

4CaaS

Building the PaaS Cloud of the Future

Use Case - Virtual private Cloud for mass-market: Scenario Definition

D 8.2.2

Version 3.0

WP8 – Experimentation

Dissemination Level: Public

Lead Editor: Stéphane Carrié

October 15th, 2012

Status: Final

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 258862



Seventh Framework Programme

FP7-ICT-2009-5

Service and Software Architectures, Infrastructures and Engineering



This is a public deliverable that is provided to the community under a Creative Commons Attribution 3.0 Unported License: <http://creativecommons.org/licenses/by/3.0/>

You are free:

to Share — to copy, distribute and transmit the work

to Remix — to adapt the work

Under the following conditions:

Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

With the understanding that:

Waiver — Any of the above conditions can be waived if you get permission from the copyright holder.

Public Domain — Where the work or any of its elements is in the public domain under applicable law, that status is in no way affected by the license.

Other Rights — In no way are any of the following rights affected by the license:

Your fair dealing or fair use rights, or other applicable copyright exceptions and limitations;

The author's moral rights;

Rights other persons may have either in the work itself or in how the work is used, such as publicity or privacy rights.

Notice — For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.

For a full description of the license legal terms, please refer to:

<http://creativecommons.org/licenses/by/3.0/legalcode>

Contributors:

Stéphane Carrié, France Télécom
Michel Dao, France Télécom
Sergio García Gomez, Telefónica I+D
Miguel Jimenez (UPM)
Nico Kruber, ZIB
Boris Moltchanov, Telecom Italia
Axel Spriestersbach, SAP
Pablo Arozarena, Telefónica I+D
Nicolas Chabanoles, Bonitasoft

Internal Reviewer(s):

Anna Gatzoura, NTUA
Charles Souillard, Bonitasoft

Version History

Version	Date	Authors	Sections Affected
0.1	1/3/2011	Stéphane Carrié	TOC created
0.2	10/6/2011	Stéphane Carrié	Initial Version
0.5	4/8/2011	Stéphane Carrié	Release for review
0.6	8/8/2011	Charles Souillard, Sergio García Gomez	Miscellaneous review updates
1.0	8/8/2011	Stéphane Carrié	D8.1.2 delivery
2.1	17/1/2012	Stéphane Carrié	S-Cube methodology Mass-market Business goals Use cases
2.2	31/1/2012	Sergio García Gomez Axel Priestersbach Pablo Arozarena	Global business goals
2.3	7/2/2012	Miguel Jimenez Nico Kruber Boris Molchanov	Scenario concept
2.4	17/2/2012	Anna Gatzoura, Charles Souillard	Miscellaneous review updates
2.5	22/2/2012	Stéphane Carrié	D8.1.2 resubmission delivery
2.6	10/4/2012	Stéphane Carrié	D8.2.2 initial version
2.7	10/8/2012	Miguel Jimenez	Update of prototype description
2.8	10/8/2012	Miguel Jimenez Nico Kruber Nicolas Chabanoles	Prototype configuration requirement section
2.9	10/9/2012	Stéphane Carrié	Inclusion modification requests from D8.2.5 evaluation report
2.10	10/10/2012	Anna Gatzoura, Charles Souillard	Miscellaneous review updates
3.0	10/15/2012	Stéphane Carrié	D8.2.2 final

Table of Contents

0.	Introductory note.....	9
1.	Executive Summary.....	10
2.	Introduction.....	11
3.	Methodology.....	12
4.	Domain description.....	12
4.1.	Glossary of main terms	12
4.1.1.	External actors	12
4.1.2.	Software provider actor	12
4.1.3.	Service provider actor	12
4.1.4.	Service customer actor.....	13
4.1.5.	Service user actor	13
4.1.6.	Software artefact	13
4.1.7.	Software partition (tenant)	13
4.1.8.	Software price plan.....	13
4.1.9.	Blueprint.....	13
4.2.	Description of relationships	14
4.3.	Domain laws	14
4.4.	Strategic dependency	15
4.5.	Context diagram.....	16
4.6.	Role sizing	17
5.	Scenario concept (S-Cube).....	18
5.1.	Description of the prototype	18
5.2.	Architecture of the prototype	19
5.3.	Integration with 4CaaSt platform.....	21
5.4.	Configuration of the prototype	22
5.4.1.	Wire cloud front-end	22
5.4.2.	Wikipedia page renderer	22
5.4.3.	SCALARIS database.....	22
5.4.4.	Bonita shop	22
6.	Business value / 4CaaSt business goals (S-Cube)	23
6.1.	BG#WP82_001 Support trading of service ecosystem	23
6.2.	BG#WP82_002 Lower marginal hosting cost	24
6.3.	BG#WP82_003 No limit scalability and reduced development cost.....	24
7.	Use cases (S-Cube)	25
7.1.	Develop cloud enabled software component.....	26

7.2.	Deploy Software in the marketplace for commercialization	27
7.3.	Commercialize Service provided by software	30
7.4.	Support customer purchase decision making (formerly Choose service in the marketplace).....	31
7.5.	Buy service from the marketplace	33
7.6.	Enforce SLA.....	34
7.7.	Enforce metering.....	36
8.	Evaluation method	37
9.	Conclusion.....	37
10.	References	38
Annex A.	Full list of Global Business Goals	38
A.1.	BG#WP2.BLUEPRINT.001	38
A.2.	BG#WP2.BLUEPRINT.002	39
A.3.	BG#WP3.MARKETPLACE.001.....	39
A.4.	BG#WP3.MARKETPLACE.002.....	40
A.5.	BG#WP4.DEPLOY.001.....	41
A.6.	BG#WP4.ELASTIC.001	42
A.7.	BG#WP4.NAAS.001	43
A.8.	BG#WP5.MON.001.....	43
A.9.	BG#WP5.MON.002.....	44
A.10.	BG#WP6.SM.001.....	45
A.11.	BG#WP7.SM.001.....	45
Annex B.	Glossary.....	46
Annex C.	References to D8.2.5	48

List of Figures

Figure 1. Relationships between actors and artefacts	14
Figure 2. Big picture activity diagram	15
Figure 3. 4CaaS context diagram	16
Figure 4. Number of Role players	17
Figure 5. Tourism demo application general architecture	19
Figure 6. Tourism demo application architecture with 4CaaS dependencies	20
Figure 7. UML diagram for use case “Develop cloud enabled software component“	27
Figure 8. UML diagram for use case “Deploy Software in the ...”	29
Figure 9. UML diagram for use case “Commercialize Service ...”	30
Figure 10. UML diagram for use case “Choose service in the marketplace”	32
Figure 11. UML diagram for use case “Buy service from the marketplace”	33
Figure 12. UML diagram for use case “Enforce SLA”	35
Figure 13. UML diagram for use case “Enforce metering”	37

Abbreviations

4CaaS	Building the PaaS Cloud of the future
ARB	Abstract Resolved Blueprint
AC	Application Component
API	Application Programming Interface
B2B	Business to Business
B2C	Business to Consumer
BAC	Business Advisory Committee
BPM	Business Process Management
CAPEX	Capital Expenditure
CSS	Cascading Style Sheets
DoW	Description of Work
DSL	Domain Specific Language
EULA	End User License Agreement
GA	General Assembly
IaaS	Infrastructure as a Service
HTML	Hypertext Mark-up Language
IP	Integrated Project
ITIL	IT Infrastructure Library
KPI	Key Performance Indicator
OVF	Open Virtualization Format
OPEX	Operational Expenditure
PaaS	Platform as a Service
PC	Project Coordinator
PIC	Product Instance Component
QoS	Quality of Service
REC	Runtime Execution Container
S&A	Service and Applications (used synonym in this document)
SaaS	Software as a Service
SB	Steering Board
SLA	Service Level Agreement
SLO	Service Level Objective
TAC	Technical Advisory Committee
TCC	Technical Coordination Committee
TCO	Total Cost of Ownership
TL	Task Leader
TM	Technical Manager
TOC	Table of Content
VM	Virtual Machine
WP	Work Package
WPC	Work Package Committee
WPL	Work Package Leader
XaaS	Anything as a Service: SaaS, PaaS, IaaS level services, but also Native or Immigrant APIs exposed as a Service

0. **Introductory note**

This document updates and supersedes deliverable D8.2.1.[7] It is very close from original document and follows exactly the same structure, only with deleted, appended or modified text. Changes from orgininal deliverable can easily be identified as deleted text is striked trough (~~this is a sample deleted text~~), and new text has a colored background (this is sample new text). Some old text has been simply deleted in wrapping chapters when it was unimportant (no text was simply deleted in use case and prototype description).

1. Executive Summary

This deliverable is one out of three 4CaaS deliverables focusing on scenarios for the evaluation of 4CaaS platform and its features against real life use cases.

This document describes usage scenarios of 4CaaS targeting mass-market segment. More specifically it focuses on features most needed for mass-market segment, similarly other scenarios focus on SMEs and large enterprises market segments.

In addition to 4CaaS generic business goals available in annex A, we have identified 3 dominant business goals:

- **Trading of services ecosystem (vs. trading of application tenant or raw IAAS resources)**

In this mass-market business goal, users are independent subscribing customer. They are typically not interested in the wholesale acquisition of an application instance or tenant. They are indeed most likely interested in a single user account, granting them access to a service. Whereas for SME scenario, you would find that users are grouped in organisational units handling many responsibilities such as subscription or bill payment. You would find that these organisational units typically buy a full application instance or tenant that supports the need of one group of users. This business goal is related to the marketplace and user management.

- **Low marginal cost**

In this mass-market business goal, marginal cost is especially important in comparison to other scenarios, because of the high number of users, typically a 6 digits number. This business goal is related to all forms of resource usage optimization.

- **No limit scalability and reduced development cost**

In this mass-market business goal, applications shall be highly scalable. But this generally implies complex architectures and thus more expensive design and higher development costs. This business goal is related to all forms of scalable service reuse and scalability features.

The achievement of business goals is demonstrated by developing a software prototype based on the "TOURISM demo application" already shown during past reviews and by using 4CaaS platform to run this software.

This WP8 deliverable was preceded by the following deliverables:

- Software prototype deliverable which provide details about the prototype itself (D8.2.7 due M20)
- Report on Integration (D8.2.3 due M20)
- Report on experimentation results (D8.2.5 due M26)

This WP8 deliverable will be followed by the following WP8 deliverables:

- Software prototype deliverable which provide details about the prototype itself (D8.2.8 due M32) Report on Integration (D8.2.4 due M32)
- Report on experimentation results (D8.2.6 due M39)

This document is itself an update of D8.2.1. It uses S-Cube methodology and provides business goals and scenarios as well as other S-Cube artefacts.

