CloudSocket

EXPLANATORY NOTES: FIRST BPAAS PROTOTYPE

D4.2 - D4.3 - D4.4

Editor Name	Joaquin Iranzo (ATOS)
Submission Date	June 30, 2016
Version	1.0
State	FINAL
Confidentially Level	PU

Co-funded by the Horizon 2020
Framework Programme of the European Union

EXECUTIVE SUMMARY

This document acts as an explanatory report to introduce the first BPaaS prototype, which integrates the complete BPaaS lifecycle through the different BPaaS environments, while it unifies and summarises the prototype deliverables: i) D4.2 First BPaaS Design and Evaluation Environment, ii) D4.3 First BPaaS Execution Environment and iii) D4.4 First BPaaS Allocation Environment.

The BPaaS Environments that support the BPaaS lifecyle are (a) BPaaS Design Environment, (b) BPaaS Allocation Environment, (c) BPaaS Execution Environment, and (d) BPaaS Evaluation Environment. Additionally the BPaaS Marketplace is required to enable the BPaaS Customer to find and buy the BPaaS bundle. All environments are independent; nevertheless, they expose different interfaces to be integrated and define the structures of the interchange data. This means that the first prototype has started to be integrated and aligned from the very beginning of the development phase. This has allowed to harmonize the different interfaces and interactions, which have been clearly defined in the CloudSocket architecture (deliverable D4.1 [1]), allowing to have a complete lifecycle including all the BPaaS environments.

In order to harmonize the implementation, a simple business process has been introduced as a guiding example for all the phases, being the main thread that runs through the complete lifecycle. This process concerns the sending of Christmas Greeting cards by email automatically. It is simple and includes all the basic concepts allowing the integration of the BPaaS first prototype, fluidly.

As this deliverable deals with the reporting of a software prototype, it includes: i) The definition of all software pieces, i.e., components, in the component wiki [19] and the respective sources to be downloaded (in case it is open-source) from the GitLab repository [20], iii) a summary documentation for the components drawn from their complete specification in the component wiki and ii) details about component instances offered as SaaS (e.g., actual endpoint), deployed on the UULM and YMENS infrastructure. Besides, it is complemented by demonstration videos [29] and web presentation [30] that allow understanding the BPaaS prototype but not only having a list of software components.

The document introduces the First BPaaS/CloudSocket prototype from different perspectives based on the main actors (BPaaS Customers and Brokers) who interact with the platform in the context of the respective running example, as it is shown in the demo videos [29]. Afterwards, the main deliverable sections include a basic level of detail for the different environments by also supplying links catering for a more detailed inspection, such as the wiki component description [19], source repository [20] and instances as SaaS. Therefore, the document explains first the demonstration and then the details of the respective environments, allowing different kinds of readers to understand it.

The structure of the document leads the reader from the prototype introduction to an in-depth view of all environments structured by the two perspectives to cover the complete lifecycle. It is not the intention to introduce all the content of this live documentation into this document; therefore, this document only contains the most relevant information to understand the prototype and the different environments, following the same structure for all of them.

PROJECT CONTEXT

Workpackage	WP4: BPaaS Implementation
Task	T4.2 – T4.3 – T4.4
Dependencies	Input to D4.1, WP3, WP5, WP8

Contributors and Reviewers

Contributors	Reviewers
Daniel Seybold, Frank Griesinger (UULM) Kyriakos Kritikos (FORTH) Antonio Gallo, Simone Cacciatore (FHOSTER) Radu Davidescu, Alex Ganga (YMENS) Joaquin Iranzo, Roman Sosa (ATOS) Wilfrid Utz, Damiano Falcioni (BOC)	Internal Review 16.06.2016: Kyriakos Kritikos (FORTH), Antonio Gallo, Simone Cacciatore (FHOSTER), Daniel Seybold, Frank Griesinger (UULM), Joaquin Iranzo, Roman Sosa (ATOS), Damiano Falcioni (BOC) Final Review 23.06.2016: Robert Woitsch (BOC), Simone Naldini (MATHEMA), Diana Irimia (YMENS)

Approved by: Joaquin Iranzo (ATOS) as WP4 Leader

Version History

Version	Date	Authors	Sections Affected
0.1	May 03, 2016	Joaquin Iranzo	Initial version, TOC
0.2	May 13, 2016	Joaquin Iranzo	Agreement of the TOC and the content, including the partners' comments.
0.3	May 30, 2016	Joaquin Iranzo, all contributors	All the sections, inclusion of the first version to consolidate the contributions.
0.6	June 15, 2016	Joaquin Iranzo, all contributors	First version for internal review.
0.9	June 21, 2016	Joaquin Iranzo, all contributors	Consolidation for final version to be reviewed
1.0	June 23, 2016	Joaquin Iranzo, all contributors	Final version to be reviewed
2.0	June 30, 2016	Joaquin Iranzo, all contributors.	Final version

