constraints that are to be ensured and imposed. These documents are processed by the cloud management platform, when requested by the Application Controller and result in a collection of alternative service state references being returned upon successful completion. The creation of a service state instance is performed by the generic service contract broker whereby contract negotiation is first attempted through any of the appropriate cloud provider interfaces, public, private and edge, local to the operator platform. If a successful local contract negotiation is not achieved, then the contract negotiation will be attempted through each of the appropriate federation members known to the local platform operator. Since the contract negotiation, or placement, service of each of the interacting federation members is based on the identical basmati federation concepts and components, then this negotiation phase may be seamlessly propagated, in a fully automated manner, across each of the federation links, between the various members of the federation, until either satisfactory placement of the required resources, is achieved, or ultimate failure ensues.

Service instances, which result from the successful completion of this service contract negotiation and placement phase, are subsequently eligible candidates for deployment, by invocation of the corresponding service “start” action. This action request, received by a service instance, will be forwarded to each of the contracts that were negotiated through the various cloud provider interfaces. This applies equally to contracts that were returned by the remote placement operations and those that were performed by remote federation members. The “start” action will be performed by the remote federation member platform, on behalf of the requesting platform operator. All transactions, which are incurred for resources that are required and consumed for the deployment of the corresponding contracts, will be attributed to the account of the requesting federation member and will enter into the subsequent
calculations of balance of payments that are required by the principles of cost and revenue sharing described in subsequent sections of this document.

3.1.2 Architecture of the Cloud Management Platform
The following diagram depicts the primary relationships between the Cloud Management Platform, the Application Controller, the Federated Cloud Management, the Edge Provider Management component, and the components handing the Cloud Providers. The numbered arrows in the diagram represent the primary sequence of API calls between the five components.

![Diagram of Cloud Management Platform architecture](image)

Figure 3. Interaction of the Cloud Management Platform with four other components

The functionalities of the four different components that are connected to the Cloud Management Platform are described in the following subsections.

3.1.2.1 Application Controller
The BASMATI Application Controller is responsible for the management and coordination of applications and their deployed and deployable Application States. Following the requirements specified in the Application Description, the Application Controller uses a collection of Application States, which allow resilient life cycle management of the Application State and its required resources.
An instance of the Application Controller is created for the combination of a service level agreement and technical manifest, resulting from or derived from the application description. Application State instances are created for each of the alternative states required in the application description. These alternative Application State instances allow a fault-tolerant or mobility-following behaviour of the Application Controller, to be achieved by switching between application states in response to specific conditions detected by life cycle state monitoring.

3.1.2.2 Federated Cloud Management

This abstraction layer, when employed by multiple, individual commercial cloud service providers, or operators, allows for automation of resource and revenue sharing between these providers, which collectively we refer to as a federation of cloud service providers, with each provider or operator being referred to as a federation member. Service level agreements (SLAs) are used to define and control the various commercial relationships between:

1. A customer and a cloud service provider (business to customer (b2c))
2. A cloud service provider and a cloud resource provider (business to business (b2b))
3. A member of the federation and another member of a federation (b2b)

The following figure depicts an example of how SLAs are used to control the three relationships. In particular, it shows how SLAs are used to control and manage resource sharing and revenue sharing between the federation members when they join a federation of cloud service providers.

Figure 4. SLA-based management of sharing between federation members
3.1.2.3 Edge Provider Management
The Edge Provider Management is responsible for the localization and exploitation of Edge Computing resources shortening the last mile to reach the actual mobile user. This component encapsulates multiple private cloud interface technologies to be able to use specialized data centres for certain application specific needs. The geo-placement algorithms allow selection of Edge Provider deployment of service components using application specific private infrastructure resources.

![Edge Provider Management Diagram]

**Figure 5.** Edge Provider Management

3.1.2.4 Cloud Providers
This component of the above diagram represents the major commercial cloud platforms, namely Amazon Web Services (AWS EC2 and ECS), Microsoft Windows Azure, Google Compute (GCE and GKE), IBM Soft Layer, Cloud Sigma and other secondary cloud providers such as RackSpace, OVH, HP, DELL, to name but a few. These commercial vendors offer Infrastructure as a Service, the foundation of modern cloud computing. Each provider publishes either a proprietary API or an adaptation of an Open API such as OpenStack, OpenNebula or Eucalyptus.