2. Introduction

The primary goals of WP8 scenarios are to provide business scenario from which platform requirements will be extracted (Deliverable D1.1.2 [8]). It will then be used to validate the 4CaaS value proposition defined in WP1 and also to provide measurable information for backing up market analysis done in task 9.1.

Evaluation of 4CaaS is complex as 4CaaS is many things at once, including a generic PaaS platform and a portfolio of built-in services:

- A generic PaaS platform: 4CaaS is first of all a framework for managing the whole lifecycle of an application, from specification to billing including pricing, rating, deploying, monitoring.
- A service portfolio: 4CaaS is also a set of predefined optional components (described by blueprints) that an application/service developer can leverage in his application/service if they want to a) inherit some of the generic 4CaaS features depending on the component, such as scalability and b) benefit by using those components “as a Service” without the need of operation, updates, maintenance etc.

The secondary goals of WP8 are

- Provide feedback on 4CaaS platform usage to Work packages 1 - 7
 - Bug report
 - Usability report
- Provide demonstration material for dissemination

Most notably, the following goals are not in the scope of WP8, although this work package helps contributing to their achievement as a side effect of iterative design and development:

- Define 4CaaS features and value proposition (done in WP 1 - 7)
- Unit test 4CaaS platform (done in WP 1 - 7 with software prototypes)

3. Methodology

In order to make case studies comparable and easy to understand, S-Cube has defined a case study description approach [2] based on the classical Requirements Engineering literature [4] that leverages from the results achieved by NEXOF-RA[3]. According to the methodology, a case study is described in terms of:

- A Domain description.
 - A glossary that defines the main terms of the world.
 - A description of the relationships between the terms of the glossary
 - A description of any law that is relevant in the world (not applicable).
 - Strategic Dependency Diagrams
 - Context Diagrams
- A list of Business Goals and/or Domain Assumptions for the case study.
 - Business Goals define objectives to be pursued and functionalities to be offered.
 - Domain Assumptions describe properties that are assumed to be true in a certain domain.
- A list of Scenario descriptions.

4. Domain description

4.1. Glossary of main terms

According to the methodology, this section contains a list of the main terms used in this document and makes this document self-consistent. Reference definition for 4CaaSt are however available in D1.2.1 [9] deliverable.

4.1.1. External actors

External actors are individuals or entities for which the 4CaaSt platform is built.

Internal actors such as “platform provider” or “platform administrator” are not listed in the present document. These are needed to run the platform but the platform was not built for them. We are only interested here in actors external to the platform, those the platform was built for.

4.1.2. Software provider actor

Software provider is an external actor in charge of creating the business software that offers some business value to a customer. Examples of such software are a “TOURISM demo application”, a web mail, a customer relationship management, etc...

4.1.3. Service provider actor

Service provider is an external actor in charge of commercializing and running the application developed by software provider. He is responsible for the day-to-day operation and service customer support.

4.1.4. Service customer actor

A service customer is an external actor who subscribes to a tenant of an application. This tenant may be an application instance in case of single-tenancy or a partition of an application instance in case of multi-tenancy. This is why this document upgrades the notion of “application subscriber” to the clearer notion of “service customer”.

Service customer is not the end user of the application. He may however have access to the application administration.

A service customer has the responsibility of managing user of the application. Depending on the nature of users, this may include pricing and billing for mass-market or internal accounting for corporation.

4.1.5. Service user actor

Service user is an external actor who is the final user of the application. In the case of a mail application, he is the one sending mail, while in customer relationship management he is the one managing customer.

4.1.6. Software artefact

Software artefact is a set of files (code, image, descriptor, xml, dependency definition etc.) that grouped together form software, such as a Customer Relationship Management application, a tourism demo application, a webmail server, etc.

4.1.7. Software partition (tenant)

A software partition (or tenant) is an application dedicated to a “Service customer”.

4.1.8. Software price plan

A software price plan defines the commercial conditions under which a “Service customer” gets access to a dedicated software partition (tenant).

4.1.9. Blueprint

A blueprint is a formalism used to describe software architectures and dependencies. It contains sufficient data to perform automations such as installation, configuration, etc.

4.2. Description of relationships

According to S-Cube methodology, this section is a description of the relationships between the terms of the glossary. The glossary alone does neither highlight the relationships between the various terms nor their relative importance. Thus we need to build a model that highlights these aspects. Class diagrams are usually a good tool for this purpose since they allow the engineer to identify main entities as classes and to express several kinds of relationships between those. Entity-relationship diagrams, as well as semantic networks, for our purposes have an expressive power that is similar to class diagrams and therefore can be used as well.

“Software provider” provides “software artefact”, such as CRM application software or “TOURISM demo application”.

“Service provider” uses a “software artefact” to create a price plan.

“Service customer” subscribes to the application. He gains access to a dedicated software partition hosting his data and servicing a set of “service user” that he can define.

“Service User” then subscribes to an account that will grant him access to the application (form of subscription depends highly on market segment)

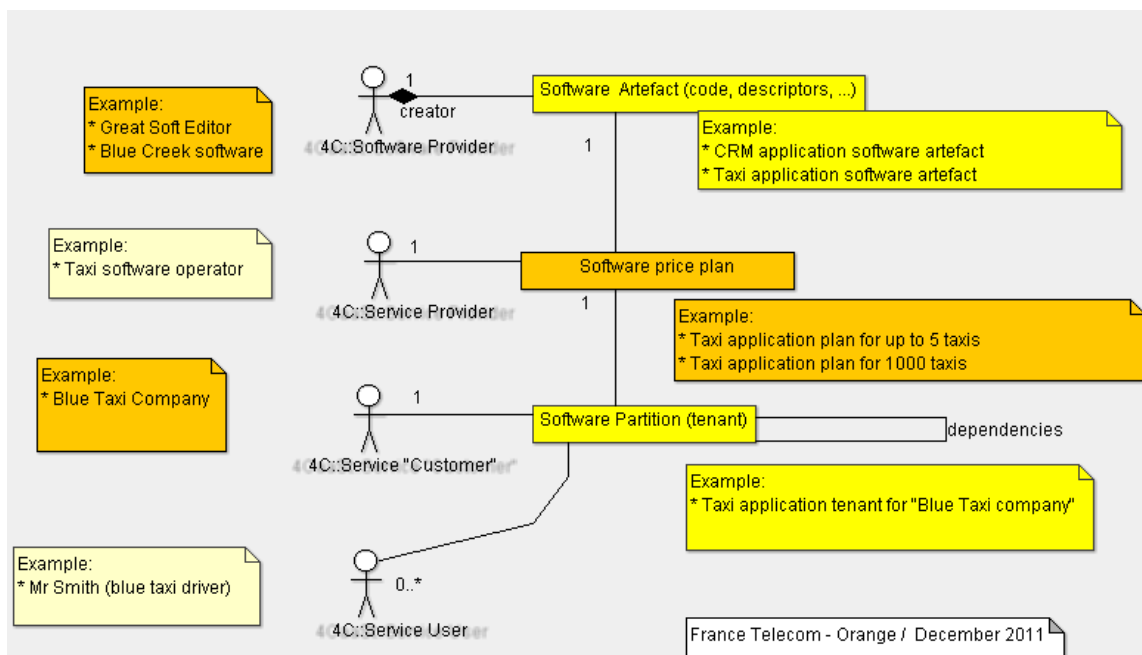


Figure 1. Relationships between actors and artefacts

4.3. Domain laws

According to S-Cube, this section is a description of any law that is relevant in the world. Such laws can be expressed in any form that is typical of the application domain that we are considering: mathematics, logics, natural language, etc.

Laws in software industry refer to formalisms of all kind used to express application logic and formalize problem field.

Software artefacts are expressed with tools and language already existing and 4CaaSt can only embrace them. This includes J2EE, SQL and other language we will not list here.

4.4. Strategic dependency

According to S-Cube methodology, strategic relationships are used to model the dependencies between different actors in an organisational context. They especially help in modelling user (roles) together with their relations. Dependency edges in the diagram link the actors with needs to actors with the capability of meeting those needs. The needs are expressed in terms of goals.