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

This document explains the First BPaaS/CloudSocket Prototype, integrating the complete BPaaS lifecycle and delivering the platform with all the respective environments involved. This document is an addendum of the three official deliverables:

- D4.2 First BPaaS Design and Evaluation Environment (M18) - First prototype implementing the BPaaS design and evaluation/analysis functionality according to requirements.
- D4.3 First BPaaS Execution Environment (M18) - First prototype implementation of the BPaaS adaptive provisioning functionality according to requirements
- D4.4 First BPaaS Allocation Environment (M18) - First prototype for the CloudSocket allocation modelling and BPaaS publishing functionality for the CloudSocket broker

The original reason to have one consolidated deliverable is to be uniform and have a complete picture for the status of the whole prototype, although the three prototypes that have been delivered separately in form of D4.2, D4.3 and D4.4. The content of these documents can be identified clearly in the following sections:

- The content of D4.2 can be found mainly inside the analysis of the Broker perspective in Section 4, which includes two main and related subsections: Section 4.3.1 for the BPaaS Evaluation Environment and Section 4.3.2 for the BPaaS Design Environment.
- The content of D4.3 is covered mainly by the analysis of the BPaaS Customer perspective in Section 3. The main related subsections are: Section 3.3.1 for the BPaaS Marketplace and Section 3.3.2 for BPaaS Execution Environment.
- The content of D4.4 is related mainly to the analysis of the Broker perspective in Section 4. The main related subsection is Section 4.3.3 mapping to the analysis of the BPaaS Allocation Environment.

Although the environments are independent (aligned with tasks T4.2, T4.3 and T4.4), the project has started to integrate and align them from the very beginning of the development phase. This allows harmonizing the different interfaces and interactions, which have been clearly defined in D4.1 [1], allowing to have a complete lifecycle including all the BPaaS (management) phases.

This harmonisation is enabled through the introduction of a simple business process that is used by all the phases in order to produce, publish, adaptively provision and evaluate the respective BPaaS bundle(s). As such, this process constitutes a common example utilised as the thread that runs through the complete lifecycle. The process concerns the automatic sending of Christmas Greeting cards via email and is shortly analysed in Section 2. This process, although simple, includes all the basic concepts, allowing integrating the platform and provoking discussion to introduce new concepts and functionalities or enhance the existing ones offered, such as the definition of the registries, the common understanding to deploy software components and services in the cloud, and the harmonization of the process definition.

This deliverable is a software prototype, so it includes:

- Access information to the software tools and service developed
- The definition of all software pieces, i.e. components, in the component wiki [18] and the respective sources to be downloaded (in case it is open-source) from the GitLab repository [20]
- Summary documentation for the components drawn from their complete specification in the component wiki [18].

- Details about component instances offered as SaaS (e.g., actual endpoint), deployed on the UULM and YMENS infrastructure.
- Besides, demonstration videos [30] and web presentation [30] in order to facilitate the understanding of the BPaaS prototype.

The document, which describes this deliverable, is not a "standardised" software development technical report which should be read only by software developers. On the contrary, it comprises content which can be understood by different kinds of readers. This is mainly enabled via the introduction of the First BPaaS/CloudSocket prototype from different perspectives based on the main actors (BPaaS Customers and Brokers) who interact with the platform in the context of the respective running example, as it is shown in the demo videos [29]. Moreover, it is also enabled via supplying a basic level of detail for the different environments by also providing links catering for a more detailed inspection for the interested reader.

The readers can access this additional documentation such as the wiki documentation [18], CloudSocket portal [23] [30] (Figure 1), demonstration videos [29] and CloudSocket GitLab repository [20], which complement the document with more detailed and deep analysis for the different environments and components. Such documentation has been the continuous result of the use of online and live tools for the documentation and source repository, as introduced in D5.1 [21]. Taking advantage of these tools, technical descriptions and manuals have been referenced in order to reduce the complexity of the document; allowing reading it incrementally and going deeper into each of the component references.