![Cloud Providers Diagram]

**Figure 6.** Examples of cloud providers with proprietary or with open APIs

3.1.3 Interfaces and Workflow of the Cloud Management Platform
The sequence of API exchanges between the various components, as shown in the primary component interaction diagram (Figure 3) and as required for the management of the Application State Life Cycle, are described in the following points:

1. The Application Controller requests the Cloud Management Platform to create an Application State Instance.
2. The Cloud Management Platform requests creation or preparation of resources through either the Cloud Providers, the Cloud Federation, or the Edge Provider Management depending on the localization of resources indicated in the API request 1.

3. The Application Controller requests the Cloud Management Platform to Start an Application State Instance.

4. The Cloud Management Platform requests deployment, installation and configuration of the resources required to fulfil the requirements of the Application State Instance through the Cloud Providers, the Cloud Federation, or through the Edge Provider Management.

5. The Application Controller requests the Cloud Management Platform to Stop an Application State Instance.

6. The Cloud Management Platform requests release of Application State instance resources by the Cloud Provider, the Cloud Federation, or the Edge Provider Management.

7. The Application Controller requests deletion of an Application State instance by the Cloud Management Platform.

8. The Cloud Management Platform deletes residual resources from the Cloud Providers, the Cloud Federation, or the Edge Provider Management.

The following diagram presents the fundamental conceptual elements of the application programming interface required for the management of the application service life cycle.

![Diagram of API conceptual elements](image)

**Figure 7.** Conceptual elements of the API for managing the application service life cycle

Each of these elements of the API will be represented by an individual OCCI category and its corresponding instance management endpoint. The characteristics and behaviour of the categories of this API are described in the deliverables related to the Application Controller.

### 3.2 Federation Business Logic

#### 3.2.1 Architecture of the Federation Business Logic and its Interaction

The following diagram shows the interactions of the Federation Business Logic Component and its sub-components, which are the Cost and Revenue Sharing Scheme component (Section 3.2.3), the Cross-Cloud Interoperability component (Section 3.2.5), the Federation SLA Manager component (Section 3.2.2), and the Application Provider Accounting and Invoicing
component (Section 3.2.4) with the Application Backend Management component and the Cloud Management Platform (Section 3.1).

In the following subsections, the sub-components of the Federation Business Logic, namely the Federation SLA Manager component, the Cost and Revenue Sharing Scheme component, the Invoicing and Accounting component, and the Cross-Cloud Interoperability component are described.

3.2.2 Federation SLA Manager

3.2.2.1 Functionality of the Federation SLA Manager
The Federation SLA Manager manages Federation Service Level Agreements. A Federation Service Level Agreement is a derivation, or specialisation, of the international standard known...
as WS-Agreement [35]. It describes a new service description element, which is used to describe the technical and commercial details of a mono- or bi-directional relationship between two federation members [31]. The resulting agreement, when introduced into either, or both, of the Basmati Ace platforms, of the partners to the agreement, guides the actions of the Federation SLA managers of those platforms. The actions comprise the automated management of the technical and commercial aspects of the subsequent mutual interactions between the partners. These technical and commercial aspects are:

- Availability of resources
- Price of resources
- Placement of resources
- Deployment of resources
- Billing of resources
- Cost and revenue sharing.

Each of these aspects is described in detail in deliverable D5.5 of the BASMATI project.

Note, that the term ‘resources’ is used equally to refer to the IaaS resources and the PaaS resources that are made available by the federation members for use within the terms established in their agreement.

The Federation SLA Manager has also to handle the construction and coordination of a variety of different cloud service federation configurations, which are enabled through the Federation Service Level Agreement. Seven of those configurations are.

**Simple Half-Duplex Configuration:** The simplest case is the case, where an application service provider is allowed access to the cloud provisioning capacity of a cloud service provider, for the delivery of application services to their customers. This configuration is referred to as the “simple half-duplex configuration”. An example is shown in the following figure.