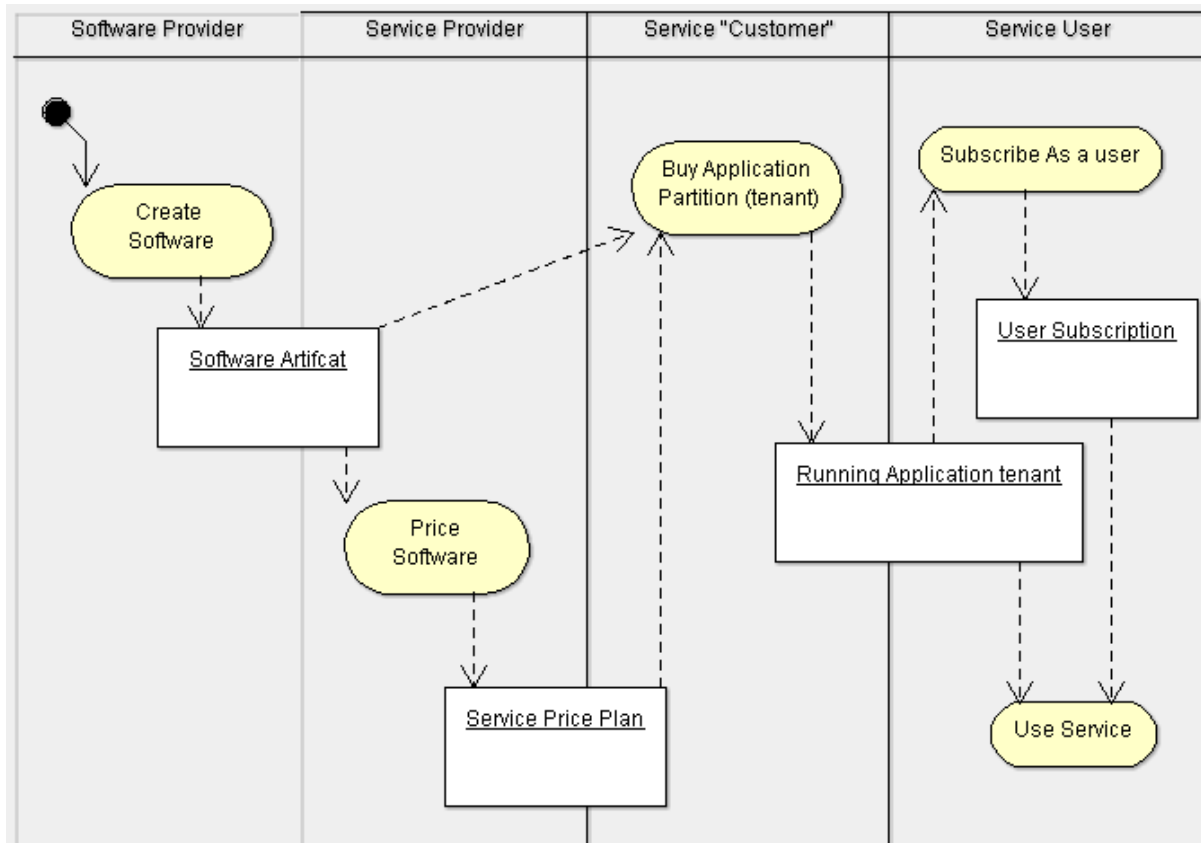
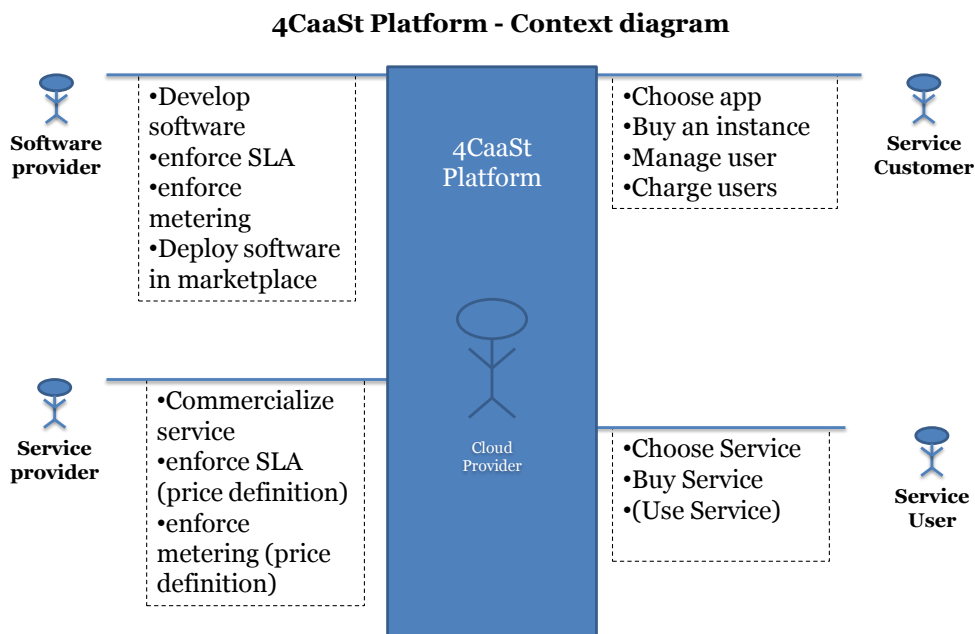


Figure 2. Big picture activity diagram

4.5. Context diagram

According to S-Cube methodology, context diagrams identify the agents that operate in a certain context, as well as the machines that need to be developed. Moreover, they highlight the phenomena shared between agents and machine. In a context diagram, any active entity of the case study to be modelled is represented as a box, while a directed arrow describes phenomena between agents. Both the SDDs and the CDs represent the agents/actors involved in the domain, but while the SDDs show the dependencies among them, CDs highlight also other kinds of relationships among them. Moreover, they clearly identify the boundaries between the machine and the world.



2011 - France Télécom / Orange

Figure 3. 4CaaS context diagram

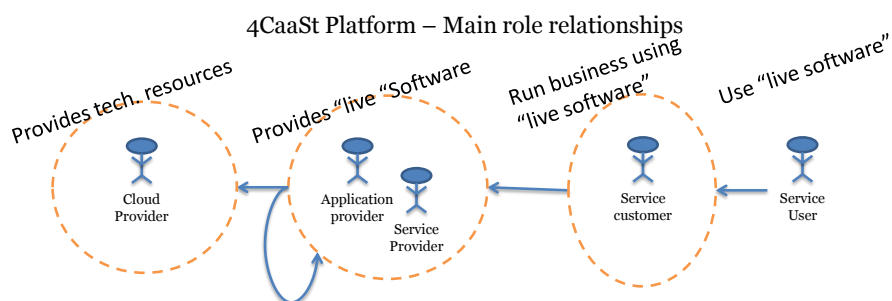
4.6. Role sizing

4CaaS targets three market segments known as Small & Medium Enterprises, mass-market and large corporations.

When targeting SME/PME we typically envision a scenario with a single platform provider, thousands of service providers, thousands of tenants per application and hundreds of users per tenant.

When targeting mass-market, we typically envision a scenario with a single platform provider, thousands of service providers and millions of service consumers. Contrary to SME/PME, every user shall be in a single tenant as there is no other envisioned grouping.

Finally, when targeting corporation, we typically envision a more complex deployment scenario with several platform providers. There could be a few service providers. Also, there may be a few tenants such as organisational units. Finally, service consumers are typically corporation employees and could be tens of thousands.



	Platform Provider	Applica provider & Service Provider	Service customer	Service User
SME/PME	1	Thousands	Thousands	Hundreds
Mass Market	1	Thousands	1 or few	Millions
Corporation	several	1 or few	1 or few	Thousands

2011 - France Télécom / Orange

← customer of

Figure 4. Number of Role players

5. Scenario concept (S-Cube)

4CaaS is a PaaS platform bundled with a built-in portfolio of optional services such as context, mashups, data store and others.

The software prototype described below shall be representative of a mass-market application using services from 4CaaS built-in portfolio. Its usefulness is twofold:

- Developing this prototype and using it from a customer perspective shall demonstrate how desirable 4CaaS portfolio of services can be for a mass-market application.
- Deploying this prototype in the 4CaaS platform and using 4CaaS features as described in this document's use cases, shall demonstrate suitability of these feature for a mass-market application.

5.1. Description of the prototype

The Tourism Demo is a showcase demonstrating the benefits of the various different technologies developed and deployed in the 4CaaS platform as ready-to-use services. This demo makes use of the functionalities Context as a Service, Data Store as a Service, Publish/Subscribe as a Service and Mashups as a Service to create a complete tourism application. The developed application uses location as contextual information to enrich the information provided. Such information is used to offer the user descriptions of touristic places nearby, updating the list of interesting places as the user is moving around the city. The information is stored in a Wikipedia clone called "Wikipedia on SCALARIS" which uses SCALARIS as its storage, exploiting enhanced easy-to-use scalability features offered by such a data store available in the form of a 4CaaS service. Users have the ability to start real-time conversations about such interesting places. Finally, the interface is built using Mashups as a Service, which integrates different internal and external back-end services used by the application through a composable interface made up from gadgets. Gadgets are simple web-technology-based interface components programmed in HTML, CSS and JavaScript which communicate with each other, invoke other services and show the result to the user. The gadgets present interesting places in a list and also put the same places as markers on a map. Both views are connected to the Wikipedia-gadget, which is used to show information, and a chat window used to share comments about an item when the user selects it, either in the list or in the map.

This demo shall show the benefits derived from the use of 4CaaS and 4CaaS services to develop a cloud-scale application instead of developing all the functionality required specifically for the application. Specifically, the following benefits come from the use of different 4CaaS Native PaaS technologies:

- A scalable data store that can support extreme number of requests increase (for descriptions of tourist places) by simply adding more resources. In order to offer this kind of scalability as well as a simple API, data store only supports key-value semantics (as opposed to relational data stores). Despite this restriction, application functionality does not need to be sacrificed which is shown by this implementation of a Wikipedia clone.
- Out-of-the-box availability of context information through the Context Management System (CMS). Development of this demo makes use of the location feature, automatically obtaining the location of the user from the CMS without having to develop such functionality. The location information can be uploaded to the CMS using different context sources such as Cell ID or an application installed in the user Smartphone. The developer is freed from the burden of integrating location information from different sources, and can easily request the latest context information.

- Mashup-based interface. This service has been used to foster the development of the interface, creating simple interface gadgets that are composed to make up the interface. These gadgets make use of back-end information services and communicate with each other, creating a composite application. This interface is highly modifiable by the user or the developer, choosing new gadgets, arranging and connecting them upon his needs. The composition and communication is stored as an object called Mashup, so the user has it ready to be used.
- Publish/Subscribe as a service provides the developer with a powerful, scalable message-based communication mechanism decoupling sender and receiver, that can be used both to communicate entities of the same application or of different and heterogeneous services.

5.2. Architecture of the prototype

The prototype is composed of several services, both internal and external to 4CaaSt, integrated through the Mashup-based interface. The interface is provided by means of the Mashup platform through the set of gadgets developed for the demo. These gadgets carry out the invocations to the services, communicate with them and show the results to the user. The following figure shows the overall architecture:

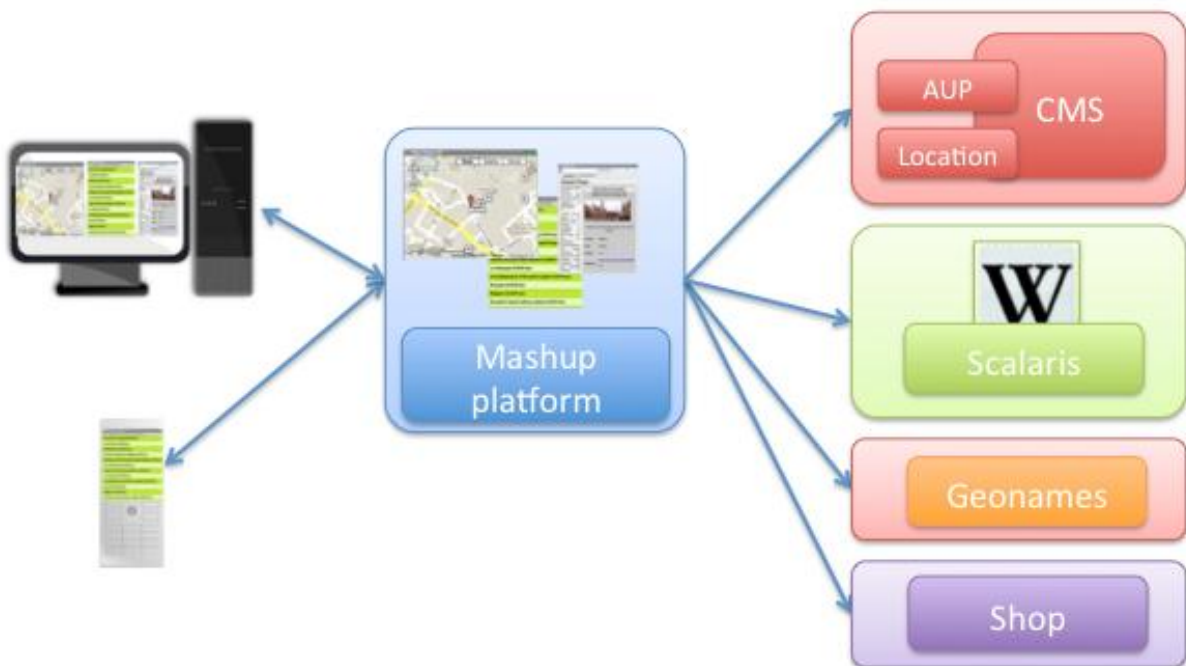


Figure 5. Tourism demo application general architecture

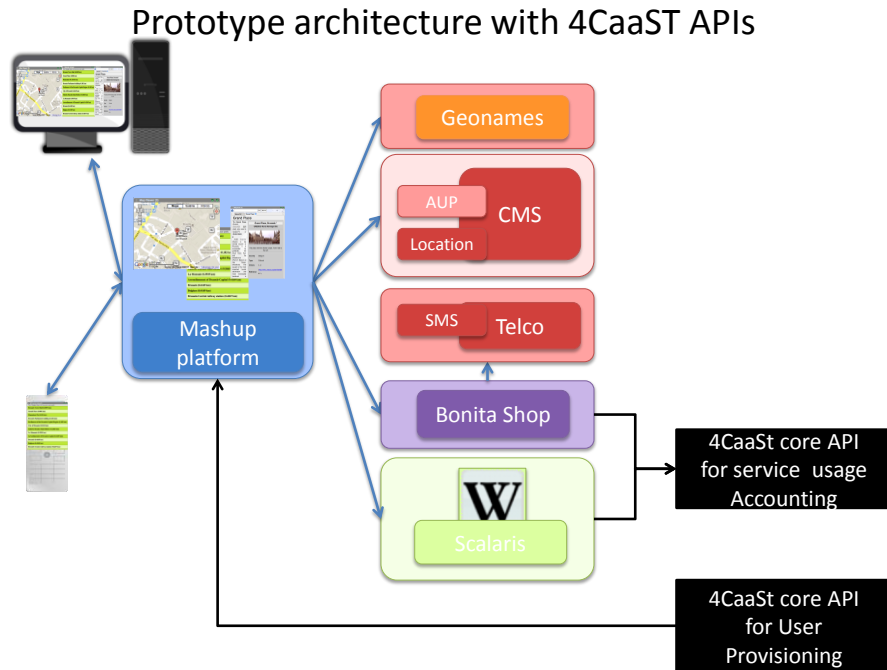


Figure 6. Tourism demo application architecture with 4CaaSt dependencies

The tourism demo application is based on three different backend services:

- ~~Context as a Service, offered by the Context Management System (CMS). This service is part of the Native PaaS technologies offered in 4CaaSt as part of WP6. Context service is used to discover location of the user, gathered through different sources, as well as some other information about the user such as social Web contacts, via the Advanced User Profile, or interesting places for the user. This service is used in the prototype to discover the location of the user, so as to personalize the list of interesting places to be shown or display user friends around him.~~
- Context as a Service, offered by the Context Management System (CMS). This service is part of the Native PaaS technologies offered in 4CaaSt as part of WP6. Context service is used to discover location of the user, gathered through different sources, as well as some other information about the user such as social Web contacts, via the Advanced User Profile, or interesting places for the user. This service is used in the prototype to discover the location of the user, so as to personalize the list of interesting places to be shown or display user friends around him.
- Wikipedia on SCALARIS. This service is a proof of concept of the features of SCALARIS as an efficient data store, and serves a subset of the Wikipedia using SCALARIS as its back-end. This service is used in the demo to retrieve information about interesting places near the user. SCALARIS is of the Native PaaS technologies offered in 4CaaSt as part of WP6.
- Museum ticket shop. This service allows for the easy purchasing of tickets of museums, and can be activated as the user gets interested in any museum shown nearby. The shop is based on the BPM process definition.
- Geonames is an external reverse location lookup service used to discover the interesting tourism places that are near a specific geographic position. The result of this service is used to populate both the list of touristic places and the map.

Another 4CaaSt service is used to create and serve the interface for all of these services, Mashups as a Service, which serves the Mashup of gadgets that make up the interface. Such gadgets are simple interface components programmed in HTML, CSS and JavaScript that perform a single action (usually using a back-end service) and have a simple web-based

interface. The gadgets can be generic enough to be used in multiple composite applications or developed ad-hoc for a specific purpose. For this prototype, a Wikipedia viewer gadget has been used together with two new gadgets, the map and the list of tourist places:

- Gadget “list of tourist places”, which makes use of the information coming from the Geonames service and prints the name of the tourist places sorted by the distance to the user.
- Gadget “map of tourist places”, which shows the same information handled by the list, plus more information such as user friends, but inside a map centred on the user position. This gadget is based on Google Maps API and also lets the user chose a place by its position on the map.
- Gadget “Wikipedia viewer”, which shows one or more Wikipedia articles. This is a standard gadget which previously existed, but now uses Wikipedia on SCALARIS service as back-end in this prototype.
- ~~Gadget “Social comments” will be included so as to allow users to carry on a conversation around a certain tourism place.~~
- Gadget “Social comments” allow users to carry on a conversation around a certain tourism place, sending and receiving comments to unknown users browsing that place.
- Gadget “Tickets shop” serves as front-end for a ticket purchasing service that allows the user to buy museum tickets and get discounts with them.

The possibility of wiring them, i.e. making them able to send notifications upon user actions and configuring such communication, makes them work together as in any other application, but independently programmed. Both the list and the map send notifications on the item selected by the user, and the Wikipedia viewer shows the information of such an item, using the Wikipedia on SCALARIS service.

The result is a highly configurable interface, easily extended with new or upgraded gadgets that can be used on desktop computers, tablets or smartphones like iPhone or Android phones. In the case of smartphones, only one gadget is shown at a time, giving the user the possibility of browsing through the rest of gadgets in the Mashup.

5.3. Integration with 4CaaSt platform

The created prototype shows the benefits derived from the use of different 4CaaSt services reducing the complexity, effort and time required to implement a cloud-enabled application. In this prototype several services, like Data Store as a Service, Content as a Service and Mashups as a Service, are integrated, using minor developments so as to have a fully usable context-aware tourism application.

These services will be offered in 4CaaSt as any other service running in the platform, using the Blueprints mechanism. The Blueprints of the services Context as a Service, Data Store as a Service and Mashups as a Service are defined in Deliverable D6.2.1[11] Native PaaS Technologies: Components Design and Open Specification. The use of these services will be as simple as stating the desired service as a requirement in the developed application, and using the corresponding programmatic API. In the case of Mashups, both gadgets and mashups will have their own Blueprint, which will indicate their deployment in the Mashup platform.

These services will be offered as self-managed services running in 4CaaSt, so the developer is freed from the burden of controlling their correct working, scalability and performance. Moreover, these features are the benefits of offering these services integrated in 4CaaSt Platform.

5.4. Configuration of the prototype

Prototype components need to be configured upon deployment. They need for example to be informed somehow of the how to reach another component they need to have access to such as a database IP address and credentials.

We have identified for Tourism demo application the following list of information that 4CaaS need to somehow provide to the components configuration logic:

5.4.1. Wire cloud front-end

Wire cloud front-end needs to be made aware of the http URL of the SCALARIS-wiki which depends on the IP address of the server where the SCALARIS-wiki is deployed. It also depends on the location of the pubsub service needed by its JavaScript API. It also depends on an external service 4CaaS should be aware of. Finally, front end needs to be aware of user allowed to connect the service by 4CaaS.

5.4.2. Wikipedia page renderer

Wikipedia page renderer needs to be made aware of the location of the SCALARIS database access node. It needs to be aware of where to send accounting data it is collecting. It needs to be aware of the SERVERNAME it was deployed to with PORT number, and SERVERPATH it is installed in.

At runtime, it also needs to be aware of the connected user to which will be associated accounting data.

5.4.3. SCALARIS database

SCALARIS database needs to be made aware of the virtual machine provisioned by 4CaaS for the tourism application, It needs to know if a node is a “first node” or not and be configured accordingly. Other servers need to be made aware of the address of the first node.

It needs to be aware of where to send metering data for 4CaaS platform that will be used for scalability.

5.4.4. Bonita shop

Bonita shop component needs to be made aware of how to connect to SMS service and how and where to send accounting data it is collecting to 4CaaS and for what user account.

6. Business value / 4CaaSt business goals (S-Cube)

According to S-Cube methodology business goals define objectives to be pursued and functionalities to be offered. Domain Assumptions describe properties that are assumed to be true in a certain domain. The table below defines the descriptive field that should be provided for describing Domain Assumptions and Business goals.

Field	Description
Unique ID	a unique ID for this goal/assumption
Short name	a short name for this goal/assumption
Type	Business goal or Domain assumption
Description	intention of the goal/assumption
Rationale	a justification of the goal/assumption
Involved Stakeholder	Stakeholders involved in the business goal/assumption
Supporting materials	a pointer to documents that illustrates and explains this goal/assumption (in particular those of domain analysis)
Priority of accomplishment	One of the following: Must have: The system must implement this goal/assumption to be accepted. Should have: The system should implement this goal/assumption: some deviation from the goal/assumption as stated may be acceptable. Could have: The system should implement this goal/assumption, but may be accepted without it.
Tentative scheduling	Tentative scheduling of accomplishment. To be used only if case study has to be implemented.

The following business goals add up to 4CaaST global business goals which are listed in “Annex A. Full list of Global Business Goals”. They are listed in order of decreasing importance.