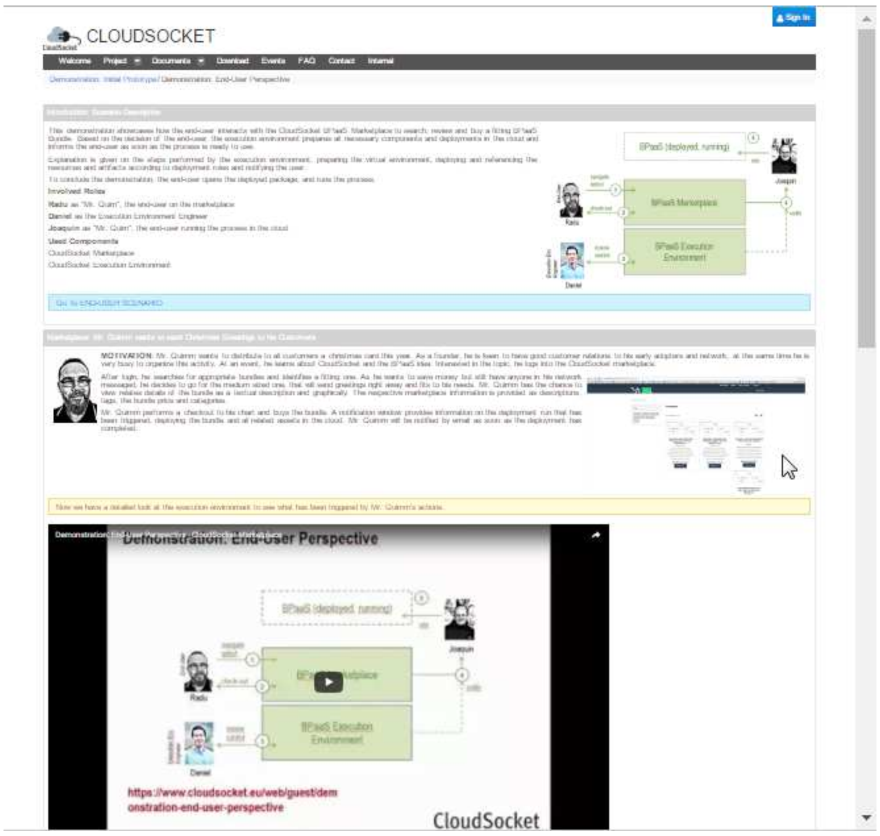

CLOUDSOCKET
Sign In
Welcome
Project
Documents
Download
Events
FAQ
Contact
Internal
Involved Roles
Used Components
BPaaS (deployed, running)
BPaaS Marketplace
BPaaS Execution Environment
Radu
Daniel
Joaquin
Demonstration: End-User Perspective
https://www.cloudsocket.eu/web/guest/demonstration-end-user-perspective
CloudSocket

Figure 1 - CloudSocket portal: demonstration BPaaS Customer perspective

The structure of the document leads the reader from the prototype introduction to the two perspectives in order to cover the complete lifecycle, allowing identifying every environment and component in the correct context. The document comprises the following sections:

- **Section 2 – CloudSocket First Prototype**: this is a core section that summarizes all the components, allowing to identify quickly what the readers need in order to later go into deeper details. Hence, this section presents the First BPaaS prototype in order to (a) understand better the overall integration and platform through the introduction of the running example process for sending Christmas cards to illustrate the idea of a BPaaS and (b) provide instant access to the deployed tools and services. In addition, it aligns the prototype to the architecture defined in D4.1 [1]. Finally, the different actors are introduced to present the two main perspectives that the document brings to the prototype.
- **Section 3 – BPaaS Customer Perspective**: this section covers the different steps of an BPaaS Customer when interacting with the BPaaS prototype. It is structured into two different main subsections in order to facilitate the reading:
  - **Section 3.2 - Demonstration**: The involved roles are explained in order to better understand the different individual role actions as well as the interactions involved. Such interactions are first identified and then shortly analysed. All the references to the CloudSocket Portal and videos of the demonstration [29] have been introduced to enable a more interactive demonstration mode in conjunction to the explanations provided in the main deliverable text.
  - **Section 3.3 - Environments**. After showing the demonstration, this section introduces the involved environments in the different BPaaS management phases related to this demonstration. For all the environments, the components are introduced via the same structural analysis which is complemented by the references for the live documentation (i.e., through the portal, wiki and gitlab) and for the different environment instances offered as a SaaS.
- **Section 4 - Broker Perspective**: this section covers the CloudSocket Broker perspective and especially the use of process analytics to identify potential improvements of an existing BPaaS and the necessary steps to create and publish a new BPaaS bundle into the Marketplace. It is similarly structured as in section 3 in different subsections so as to facilitate the reading.
  - **Section 4.2 - Demonstration**: The same approach, as used in section 3.2- Demonstration for the End-User perspective, is followed for this section, but now focusing on Broker-based respective interactions.
  - **Section 4.3 - Environments**. The same approach, as used in Section 3.3 - Environments for the End-User perspective, is followed for this section.
- **Section 5 - Conclusion:** This section concludes this deliverable and outlines the future work to be conducted and reported in the next and final version of this deliverable (consolidated form of D4.6-4.8 deliverables).