![Simple Half-Duplex Configuration](image)

**Application Service Provider**  **Cloud Service Provider**  

Figure 9. An example of the simple half-duplex configuration

**Simple Full-Duplex Configuration:** The logical extension to the simple half-duplex configuration would see the two federation members making their surplus service capacity available to each other through complementary federation agreements as depicted below. This configuration can be referred to as the “simple full-duplex configuration”.

![Simple Full-Duplex Configuration](image)
**Duplex Chain Configuration:** The basic building block combinations (i.e., simple half-duplex configuration and the simple full-duplex configuration) can be chained together, to compose a linear federation configuration. In this linear federation configuration, each member of the federation is in relation with one or two other federation members. An example is depicted in the following figure. This configuration can be referred to as the “duplex chain configuration”.

In this configuration, the value of the placement algorithm property of the service level agreements between pairs of members would be used to control secondary federation visibility. The standard values of “quota:default” and “quota:federation” are to be used respectively to this effect. In the above example, if each agreement specified “quota:federation”, as the value of the placement algorithm, then requests for provisioning issued by the federation member A could eventually be satisfied by the federation member E, by being passed down across the chain by each of the intermediate members B, C, and D.

**Captive Configuration:** A more complex but probably more realistic configuration can be envisaged, if a federation management “member” will enter into individual federation agreements with application service providers and cloud provisioning providers. An example of this “captive configuration” is depicted in the following figure.

In this case, the Federation Management Member would be responsible for dispatching service requests to the individual federation members, A, B, ..., and E.

**Captive Duplex Chain Configuration:** An extension to the preceding configurations is the captive duplex chain configuration. In this configuration, federation members would be exposed to and
managed through a central authority (i.e., federation management member) for their introduction to the federation. The federation management member provides them with the equivalent of a federation resource catalogue, which would allow them to establish point-to-point operations between members as and when required. The following figure depicts this hybrid configuration.

**Multipoint Full-Duplex Configuration:** The “duplex chain” configuration can be further enhanced resulting in the case with all federation members being effectively connected to all other federation members. Member relationships are no longer the result of being chained together but resemble the spokes of a wheel. In this case, all members of the federation would enter into bi-lateral, full-duplex service level agreements with all other members. It can be referred to as the “multi-point full duplex configuration”. It should be noted that the total number of relationships and their accompanying agreements increases exponentially with the size of the federation. The following figure depicts a simple four-member example of this type of configuration.
**Captive Multipoint:** The preceding model (i.e., the multipoint full-duplex configuration) can be adapted to incorporate a central federation management member. It results in a configuration, which is referred to as the “captive multipoint configuration”, that allows the efficient collection of specific data and the efficient management of the members. An example is depicted in the following figure.

![Diagram of captive multipoint configuration](image)

**Figure 15.** An example of the captive multipoint configuration

### 3.2.2.2 Architecture of the Federation SLA Manager

Please refer to the deliverable D5.5 for details on the architecture of the Federation SLA Manager.

### 3.2.2.3 Interfaces and Workflow of the Federation SLA Manager

Please refer to the deliverable D5.5 for details on the workflow and the interface specifications of the Federation SLA Manager.

### 3.2.3 Cost and Revenue Sharing

#### 3.2.3.1 Functionality of the Cost and Revenue Sharing

In the case of a commercial cloud federation, with cloud service providers and application service providers working together in a cooperative manner for the collective provision of added
value application service to their collective customers, cost and revenue sharing must be clearly defined. It will be defined in the service level agreement, independent of the chosen federation configuration (Section 3.2.2).

Cost and revenue sharing mechanisms are important for cloud federations due to two factors. Firstly, cloud providers need an effective revenue sharing method, encouraging them to participate in a federation. That means, cloud providers will cooperate, if they receive a benefit [40].

Secondly, it determines how the allocation of revenue is performed. A fair system is needed, ensuring that all cloud providers are properly compensated for the amount of resources that they invested into the federation [39]. For this study, fairness is defined as self-centered inequity aversion. This term relates to the behaviour, at which “people resist inequitable outcomes; i.e., they are willing to give up some material payoff to move in the direction of more equitable outcomes” [41].