6.1. BG#WP82_001 Support trading of service ecosystem

Field	Description
Unique ID	BG#WP82_001
Short name	Support trading of service ecosystem
Type	Business goal
Description	4CaaSt shall support the generic interactions needed for an ecosystem of service trading involving software providers, service providers and mass-market service users.
Rationale	Actors don't have to develop 1 to 1 relationships in order to become a member of the ecosystem and because it is a shared environment an ecosystem of service trading can develop.
Involved Stakeholder	Service Provider, Software Provider, Service User
Priority of accomplishment	Must have

Field	Description
Tentative scheduling	RP3

6.2. BG#WP82_002 Lower marginal hosting cost

Field	Description
Unique ID	BG#WP82_002
Short name	Lower marginal hosting cost
Type	Business goal
Description	In order to address mass-market segment, 4CaaS shall provide features that allow the reduction of IAAS and human resources marginal costs for a given quality of service and for an arbitrary large number of users.
Rationale	When a mass-market service becomes successful, marginal costs tend to become more important than fixed costs and their reduction becomes a major concern.
Involved Stakeholder	Service Provider
Priority of accomplishment	Must have
Tentative scheduling	RP2

6.3. BG#WP82_003 No limit scalability and reduced development cost

Field	Description
Unique ID	BG#WP82_003
Short name	No limit scalability and reduced development cost
Type	Business goal
Description	Developing an application for deployment on 4CaaS shall be cheaper even for Mass-market segment which requires no limit scalability.
Rationale	Architectural complexity of high scalability is expected to be handled by 4CaaS thus lowering engineering effort. Some basic reusable services are also expected to be provided by 4CaaS and simply be reusable, thus lowering standard development effort.
Involved Stakeholder	Service developer
Priority of accomplishment	Must have
Tentative scheduling	RP2

7. Use cases (S-Cube)

According to S-Cube, the phenomena shared between the world and the machine can be detailed through scenario descriptions. Scenarios have an operational flavour in the sense that they describe the steps that need to be followed by the machine and the world entities in order to accomplish a certain task. The table below describes how S-Cube scenarios should be detailed and described, and it should be used as a template for any single scenario description. Here, a scenario is described using information about the business goals or the domain assumptions they refer to, the operational description of the scenario, the possible problems involved and the supporting material.

Mass-market scenario focuses on the following 7 scenarios designed to satisfy mass-market needs. You will find a consolidated derived list of features in deliverable D.1.1.x.

Mass-market use cases UC.8-2-00?	1 Develop cloud enabled software component	2 Deploy Software in the marketplace for commercialization	3 Commercialize Service provided by software	4 Support customer purchase decision making Choose service in the marketplace	5 Buy service from the marketplace	6 Enforce SLA	7 Enforce metering
Business goals							
WP82_001 Support trading of service ecosystem			Y	Y	Y		
WP82_002 Lower marginal hosting cost						Y	
WP82_003 No limit scalability and reduced development cost	Y	Y				Y	Y
WP2.BLUEPRINT.001 Blueprint: Abstract application design	Y					Y	Y
WP2.BLUEPRINT.002 Application Lifecycle Management					Y		
WP3.MARKETPLACE.001 One Stop Shop		Y	Y	Y	Y	Y	Y
WP3.MARKETPLACE.002 Business customization							
WP4.DEPLOY.001 Automated Provisioning and Operation for Heterogeneous Components					Y	Y	Y
WP4.ELASTIC.001 Platform Elasticity						Y	
WP4.NAAS.001 NaaS Services - Layer 2/3 virtual networks for the deployed applications							
WP5.MON.001 Integrated Monitoring						Y	
WP5.MON.002 Integrated Accounting of services							Y
WP6.SM.001 Integration of native services	Y						
WP7.SM.001 Cloud aware software platforms	Y				Y	Y	Y

Some goals may not have matching use cases in this scenario, but not in all three WP8 scenarios.

7.1. Develop cloud enabled software component

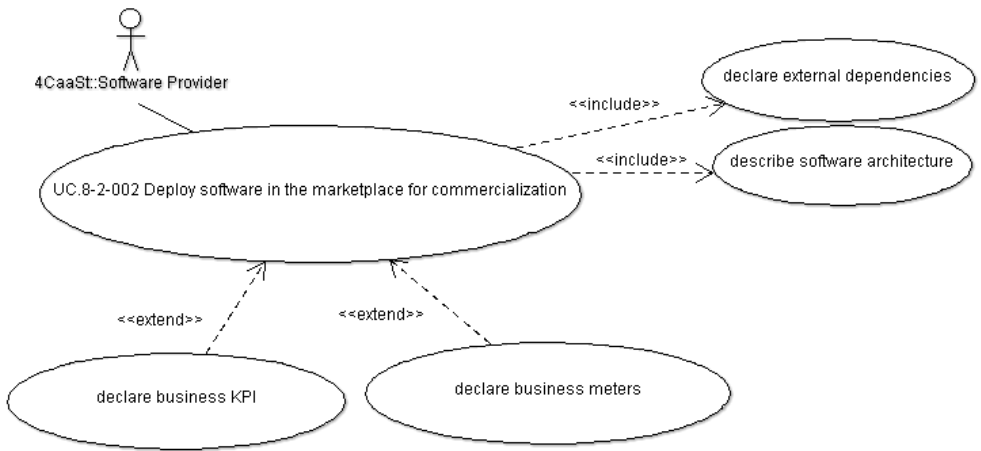
Field	Description
<i>Unique ID</i>	UC.8-2-001
<i>Short name</i>	Develop cloud enabled software component
<i>Related to</i>	BG WP82_003 No limit scalability and reduced development cost BG WP2.BLUEPRINT.001 Blueprint: Abstract application design BG WP6.SM.001 Integration of native services BG WP7.SM.001 Cloud aware software platforms
<i>Involved actors</i>	Software Provider
<i>Detailed Operational Description</i>	<p>➤ At the beginning of this use case, a software developer needs to develop a software component that implements a desired feature</p> <p>➤ At the end of this use case, a software developer has a software component providing the desired feature.</p> <p>A software developer wants to develop an application such as “Tourism demo application”.</p> <p>He would like to save on training, on coding, on debugging and as a result have a shorter time to market.</p> <p>He is able to reuse the following 4CaaS features or services:</p> <ul style="list-style-type: none"> • Write “context query” in “context query language” to get access to smart context information based on available context sources (location, weather forecast, social network) • Write new web GUI components using Mashup platform • Integrate Mashup components • Write an html page generator in Java for rendering Wikipedia pages from scalable key/value storage. • Store data in scalable key/value storage using a standard java storage mechanism developer are already accustomed. <p>He is able to write “glue code” easily in order to integrate with 4CaaS architecture. He is able to derive from existing glue code and write only what is specific to his requirements.</p> <p>He is able to access required information from 4CaaS in order to configure his software components (database address ...).</p> <p>He is able to add a new technical component to 4CaaS such as a new storage component (NB: we will consider here that SCALARIS is provided by the prototype and not by 4CaaS and that integration cost of SCALARIS into 4CaaS is endorsed by the service provider). He is able to have his component scale up and scale up horizontally following 4CaaS requests.</p>
<i>Problems and challenges</i>	Although 4CaaS provides a set of atomic development tools, it is deliberately not in the scope of 4CaaS to provide an integrated development environment and methodology. Thus evaluation only covers atomic features, and standard practises and tools used for development are left aside.

Field	Description
Additional material	<p>Figure 7. UML diagram for use case “Develop cloud enabled software component”</p>

7.2. Deploy Software in the marketplace for commercialization

Field	Description
Unique ID	UC.8-2.002
Short name	Deploy Software in the marketplace for commercialization
Related to	BG WP82_003 No limit scalability and reduced development cost BG WP3.MARKETPLACE.001 One Stop Shop
Involved actors	Software Provider

Field	Description
Detailed Operational Description	<p>➤ At the beginning of this use case, a software developer has a developed application. (Development is not fully in the scope of 4CaaSt).</p> <p>➤ At the end of this use case, the software is published in the marketplace and is ready to be commercialized, meaning no subsequent development action is needed.</p> <p>A software developer has developed an application such as “Tourism demo application” (A bootstrap version is shown in deliverable D1.3.1 [10]).</p> <p>The logic is developed in his favourite language, chosen among those supported by 4CaaSt. He specifies what his requirements are regarding runtime platform using 4CaaSt supported descriptor (specific versions or range, specific editor, etc.).</p> <p>He provides the numerous binaries or source of his application in the 4CaaSt supported format.</p> <p>He provides 4CaaSt descriptor specifying the desired architecture of his application and how his components are interconnected.</p> <p>He provides information about dependencies toward external services. He provides dependencies towards 4CaaSt built in services such as key/value data store, context or any other service.</p> <p>He is easily able to detect error in his graph of dependency.</p> <p>He can include free text information documenting the intent of such of such software element.</p> <p>He provides meta-data about application specific meters that 4CaaSt needs to be made aware of.ⁱ</p> <p>He provides meta-data about application specific KPIs, that 4CaaSt needs to be made aware of.ⁱⁱ</p> <p>He provides meta-data information about how his application nodes can be scaled up or notⁱⁱⁱ</p> <p>If necessary he provides 4CaaSt optional information such as</p> <ul style="list-style-type: none"> • specific initialization logic (initial data set, etc.), • upgrade or downgrade logic (data schema modifications, etc.);^{iv} • self diagnostic and monitoring logic or meta-data.^v
Problems and challenges	Expressiveness vs. complexity of descriptor languages provided by 4CaaSt

Field	Description
Additional material	 <pre> graph TD Actor[4CaaSt:Software Provider] --> UC((UC.8-2-002 Deploy software in the marketplace for commercialization)) UC -.-> <<include>> UC1((declare external dependencies)) UC -.-> <<include>> UC2((describe software architecture)) UC3((declare business KPI)) -.-> <<extend>> UC UC4((declare business meters)) -.-> <<extend>> UC </pre> <p>Figure 8. UML diagram for use case “Deploy Software in the ...”</p>

7.3. Commercialize Service provided by software

Field	Description
Unique ID	UC.8-2.003
Short name	Commercialize Service provided by software
Related to	BG WP82_001 support trading of service ecosystem BG WP3.MARKETPLACE.001 One Stop Shop
Involved actors	Service Provider
Detailed Operational Description	<p>➤ At the beginning of this use case, software implementing services has been published by a software provider in the marketplace but is not available for purchase.</p> <p>➤ At the end of this use case, service provided by the software can be purchased from the marketplace and user can use it.</p> <p>Service provider selects software in the marketplace that has been put there by a software provider.</p> <p>Service provider proceeds to the definition of commercial information about the service and definition of commercial conditions under which the software may be purchased.</p> <p>In the case of the “TOURISM demo application”, a flat monthly subscription fee is defined (Please refer to enforce metering for more advanced price consideration).</p> <p>Service provider marks service as available for purchase so application becomes visible to service user.</p>
Problems and challenges	<p>Clarity and ease of use for price condition definition</p> <p>Support for trading of services (vs. Applications)</p> <p>Support for revenue sharing associated with sub-services.</p>
Additional material	<pre> graph TD Actor1[4CaaSt:Service Provider] --- UC1((UC.8-2.003 Commercialize Service provided by software)) Actor2[4CaaSt:Service User] --- UC2((Open Service to subscription)) UC1 -.-> <<include>> UC3((Select Software)) UC1 -.-> <<include>> UC4((Define price plan for service use)) UC1 -.-> <<include>> UC2 UC3 -.-> <<include>> UC4 UC5((Define price plan relative to SLA)) -.-> <<extend>> UC4 UC6((Define price plan relative to business meters)) -.-> <<extend>> UC4 </pre> <p>Figure 9. UML diagram for use case “Commercialize Service ...”</p>

7.4. Support customer purchase decision making (formerly Choose service in the marketplace)

Field	Description
Unique ID	UC.8-2-004
Short name	Support customer purchase decision making ^{vi} Formerly (Choose service in the marketplace)
Related to	BG WP82_001 support trading of service ecosystem BG WP3.MARKETPLACE.001 One Stop Shop
Involved actors	Service Customer
Detailed Operational Description	<p>➤ At the beginning of this use case, a customer is in demand of a service matching his needs and expectations.</p> <p>➤ At the end of this use case, a customer has found a service that fulfils his needs and expectations.</p> <p>A customer connects to the marketplace website.</p> <p>Marketplace contains so many services that it is not practical to browse trough all of them in order to find the right one.</p> <p>He enters a set of criteria and lets the marketplace find matching services.</p> <p>There are still a few services remaining. He looks at the first one. The marketplace displays additional information about the chosen service. There are also links to similar services. He decides to navigate to another service.</p> <p>This time he has found a service that could match his expectations. But he has never heard of this service and needs advices and reassurance.</p> <p>He decides to look at comments left on the site by previous customers. Some customer seems to have had a good experience with this service.</p> <p>He is now reassured and decides that he will subscribe to the service.</p>
Problems and challenges	<p>Support for trading of services (vs. Applications)</p> <p>Pertinent search method</p> <p>Efficient ways to populate suggestion database</p> <p>Ease of use suitable for mass-market user</p>

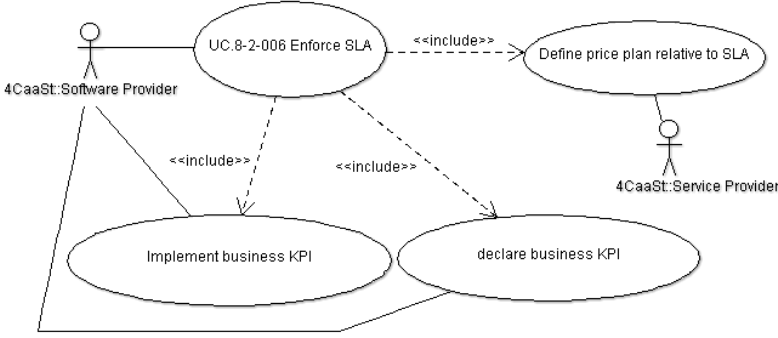
Field	Description
Additional material	<pre> graph LR Actor[4CaaSt:Service "Customer"] --> UC1((UC.8-2-004 Choose service in the marketplace)) UC1 -.-> <<include>> UC2((Search for a service)) UC2 -.-> <<extend>> UC3((Find related services)) UC1 -.-> <<include>> UC4((Seek customer's opinion about a service)) </pre> <p>The diagram illustrates the use case 'UC.8-2-004 Choose service in the marketplace'. It features an actor '4CaaSt:Service "Customer"' who initiates the process. The main use case is 'UC.8-2-004 Choose service in the marketplace'. This use case includes two sub-use cases: 'Search for a service' and 'Seek customer's opinion about a service'. The 'Search for a service' use case further includes 'Find related services' via an extend relationship.</p> <p>Figure 10. UML diagram for use case “Choose service in the marketplace”</p>

7.5. Buy service from the marketplace

Field	Description
Unique ID	UC.8-2-005
Short name	Buy service from the marketplace
Related to	BG WP82_001 support trading of service ecosystem BG WP2.BLUEPRINT.002 Application Lifecycle Management BG WP3.MARKETPLACE.001 One Stop Shop BG WP4.DEPLOY.001 Automated Provisioning and Operation for... BG WP7.SM.001 Cloud aware software platforms
Involved actors	Service Customer
Detailed Operational Description	<p>➤ At the beginning of this use case, a customer has selected a service that he wishes to subscribe to</p> <p>➤ At the end of this use case, a customer has a contract granting him access to the service he selected and he has gained access to the service.</p> <p>Customer is on the marketplace and has decided to buy a service. He is on a page displaying the service.</p> <p>He has selected a service but there are still configurable items such as sub-services used or price plan.</p> <p>Marketplace provides hints about what would be the best priced combination for him.</p> <p>Customer decides based on this information and proceeds to subscription.</p> <p>Marketplace proceeds to service SaaS provisioning.</p> <p>Sub services get informed of customer purchase (telco, CaaS).</p> <p>Customer is informed of delivery evolution.</p> <p>Customer gets access to the service.</p> <p>Customer receives a bill and gets charged.</p>
Problems and challenges	Flexibility vs. complexity of configuration
Additional material	<pre> graph TD Actor[4CaaS:Service "Customer"] --- UC((UC.8-2-005 Buy service from the marketplace)) UC -.-> <<include>> CS([Configure service]) UC -.-> <<include>> CS2([Contractualize service]) CS2 -.-> <<include>> GSA([Get service access]) GSA -.-> <<include>> GC([Get Charged for service]) </pre> <p>Figure 11. UML diagram for use case “Buy service from the marketplace”</p>

7.6. Enforce SLA

Field	Description
Unique ID	UC.8-2-006
Short name	Enforce SLA
Related to	BG WP82_002 Lower marginal hosting cost BG WP82_003 No limit scalability and reduced development cost BG WP2.BLUEPRINT.001 Blueprint: Abstract application design BG WP3.MARKETPLACE.001 One Stop Shop BG WP4.DEPLOY.001 Automated Provisioning and Operation for... BG WP4.ELASTIC.001 Platform Elasticity WP5.MON.001 Integrated Monitoring BG WP7.SM.001 Cloud aware software platforms
Involved actors	Software developer, Service Provider
Detailed Operational Description	<p>➤ At the beginning of this use case, a service is available in the marketplace but KPI and SLA have not been defined and are not enforced.</p> <p>➤ At the end of this use case, KPI is defined and measured and SLA is defined and enforced.</p> <p>Software developer is developing an application such as “TOURISM demo application”. It is decided what are the meaningful key performance indicator of the application.</p> <p>These indicators are business related and need some level of support from the application. As an example, it could be that a special back office process should run in less than 15 minutes, or that a page rendering should be less than so many millisecond or whatever.</p> <p>Software developer develops the KPI probe as he sees fits in the application. He develops the needed glue code so that the 4CaaS platform can get informed of the value of the KPI.</p> <p>Software developer extends the application descriptor so that 4CaaS knows that a KPI probe is provided by the application.</p> <p>He provides meta-data information about how his application nodes can be scaled up or not^{vii}</p> <p>The marketplace is automatically aware of the existence of the KPI.</p> <p>In the marketplace, service provider can specify SLA based on KPI constraints associated with a price plan. Some customer may choose stronger SLA constraints than others and thus be charged different prices.</p> <p>When application is deployed in the marketplace, 4CaaS is aware of the KPI and setups the environment so the KPI value will be picked up by the monitoring software.</p> <p>He is able to inform 4CaaS if his custom component (i.e. Similar but not a 4CaaS built in component such as SCALARIS or Jonas) has horizontal scalability capability and under which limits. He is able to inform 4CaaS of his component multi-tenancy model.</p>

	<p>4CaaS is aware of the desired SLA, is able to monitor the KPI values and can detect if a SLA breach is happening or may happen in a near future.</p> <p>4CaaS takes appropriate actions to ensure SLA respect. 4CaaS optimizes resource usage in order to reach SLA at a minimal cost.</p> <p>Minimal cost shall be proportional to load. In other word, if load doubles, cost is expected to double and if load is divided by a factor of 2, cost is expected to be divided by 2 as well.</p> <p>4CaaS scalability will be evaluated using the following scenario.</p> <p>First, We need to develop some additional code:</p> <ul style="list-style-type: none"> • Implement a basic load injector to simulate many parallel page loads on the “TOURISM demo application” web component, in order to simulate load on context, on J2EE server on SCALARIS key/value store. • Add a dummy load simulator in the page generator in java to ease load simulation on a J2EE server. It could be used to arbitrarily consume variable quantities of processing power or memory. • Extend Flexiant platform with a new fine grained resource usage report. This report would provide per virtual machine report on a given time interval and allow fine grain analysis of 4CaaS resource consumption. <p>Then, a 3-steps test is performed.</p> <ul style="list-style-type: none"> • First a load factor of 100 will be applied to the platform and resource consumption will be measured (RC100). • Then load factor will be lowered to 50 and resource consumption will be measured again (RC50) • Finally, load factor will be raised back to 100 and resource consumption will be measured one last time (RC100back) <p>Analysis of 4CaaS platform resource consumption will provide insight of 4CaaS scalability efficiency.</p>
Problems and challenges	<ul style="list-style-type: none"> • Flexibility vs. complexity of KPI and SLA definition • Reliable automation
Additional material	 <pre> graph TD Actor1[4CaaS:Software Provider] Actor2[4CaaS:Service Provider] UC1((UC-8-2-006 Enforce SLA)) UC2((Define price plan relative to SLA)) UC3((Implement business KPI)) UC4((declare business KPI)) Actor1 --> UC1 Actor1 --> UC3 Actor2 --> UC2 Actor2 --> UC4 UC1 -.-> <<include>> UC2 UC1 -.-> <<include>> UC3 UC1 -.-> <<include>> UC4 UC3 -.-> UC4 </pre> <p>Figure 12. UML diagram for use case “Enforce SLA”</p>