There are several well-known mechanisms for cost and resource sharing in game theory models. However, each one of them provides a different benefit, fairness, and stability values to the collaborations. This may affect how the federations are created, and even how they are dissolved. In the following, we introduce 3 different mechanisms of cost and revenue sharing: Assigned Resources Mechanism, Outsourcing Mechanism, and Shapley Value Mechanism [14].

**Cost and Revenue Sharing Mechanisms**

With the Assigned Resources Mechanism, each cloud provider will obtain a revenue share that is proportional to the amount of resources contributed (proportional revenue sharing mechanism) [39]. This mechanism is particularly strong in its fairness. This is a simple mechanism to implement, as it only considers the resource contributions of collaborating cloud providers for the calculation of the amount of revenue that each one of them gets. Besides, it allows combining resources that could not be sold separately [14].

With respect to the Outsourcing Mechanism, cloud federations have also been seen as a way for cloud providers to outsource some of their business to other cloud providers. Following this logic, collaborating cloud providers can implement a mechanism, by which the outsourcing provider will get a percentage of the revenue, or a fixed fee. This revenue sharing would allow a cloud provider to keep some of the revenue of the business it secured, even though it would not be able to fulfil it alone. For this mechanism, the variable alpha is defined as the percentage of revenue that will be forwarded by the outsourcing cloud provider to the service-fulfilling cloud provider [14].

The Shapley Value Mechanism is named after Lloyd Shapley, who proposed a method to calculate the overall gain of all alternatives of a player that participates in a game with a large number of agents. The Shapley Value is calculated through:
\[
\varphi_i(v) = \sum_{S \subseteq N} \left(\frac{(|S| - 1)!}{|N|!}\right) \left(\frac{(|N| - |S|)!}{|N|!}\right) \left[v(S) - v(S - \{i\})\right]
\]

where \( \varphi_i \) represents the Shapley value, which is the gain of player \( i \). \( S \) represents a possible coalition, which is a subset of all players \( N \). The function \( v : 2N \rightarrow \mathbb{R} \) represents the worth of a coalition [42].

In cloud computing, the Shapley value is used to represent the marginal contributions of any cloud provider to the federation it belongs to. In contrast with other revenue sharing schemes, this scheme allows federations to allocate revenue according to the value created. Using this scheme, other types of contributions made by cloud providers can be considered, not just the resources provided by the cloud providers [14].

**Conditions**

Independent of the cost and revenue sharing mechanism implemented, it has to be considered that federation members, who bring consumable resources to the federation, such as virtual machines, disk space, network bandwidth, IP addresses, application licenses, incur costs for the resources that they provide. This can be either the result of their subscription accounts with public cloud providers such as AMAZON, GOOGLE, MICROSOFT, or the result of the day to day operational costs of running their own data centre. The cost includes power consumption, operation cost, and maintenance cost. The cost for running the same type of virtual machine may be different for different service providers depending on the geographical location as well as security and other provisions implemented by individual CSP [19]. The cost may also be affected by dynamic factors, like the current workload at the cloud data center. A cloud provider, that is not willing to accept many requests from the federation due to an internal high workload, may assign a prohibitively higher price than the actual cost for the resource [13]. In any case, these federation members will be responsible for covering the cost or paying the invoices received from their suppliers for the provision of their basic services.

In either case, it is normal to expect that the federation member providing these resources to the federation, for use by other federation members, be reimbursed at least at cost value and to some degree of financial gain.

Therefore, the federation service level agreement, through which a cloud provider offers its resources through the federation, for the composition and operation of service, describes the price of the corresponding offer of resources. Any service provider, consuming resources made available through the federation, are required to make payment to the corresponding federation members providing these resources, on presentation of the relevant invoicing.
Federation Configuration Structure

The federation configuration structure (Section 3.2.2) has an impact on the implementation of the mechanisms, conditions, and the workflow mentioned above. Therefore, we must distinguish between a decentralized federation configuration structure and a central federation configuration structure.

In a decentralized federation configuration structure (a federation without a central federation management member), the implementation of the cost and revenue sharing scheme incurs a low cost of communication between all cloud federation members that have established a service level agreement. The cost of communication increases exponentially, if the number of federation members increases linearly.