7.7. Enforce metering

Field	Description
Unique ID	UC.8-2-007
Short name	Enforce metering
Related to	BG WP82_002 Lower marginal hosting cost BG WP82_003 No limit scalability and reduced development cost BG WP2.BLUEPRINT.001 Blueprint: Abstract application design BG WP3.MARKETPLACE.001 One Stop Shop BG WP4.DEPLOY.001 Automated Provisioning and Operation for... BG.WP5.MON.002 Integrated Accounting of services BG WP7.SM.001 Cloud aware software platforms
Involved actors	Software developer, Service Provider
Detailed Operational Description	<p>➤ At the beginning of this use case, a service is available in the marketplace without metering capabilities.</p> <p>➤ At the end of this use case, a service is available in the marketplace and its price model includes business and/or technical metering data.</p> <p>Software developer is developing an application such as “TOURISM demo application”.</p> <p>4CaaS marketplace is providing built-in meters. Although useful, these meters are not sufficient for charging a SaaS service.</p> <p>It is decided that it is meaningful to charge customer based on the number of pages viewed. but only those that contain more than a certain number of characters. Pages that are one-click away from a page charged in the last 48 hours are not charged. A page that has been displayed in the last 48 hour is not charged again.^{viii}</p> <p>This example meter, like many real life meters, requires applicative knowledge from the application, and thus must be implemented by the application providing the service.</p> <p>Software developer develops the meter probe as he sees fits in the application. He develops the needed glue code so that the 4CaaS platform can get informed of the value of the meter.</p> <p>Software developer extends the application descriptor so that 4CaaS knows that a meter is provided by the application. (He provides meta-data about application specific meters)</p> <p>The marketplace is automatically aware of the existence of the meter.</p> <p>In the marketplace, service provider can specify price plan taking into account the value of the meter. Some customers may have a small or big plan and get different pricing scheme for usage of the application.</p> <p>When application is deployed in the marketplace, 4CaaS is aware of the meter and setups the environment so the meter value will be picked up by the metering infrastructure.</p>
Problems and challenges	<ul style="list-style-type: none"> • Flexibility vs. complexity of meter definition and price plan definition • Aggregation strategies and handling of billing periods • Freshness & reliability of metering vs. IAAS cost of metering

Field	Description
Additional material	<pre> graph TD UC82007([UC-8-2.007 Enforce metering]) SProv([4CaaS:Software Provider]) SProvServ([4CaaS:Service Provider]) IM([Implement business meters]) DPM([Define price plan relative to business meters]) DBM([declare business meters]) SProv --- UC82007 SProvServ --- UC82007 UC82007 -.-> <<include>> IM UC82007 -.-> <<include>> DPM UC82007 -.-> <<include>> DBM SProv --- IM SProv --- DBM SProvServ --- DPM </pre> <p>Figure 13. UML diagram for use case “Enforce metering”</p>

8. Evaluation method

The content of this section has been moved to deliverable D8.4.1 Reports on Experimentation, Experimentation Guidelines [5]

9. Conclusion

This document is one out of three documents describing the use cases planned to be implemented on top of 4CaaS in order to evaluate the added value envisioned by the project in some context. All use case documents follow the same structure, while the overall value propositions and evaluation method are not part of the documents, but are outlined in the 4CaaS Value Proposition Whitepaper [1].

This document has been updated following release of experimentation report [5]. It contains an updated description of the prototype components, a new section for prototype configuration, and miscellaneous updates of the scenario use cases.

The next steps are targeted towards planning and starting partial implementation of the use cases based on the results from the technical WPs. The description of what is implemented will be part of the next iteration of this deliverable with a map of features that have been used. Before that the evaluation criteria for the features are to be defined based on evaluation method.

10. References

- [1] 4CaaSt Value Proposition Whitepaper
- [2] [SCUBE,2010], <http://www.s-cube-network.eu/news/s-cube-use-case-description-methodology>
- [3] [NEXOF-RA, 2009], <http://www.nexof-ra.eu/>
- [4] [Jackson,1997] Four Dark Corners of Requirements Engineering, P. Zave and M. Jackson, 1997, <http://dl.acm.org/citation.cfm?id=237434>
- [5] D8.4.1 Reports on Experimentation, Experimentation Guidelines, Lead Editor: Michel Dao, FT, July 2012
- [6] D8.2.5, Experimentation_Report_v1.0
- [7] D 8.2.2 Use Case - Virtual private Cloud for mass-market: Scenario Definition
- [8] D1.1.2 Analysis of the State of the Art and Definition of Requirements
- [9] D1.2.1 Conceptual Model and Reference Architecture
- [10] D1.3.1 Integration of Results
- [11] D6.2.1 Native PaaS Technologies: Components Design and Open Specification

Annex A. Full list of Global Business Goals

This section contains all global business goals defined for 4CaaSt.

A.1. BG#WP2.BLUEPRINT.001

Field	Description
Unique ID	BG#WP2.BLUEPRINT.001
Short name	Blueprint: Abstract application design
Type	Business goal
Description	<p>A formalism shall be provided for specifying application software architecture and dependencies toward other components and technology enablers.</p> <p>The system shall enable the software providers to design applications describing the resources (components and technology enablers) they need to deploy in an abstract way, providing the functional and non functional constraints. This way, the software providers will be able to support an abstract application design that:</p> <ul style="list-style-type: none">• allows to offer software “service” in a generic way and• “use” software “service” from other providers. <p>This mechanism should be supported by a composition tooling that will help and drive the developer through the process specifying all the variants that make sense for a given application.</p>
Rationale	Lower cost and length of development and maintenance by enabling software providers to specify the reuse of existing software components (accessible either in an IaaS or SaaS way) and to focus on his part of a solution in which he is the expert.

Field	Description
Involved Stakeholder	Service Provider, Software Provider
Supporting materials	
Priority of accomplishment	Must have
Tentative scheduling	RP1 (first iteration, focus on functional aspects) RP2 (second iteration, focus on scalability and other non-functional) RP3 (third iteration, focus NaaS)

A.2. BG#WP2.BLUEPRINT.002

Field	Description
Unique ID	BG#WP2.BLUEPRINT.002
Short name	Application Lifecycle Management
Type	Business goal
Description	Platform shall provide full automation of application deployment, configuration, monitoring, healing, optimization using formalization data provided by the software. The system shall help the software provider in the management of the lifecycle of their applications. Basically, the software provider would concentrate only on the development phase, while the deployment, taking into account the service providers' and the customers' preferences and policies would be handled by the system itself. This includes deciding which concrete services and resources to use for an application both from the functional and non functional points of view, and both from the technical and the business point of view.
Rationale	Lower cost and length of deployment by enabling the software providers to subcontract the deployment management of their applications to the system. Be able to satisfy a broader spectrum of customers by enabling the service providers and customers to get customized deployments.
Involved Stakeholder	Service Provider, Software Provider, Customer
Supporting materials	
Priority of accomplishment	Must have
Tentative scheduling	RP1 (first iteration, focus deployment) RP2 (second iteration, focus scalability) RP3 (third iteration, focus NaaS)

A.3. BG#WP3.MARKETPLACE.001

Field	Description
Unique ID	BG#WP3.MARKETPLACE.001
Short name	One Stop Shop
Type	Business goal

Description	<p>4CaaS platform shall support the trading of any kind of Cloud Service (SaaS,PaaS,IaaS) in a unified way through the abstract description of the applications. This unified marketplace would handle different type of usage models for different types of roles:</p> <ul style="list-style-type: none"> • A service provider may sell a running service to be used as SaaS in a multi-user way. • A software provider may sell an application to be deployed over concrete resources after customer contract. • A customer may contract the access to an application or the deployment of an application for own usage. • Etc. <p>All the resources of the application, together with the own application, may be provided by different software/service providers and the generated incomes would be distributed among them according to the specified policies.</p>
Rationale	<p>Be able to satisfy a broader spectrum of service providers and customers by providing flexible configuration of cloud (XaaS) services and supporting multiple business models.</p> <p>Allow service providers to benefit from automatic settlement for the usage of their services.</p>
Involved Stakeholder	Service Provider, Software Provider, Customer, End User
Supporting materials	
Priority of accomplishment	Must have
Tentative scheduling	<p>RP1: publication, contracting</p> <p>RP2: social enhancement, settlement</p> <p>RP3: improved business models</p>

A.4. BG#WP3.MARKETPLACE.002

Field	Description
Unique ID	BG#WP3.MARKETPLACE.002
Short name	Business customization
Type	Business goal
Description	Customers shall be able to configure how the applications they have contracted are going to be deployed or provisioned. Customers will define several non functional business related options that will be used to decide which of the possible solutions to deploy fulfils better their preferences.
Rationale	Be able to satisfy a broader spectrum of customers by enabling them to customize their contracted applications.
Involved Stakeholder	Service Provider, Customer
Supporting materials	
Priority of accomplishment	Must have
Tentative scheduling	<p>RP1: basic preferences</p> <p>RP2: role based customization</p> <p>RP3: dynamic roles based on history</p>

A.5. BG#WP4.DEPLOY.001

Field	Description
Unique ID	BG#WP4.DEPLOY.001
Short name	Automated Provisioning and Operation for Heterogeneous Components
Type	Business goal
Description	<p>Application component and its required services and resources shall be able to choose among different platforms to be deployed to and be operated automatically.</p> <p>Based on this selection 4CaaSt will be able to automatically generate suitable deployment designs and automatically provision the corresponding resources. Resource provisioning will include selection of the most appropriate deployment designs considering resource, QoS as well as scalability requirements, the automated technical construction and configuration of virtual machines and their final deployment.</p>
Rationale	<p>Automated deployment reduces the operational and capital invests. An application provider must not care how the underlying are build, deployed and operated and therefore concentrates on the application/service development.</p> <p>Automated operation contributes reduced operational cost; the application provider must not care about all required services; the PIC provider must not care about the REC</p>
Supporting materials	<p>Referred to as</p> <ul style="list-style-type: none"> • “Resolution process in combination with Blueprints” in the list of WP2 Innovations and • “Automated VM Construction and Provisioning for Heterogeneous Components” on WP4 Innovations
Priority of accomplishment	Must have
Tentative scheduling	RP1, RP2

A.6. BG#WP4.ELASTIC.001

Field	Description
Unique ID	BG#WP4.ELASTIC.001
Short name	Platform Elasticity
Type	Business goal
Description	<p>An application component running on top of 4CaaS shall be elastic meaning that it (horizontally and vertically) scales automatically while the 4CaaS platform will use the minimal required resources to achieve the application's required SLA.</p> <p>An application component deployed on a 4CaaS cloud will scale automatically and with the minimal resources consumption in terms of virtual machines used within given boundaries such as</p> <ul style="list-style-type: none"> • defined SLAs • defined scale options of the AC (stateful, stateless, etc.) • defined scale/deployment options of the underlying PICs <p>Minimal resource means that the selected deployment will try to use the deployment that uses the least resources to fulfil the defined SLA (i.e. prefer a one low end machine setup to a high-end load balanced setup)</p> <p>One way of minimizing the resource consumption could be for some workload introducing multi-tenancy.</p>
Rationale	Reduce operational cost by enforcing SLAs while using minimal resources.
Supporting materials	Referred to as "Elasticity in the PaaS Layer" in the list of WP4 Innovations
Priority of accomplishment	Must have
Tentative scheduling	RP2 (stateless), RP3