In a centralized federation configuration structure, the implementation of the cost and revenue sharing scheme incurs low cost of communication, as all members of the federation have to communicate their input only to the central entity (i.e., the federation management member). The cost of communication increases only linearly, if the number of federation members increases linearly.

Fairness of Resource Utilization

In a federation, the members of the federation should achieve nearly the same increase in additional utilization and profit gain over an appropriately long period of time. This would guarantee that not only a few cloud providers utilize their resources fully, while other cloud providers are not attributed tasks, leaving them at the same level of utilization as before joining the cloud federation. In this case, joining a cloud federation would only cause additional management cost.

3.2.3.2 Architecture, Interfaces, and Workflow of the Cost and Revenue Sharing

The architecture of the cost and revenue sharing is simple. It comprises a formula (i.e., a pricing scheme) that is expressed within the federation service level agreement, using WS-Agreement [35]. This formula will be read by the Invoicing and Accounting component, to calculate the final charge and balance of payments. Consequently, there is no specific interface or workflow for the Cost and Revenue Sharing. Instead, the Invoicing and Accounting component would have to obtain the required input for the formula and calculate the corresponding charges based on the formula.

3.2.4 Invoicing and Accounting

3.2.4.1 Functionality of the Invoicing and Accounting

All actions performed by cloud providers and application service providers alike, for which an element of cost has been defined, will result in a financial transaction being debited and
credited to the to the accounts of the involved parties, for the amount described in the terms of the service level agreement.

Invoice processing, often referred to as transaction collation, is performed on an account by account basis. It is performed by the accounting service of each platform operator. The resulting invoices are issued to the customers and the consumers of service, whether external or internal to the federation. All customers are liable for payment.

Due to the distributed and fully automated nature of the BASMATI federation and the cloud abstraction technology provided by the AMENESIK Cloud Engine, it can be envisaged that certain members of the federation could specialise in the management of accounting, invoicing, and cost and revenue sharing [38].

With respect to federation members, who provide application services to customers and their end users, they will invoice their customers for the services that they provide. This revenue stream will be negotiated and decided between the customer and the application service provider and will be clearly expressed in the terms of the service level agreement between the customer and the application service provider. For this, the charges can either be simply calculated based on the data collected for a specific customer or require the collection of accounting data from other members of the federation.

3.2.4.2 Workflow of the Invoicing and Accounting
The Invoicing and Accounting component requires the following steps to be executed, in order to calculate the final charge for each member of the federation:

- Obtaining the cost of operation (service provisioning), which depends on the cloud provider, to which a partial job of customer job has been assigned
- Calculation of the total revenue from a job
- Finding the cost and revenue sharing mechanism used during service consumption
- Calculation of the revenue for each federation member.

3.2.5 Cross-Cloud Interoperability

3.2.5.1 Functionality of the Cross-Cloud Interoperability
The objective of the Cross-Cloud component is to validate the impact of high-level interoperability and portability on the value creation of cloud computing platforms, which is needed for cloud federations. This objective translates into two major research questions: (1) How does an increase in the level of interoperability and portability relate to the change in the value of platform providers and their users? (2) How do investments in interoperability and portability affect profit, net present value, return on investment, and discounted return on investment?
In order to tackle the two research questions, we develop a value creation model for cloud computing platforms, which shows the stakeholders and the value exchange between the stakeholders, based on the work by Gebregiorgis and Altmann [3]. Using this value creation model and system dynamics methodology, first, we analyze the relationship between the level of interoperability and portability and the value obtained by providers and users. Second, we extend the model for the analysis of the impact of investments in interoperability and portability. For the analysis, we can calculate profit, return on investment, net present value, and discounted return on investment.

The value creation model designed here for describing the impact of interoperability of cloud services on the value of cloud customers and the revenue of cloud providers can show the dynamics between two stakeholders: the end-users and the cloud provider. Application service developers are not explicitly modelled in this value creation model. It is implicitly assumed that the number of application service providers corresponds to the number of services offered on the platform. An end-user (user) can decide whether to join a cloud computing platform, depending on the net benefit obtained from the platform. The user’s decision is based on whether compatible services (functionalities) are available on the platform. Furthermore, a user’s decision also depends on whether it is possible to connect one platform service to another platform service, requiring interoperability between the cloud providers. For example, a user might want to interconnect services running on its private cloud to services on a public cloud.