A.7. BG#WP4.NAAS.001

Field	Description
Unique ID	BG#WP4.NAAS.001
Short name	NaaS Services - Layer 2/3 virtual networks for the deployed applications
Type	Business goal
Description	<p>Applications and services with a defined SLA need to be able to request certain SLAs from the underlying network that connects it's components via network links within or across data centers. For some applications these network links may require certain functional and non-functional qualities, such as secure links, certain network latency or bandwidth guaranties as they effect the high level Application/Service SLAs. For example an application SLA that promises 99.9% availability requires at least 99.9% availability of all network links between the components. Therefore clients must be enabled to customize the defined networks (private or public, IP range, QoS requirements like throughput or latency) and dynamically assign virtual machines to the defined networks.</p> <p>IaaS Clouds provide fundamental computing resources to the consumer including processing, storage and network resources. Thus, logically NaaS is part of the IaaS offer. However, mainstream state-of-the-art IaaS Clouds provide only Layer 3 networking options that will not be sufficient for complex applications, especially in enterprise and telecommunications domains.</p> <p>Many use cases including legacy applications require functionalities like broadcast traffic, multicast networking, control over the IP address range used, multiple networks accessed by an application, guaranteed quality of service for network connections, etc. These functions are not provided by the IaaS Layer 3 network: the possibility to allow IaaS clients to have Layer 2 networks for their own purposes is needed.</p>
Rationale	Guarantee Application or Services level SLA for distributed applications.
Supporting materials	Referred to as "NaaS" in the list of WP4 Innovations
Priority of accomplishment	Must have
Tentative scheduling	RP2 (Design / Models) RP3 (Implementation)

A.8. BG#WP5.MON.001

Field	Description
Unique ID	BG#WP5.MON.001
Short name	Integrated Monitoring
Type	Domain Assumption
Description	<p>The system shall collect custom and built-in KPIs for any resource deployed and managed.</p> <p>This means that the system shall support capturing and aggregating specific KPIs for the different layers of the cloud stack (S/P/I/aaS), in order to enable advanced management capabilities, like scaling.</p>

	In order to help developers to deploy monitorable applications, a set of tools will be provided to specify, trigger and generate the appropriate monitoring information
<i>Rationale</i>	Enable manageability of cloud applications and the resources they use.
<i>Involved Stakeholder</i>	Service Provider, Software Providers, Platform owner.
<i>Supporting materials</i>	None
<i>Priority of accomplishment</i>	Must have
<i>Tentative scheduling</i>	Iterative increments throughout the project.

A.9. BG#WP5.MON.002

Field	Description
<i>Unique ID</i>	BG#WP5.MON.002
<i>Short name</i>	Integrated Accounting of services
<i>Type</i>	Business Goal
<i>Description</i>	The system shall provide accounting for custom and built-in counters involving usage of the platform. This will allow for example detailed rating and bills to be generated. Different types of accounting capabilities can be provided so that developers and providers can make use of them and define their price models accordingly.
<i>Rationale</i>	Allow customers to have lower bills by enabling pay-per-use of services
<i>Involved Stakeholder</i>	Service Provider, Software Provider, Customer.
<i>Supporting materials</i>	None
<i>Priority of accomplishment</i>	Must have
<i>Tentative scheduling</i>	RP2

A.10. BG#WP6.SM.001

Field	Description
Unique ID	BG#WP6.SM.001
Short name	Integration of native services
Type	Business goal
Description	The software providers shall have the possibility of using external existing services and APIs in an easy way. A representative portfolio of services is needed, together with the mechanisms to easily integrate them in the applications. Software provider shall not have to manage those services nor their lifecycle.
Rationale	Lower cost and length of development and maintenance by enabling developers to integrate existing services without cost.
Involved Stakeholder	Software developers
Supporting materials	None
Priority of accomplishment	Must have
Tentative scheduling	RP2

A.11. BG#WP7.SM.001

Field	Description
Unique ID	BG#WP7.SM.001
Short name	Cloud aware software platforms
Type	Business goal
Description	The system shall provide a set of well-known platform technologies available for use by software providers using basic configuration. Therefore, software providers will not have to worry about the management of those technologies (installation, deployment of components or packages, configuration, etc.). Moreover, the software providers shall benefit from scaling and multi-tenancy features of those platforms. 4CaaSt platform shall provide facilities to integrate application servers, orchestration engines, relational databases and integration technologies like ESBs.
Rationale	Lower cost and length of development and maintenance by providing a set of cloud-aware well known technologies to support software development.
Involved Stakeholder	Software developers
Supporting materials	Description of Work
Priority of accomplishment	Should have
Tentative scheduling	Several examples of technologies must have integrated by the end of the project.

Annex B. Glossary

Application: An Application is the software that has been developed based on a given Platform and, therefore, can be deployed and executed (hosted) on any 4CaaS-compliant Cloud that supports such a Platform. An Application is composed by Application Components.

Blueprint: A Blueprint is the descriptor of an application component/artifact (i.e. AC, PIC...) available for a 4CaaS-compliant Cloud. The Blueprint contains information about the component/artifact itself (e.g. its name) its abstract offerings (e.g. a servlet container) and its requirements (i.e. a Java runtime).

Abstract Resolved Blueprint / Application Blueprint: An Abstract Resolved Blueprint / Application Blueprint is the descriptor of an application stack hosted in a 4CaaS-compliant Cloud. It's build from individual Blueprints, their offerings and requirements that are matched and results in a set of potential deployments.

Application Component: An Application Component is one of the building blocks in which the software of a given application is structured. For example, web pages, EJBs, servlets, workflow programs, etc.

Application Component Instance: An Application Component is running on top of a specific Product Instance.

Enabler: An Enabler provides/supports the offered Platform Service, allowing Application Providers to access some functionality (e.g. DBRepository, Telco Services, etc.). The Enabler facilitates the Product Instance/s to be used. Each Enabler must provide the corresponding API.

Mashup: A mashup uses and combines data, presentation or functionality from two or more sources/services to create new services. Examples maybe Apps (Android/Apple) that combines services available from the Internet to provides value added services.

Network Virtualization: In computing, Network Virtualization is the process of combining hardware and software network resources and network functionality into a single, software-based administrative entity, a virtual network. Network virtualization involves platform virtualization, often combined with resource virtualization.
(Source: http://en.wikipedia.org/wiki/Network_virtualization)

Platform: A Platform is a suite of Platform Technologies.

Platform Technology: A technology that can be used to develop Application Components. A J2EE container, a BPEL container, an elastic data store, a Cloud RDBMS or an SMS/MMS service are examples of Platform Technologies. A Platform Technology is linked to a well-defined set of APIs used to program Application Components. Note that Platform Technologies are not linked to particular products (e.g., Oracle BPEL engine, Apache Tomcat or PostgreSQL) but well defined APIs. Some of these technologies can be considered "native" to the Cloud, since they were developed and flourished upon the advent of this new conception of computing. Other, "immigrant", technologies are well-established technologies available before the age of Cloud computing (e.g. application servers, database systems, composition engines, ESBs), which need to be adapted to suit the requirements imposed in this new environment and become first-class Cloud technologies.

Product: A Product implements a given Platform Technology (E.g. Jonas, JBoss, Apache Web Server, etc.).

Product (eMarketplace): A Product in the eMarketplace is a commercial offer that results from adding specific commercialization conditions to the actual functionality (service or application).

Platform Instance (PI): A combination of Platform Instance Components (PIC) which materialize all technologies linked to a given Platform.

Product Instance Component (PIC): A product which on its own, or combined with other Product Instances, implements a given Technology. Examples of Product Instances

would be Jonas, JBoss, Apache Web Server, Apache Tomcat, PostgreSQL, RedHat Linux, etc.

Virtual Platform: A combination of Product Instances which materialize all technologies linked to a given Platform.

Runtime Execution Container (REC): It represents the VM, together with the runtime images associated to a set of Product Instances, properly configured to support the execution of a number of Application Components Instances.

Runtime Execution Container (REC) Manager: An entity in charge of the management of existing RECs in the Cloud.

ScaleOut-ScaleIn: ScaleOut means adding more computing nodes to a system and distribute software application with potentially some added resource such as a Load Balancer. The opposite of ScaleOut is ScaleIn.

ScaleUp-ScaleDown: ScaleUp means adding resources to a single node in a system, such as more CPUs, more memory or more network bandwidth. The opposite of ScaleUp is ScaleDown

Service: Piece of software that offers functionality ready for integration into an application. A service is to be used by developers.

Storage Object: It is the abstraction of any storage resource accessible through its URI. A container of data regardless the type or format of the data it contains.

Technology: See Platform Technology.

User & User Roles: See D1.1.1(b) User and User roles

Virtual Machine: An instance of a virtual computer, hosting a functional operating system.

Virtual Network: A virtual network is a computer network that consists, at least in part, of virtual network links. A virtual network link is a link that does not consist of a physical (wired or wireless) connection between two computing devices but is implemented using methods of network virtualization. (Source http://en.wikipedia.org/wiki/Virtual_network)

Virtual Private Network (VPN): A virtual private network (VPN) is a computer network that uses a public telecommunication infrastructure such as the Internet to provide remote offices or individual users with secure access to their organization's network. It encapsulates data transfers between two or more networked devices which are not on the same private network so as to keep the transferred data private from other devices on one or more intervening local or wide area networks. (Source http://en.wikipedia.org/wiki/Virtual_network)

Virtual Platform: The Virtual Platform represents the instantiation of a resolved Platform. It is composed by Virtual Products which execute the Application Components. The Virtual Platform provides a uniform view of the execution context of an Application.

Virtual Product: A Virtual Product is a proxy interface to the Product Instance(s) that executes the Application Components. It allows configuring and deploying Application Components.

-
- i From D8.2.5, this element is not relevant to this use case. It can be found in use case 7.
 - ii From D8.2.5, this element is not relevant to this use case. It can be found in use case 6.
 - iii From D8.2.5, this element is not relevant to this use case. It can be found in use case 6.
 - iv From D8.2.5, this feature is out of the scope of 4CaaSt although it would be useful in a real life platform.
 - v From D8.2.5, this is not needed in this base use case and is anyway covered in use cases 6 and 7.
 - vi From D8.2.5, a new name for scenario is suggested. We keep both name in title to ease mapping from other documents.
 - vii From D8.2.5, this element is not relevant to this use case. It can be found in use case 6.
 - viii From D8.2.5, implementing such metering strategy requires applicative logic that is not in the scope of 4CaaSt. We remove this section as it is application specific and will not be done.