This value creation model with its factors and relationships is shown in the following figure. Each of the relationships is supported through theories or observations made in earlier research, such as the marginal utility theory of Gompertz [15] and the relationship between quality of service (i.e., functional QoS (e.g., development of new services) and non-functional QoS (e.g., interoperability, portability, usability) and financial accountability [16].
Figure 16. Value creation model considering the level of interoperability

In general, the provider revenue is increased, if new users are attracted to the platform as a result of their positive valuation of the platform. The factors that impact the cloud provider’s revenue are:

- A user’s positive valuation corresponds to their net value being larger than zero.
- A user’s perception of the platform value is impacted by its level of interoperability and portability (e.g., level of customer lock-in, level of accessibility of compatible services).

For details on the value creation model, a published scientific article, which has been funded by the BASMATI project, is available from Haile and Altmann [17].

3.2.5.2 Architecture of the Cross-Cloud Interoperability Component

The Cross-Cloud Interoperability component is a tool based on the tool Vensim [48], which allows simulating system dynamics models. It comprises three parts: the model parameter setting part, the output specification part, and the data visualization part. Vensim supports the development of all these three parts.

The model parameter setting part of the Cross-Cloud Interoperability component:

- Provides graphic representation of the model
- Enables flexibility and adaptability of models as the tool user needs, e.g., change in service price, amount of investment
• Allows setting the functional relationships between any two factors in the model.

An example to the model parameter setting part for the Cross-Cloud Interoperability component is shown in the following figure.

Figure 17. Model parameter setting in the Cross-Cloud Interoperability component

After a simulation has run for a certain model parameter setting, the tool (i.e., the output specification part) enables the tool user to compile information from raw data produced by the simulations. In detail, the tool user can

• Specify the way the raw data should be presented, by selecting specific variables
• Determine combinations datasets of raw data and variables can be chosen
• Store data for later use.
An example to the output specification part for the Cross-Cloud Interoperability component is shown in the following figure.

![Cross-Cloud Interoperability Component](image)

**Figure 18.** Output specification in the Cross-Cloud Interoperability component

After the data obtained from the simulation runs are organized and combined, the data can be visualized in a form of graphs to:

- See results for one or more different settings of one value factor.
- Compare results of all connected factors using the same setting.

An example to the data visualization part for the Cross-Cloud Interoperability component is shown in the following figure.
Figure 19. Data Visualization in the Cross-Cloud Interoperability component

3.2.5.3 Interfaces and Workflow of the Cross-Cloud Interoperability Component
As the Cross-Cloud Interoperability component is a stand-alone component, there are no interfaces to other BASMATI components. Instead there is the user interface as can be seen from the previous three figures. The user interface allows the cloud provider to analyse the investment needed to increase the interoperability to the optimal level.

However, an extension to the Cross-Cloud Interoperability component could be the comparison of two service level agreements to understand the differences between the services offered by the federation and the services that could be provided to the federation. For this extension, the potential interfaces and a potential workflow of the Cross-Cloud Interoperability component is shown in the following sequence diagram. In this case, the Federation SLA manager would send the SLA IDs of the federation SLA and a cloud provider SLA offering to the Cross-Cloud Interoperability component.
Assuming the extension of the Cross-Cloud Interoperability component, the detailed workflow of the federation SLA compliance checking sequence is as follows:

- The Federation SLA Manager requests from the Cross-Cloud Interoperability component to check the compliance level of a cloud provider by comparing its SLA offer with the needed Federation SLA. For this, the Cross-Cloud Interoperability component gets two SLA IDs given.
- The Cross-Cloud Interoperability component requests the two SLA descriptions from the SLA repository.
- The Cross-Cloud Interoperability component gets the descriptions of the SLAs from the SLA Repository.
- After comparing the two SLA descriptions, the Cross-Cloud Interoperability component sends to the Federation SLA Manager the level of compliance calculated.
- The Federation SLA Manager might consider it for setting the cost and revenue sharing parameters of the federation.

3.3 Federation Data Management

3.3.1 Functionality of the Federation Data Management
Federation Data Management (FDM) supports a wide range of functionalities, which address data and data store management, enabling the BASMATI cloud federation.
In detail, there are two main functionalities of the FDM. The first functionality is to automate and coordinate the scaling of data stores through the cloud providers that have joined the BASMATI federation. This scaling comes after a request from the application owners, trying to deal with overload or delays due to long distance issues with their data servers.

The second FDM functionality addresses data migration and data replication across the various cloud providers, which are members of the BASMATI cloud federation. This may be needed for data redundancy or to counter delays in the data transfers between a moving user and a data store.

3.3.2 Architecture of the FDM
The Federation Data Management (FDM) component, called BASMATI Unified Data Management Framework (BUDaMaF), is composed of four basic sub-components. These components, developed in Java, are communicating with each other using Restful web services, with publicly available APIs. Thus, the BUDaMaF is using a loosely coupled architecture, having each sub-component perform a specific function. The general architecture of this framework is depicted in the following figure (Figure 1), followed by a short description of each sub-component and its function.
3.3.2.1 Core
The Core component is the most basic sub-component in BUDaMaF. It is the connection between all other components, coordinating their functions. It also provides an intelligence that performs real time decision making, in order to automatically perform offloading, replication.
and scaling actions. Using a simple task registry, it provides basic access control functionality and by communicating with the security and anonymization engine it can enforce data security rules to all data trafficking through the framework. Finally, having one central access point for the whole framework provides uniformity both in API calls and data jobs.

### 3.3.2.2 Offloading APIs

This component is, in essence, a docking point for a plethora of data store adapters, called wrappers. This docking point provides APIs that can perform a wide range of data store related tasks, such as data select, update, delete and data store scaling features. Each wrapper is created separately, and it is loosely connected to the offloading APIs components by getting the API calls and translating them into data store commands. In detail, each wrapper will be responsible for performing data store tasks and queries with certain types of data stores. After performing these tasks, the results will be forwarded to the docking point in an easily understood form (JSON). This will enable the BASMATI application to have seamless integration with the BUDaMaF and no concerns about the underlying data store types.

### 3.3.2.3 Analytics Engine

This component also serves as a docking point. It is loosely connected to independently developed Analytic Modules. The docking point is responsible for providing a unified API for all modules connected to it, giving them access to the data they need. After their analysis is completed, the docking point also provides the modules a unified API for storing and updating their analysis results in a BASMATI controlled data store. From this data store, all other components can access and use these analytic results for their own purposes.

### 3.3.2.4 Security and Anonymization Engine

This component has a direct connection to the Core component, providing it with a customized policy for data anonymization and security, depending on the current state of the art in data security and the types of data we need to manage. The component responsible for enforcing this policy will be the Core, making sure that it is enforced globally inside the BUDaMaF and all the data passing through it.

### 3.3.3 Workflow

The basic function of the BUDaMaF is providing scaling and data migration transparency to the federation and to the hosted applications. The following sequence diagram shows how such a process can be accomplished, using the sub-components described in the previous chapter (5.1).

In detail, we can see an application requesting a “data offload” task, which is a generalized description referring mainly to scaling processes. The details of this task contain all the information about the specific task requested (scale in, scale out or data relocation) and the exact requirements for that request (size of data store, data redundancy, location restrictions, budget and others). The Core sub-component creates a new job instance, in order to ensure logging and access control capabilities and, then, forwards the request to the Offloading APIs
sub-component. This component analyses the request and its requirements and finds the correct wrapper to handle this request. Finally, it assigns the request to this wrapper for execution. The Wrapper then takes the necessary actions, communicating either with ACE in order to create new data store hosts or an existing data store, in order to execute administrative actions or both if needed. Then, the success or failure message is returned to the application through the pipeline, containing all the necessary information corresponding to the request. For example, if the request was a scale out task, the response message will contain an access point and credentials, in order for the application to have access to the newly scaled cluster.

The second part of the diagram depicts the extra actions needed for data related tasks, such as duplication or migration. The application needs to send the data, or the credentials to access the data, to the Core which updates the job instance created during the initial request. Then the Core requests the current security rules, corresponding to these specific data, from the security and anonymization engine and applies the necessary anonymization or encryption. After that the data or the credentials are passed through the pipeline to the correct wrapper, reaching the targeted data store and a response is returned to the application containing either a success or failure message or the data themselves if the request was of a data retrieval type.

Figure 22. Data Offload/Replication Sequence Diagram
BUDaMaF supports one public and four internal APIs in the form of RestFul Web services. The internal APIs are used to communicate with the Wrapper, the Offloading APIs, the Analytics Modules and the Analytics Engine. We will not go into detail for these APIs as they are covered in deliverable D5.1. The public API is provided by the Core subcomponent, allowing users and applications to use the BUDaMaF functionality.

All POST requests are expecting input in the form of a JSON object following the format:

```
{
    "job_description": "One of scale_data_store, offload_data, replicate_data, get_application_data, get_security_report or get_monitor_data",
    "job_details": "The relevant job details for the chosen description in JSON object format, for example if the job is scale_data_store, the request should contain the target resources, target location, existing data store credentials if applicable and a budget threshold."
}
```

Then, the Core will start processing the request and return a JSON object containing either

- A “job running” reply and the initiator ID and password assigned to it as well as the job ID created for this job or
- The actual response if it is available in real time or
- A failure message detailing the error.

In the first case, the request owner must perform a GET request later, providing the credentials assigned to his request, in order to see the results or the status if the job is still running.
4 Advancement beyond the State of the Art

This section gives a brief overview about the advancements that have been achieved within BASMATI for the Federated Cloud Management Platform, the Cost and Revenue sharing, and the Cross-Cloud Interoperability component. Details about the advancements for the Federation Data Management component and the Federation SLA manager are given in deliverable D5.5 and deliverable D5.1, respectively.

4.1 Further Developments for BASMATI

The cloud federation-enabling components of the BASMATI Platform, including the architecture, the interaction with other BASMATI components, and the APIs are new and have been developed within the framework of BASMATI. With respect to the Cost and Revenue Sharing, the new development is the detailed definition of the factors that impact the sharing schemes. The value creation model that evaluates the interoperability of two cloud platforms is new as well.

In detail, we created a value creation model for analyzing the impact of investments in interoperability and portability, considering two platform maturity levels, two amortization periods, and different amounts of investments by providers. For the analysis, we can calculate profit, return on investment, net present value, and discounted return on investment. This is different from existing economics-based research studies that have analyzed portability and interoperability of platforms differently. For example, Matutes and Regibeau investigated product compatibility without network externalities [44]. Incompatibility of closed platforms obligates users to buy components from the same platform. Complementing research by Schiff analyzes the compatibility between platforms, considering closed systems of two-sided networks [45]. Arora and Bokhari compared closed systems with open systems [46]. The find that, while firms running open systems can specialize in producing a single component, firms running closed systems cannot specialize and must produce all components. IT service platforms, which improve portability and interoperability, increase the possibility of software services integration [47]. Research on value creation through the federation of IT service platforms has been considered by Haile and Altmann [18].

4.2 Challenges and Highlights

Within BASMATI and the implementation of the Cloud Management Platform, we followed the idea of providing automated cloud federation together with cost and revenue sharing schemes. Through this approach, the overall business process is streamlined, independent of whether there is a business relationship between service providers, between service providers of a federation, and between service providers and customers.

With respect to Cost and Revenue Sharing, the highlight is that several new sharing schemes can be developed based on the components that are available. This will allow cloud federations to emerge that address specific needs of customers as well as their federation members.
The model developed for evaluating the interoperability between cloud providers, and their SLA offerings, is new. In detail, there are three contributions: Firstly, the development of the quantitative model allows assessment of the impact of interoperability and service portability on the value propositions of cloud computing platforms. Secondly, using this model, it is possible to determine the role of interoperability and portability in the creation of provider value and user value. This investment analysis could become the basis for cloud providers, to design strategies for optimizing investments in interoperability with other cloud systems in order that they may subsequently join a cloud federation.
5 References