As it has been mentioned, the document is aligned with the wiki documentation and the Gitlab repository. It is not the intention to introduce all the content of this live documentation into this document; therefore, it only contains the most relevant information to understand the prototype and the different environments. Hence, the same analysis structure is followed for all the components. Nevertheless, the environments can have their own peculiarities with respect to the component architecture involved: i) in some cases, the environment maps to an overall component that is offered as a whole (e.g., in the form of a SaaS or tool) and only this component is detailed; ii) in other cases, the environment maps to multiple main components with their own internal architecture and the analysis focuses one all these main components.

This same analysis structure of the components comprises the following 7 parts provided with the order of their subsequent presentation. In some cases, particular parts, like the architecture design, are not provided, as the component may not comprise an internal architecture:

- **Summary:** introduces a general overview, the functionalities, the kind of interfaces as well as the actors and roles involved. It also includes a classification to quickly see the details of the component:
  - Type of ownership: Indicates the type of relation of the owner with this component. The possible values are: i) Creation (indicates that it is a new development to cover specific functionalities), ii) Extension (indicates that a part of existing functionalities is used and extended), and iii) Usage (the component is used directly without any change).
  - Original tool: Indicates the original tool on which this component is based. This is correlated with the "Type of ownership" field, highlighting that an existing component is re-used and mapping to options ii) Extension or iii) Usage.
  - Planned OS License: Type of license either open-source, closed-source or proprietary.
  - Reference community: Indicates the community that can provide support to the component.
  - Lead Partner: Indicates the partner is leading the component.
- **Architecture design:** details the specific internal component architecture.
- **Functionalities:** includes a list of functionalities offered by the component. For each functionality, it is indicated if it is complete for this release and what is its level of integration with the rest of the components (in the same or different environment). For example, the core functionality may be finished but it is pending to be integrated with other components by also having the complete lifecycle tested in the end for verification and debugging reasons.
- **Manuals:** introduces a short explanation, when needed, and references in order to facilitate the reading of more details about the installation guide, the offered API description, handbooks, and unit tests.
- **Download:** includes links to download the code in case it is open- or closed-source.
- **Instances:** includes the links of the instances to be used in the integration environment for the prototype.

2 CLOUDSOCKET FIRST PROTOTYPE

In the following, the software components contained in each environment are introduced using a common factsheet to ease navigation and accessibility. Detailed contextual information for each artefact is available in the documentation provided in Sections 3 and 4.

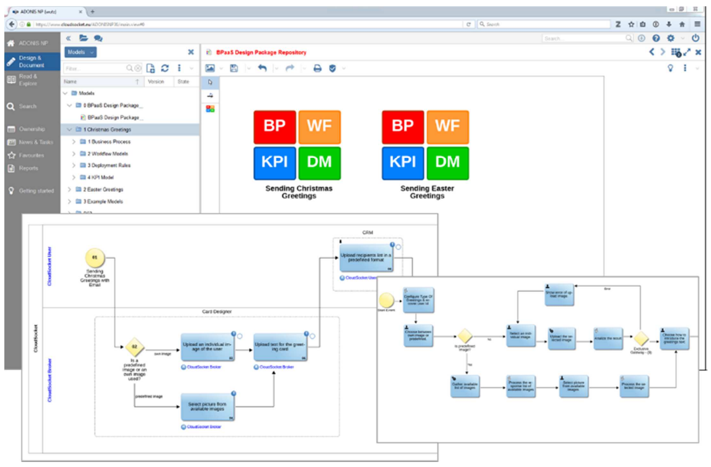
BPaaS Design Environment	
The BPaaS Design Environment has the overall goal to model aspects of a BPaaS by focusing on higher levels of abstraction. This leads to a generation of a BPaaS Design Package which describes an un-allocated BPaaS at the IT/cloud level by including various types of information, such as a domain specific business process model, an executable workflow-model, and a set of KPIs/requirements mapping to these two models. In addition, to enable the re-use of design knowledge as well as the automatic or semi-automatic alignment between business process and workflow models, the BPaaS Design Environment enables the storage, querying and retrieval of all model artifacts generated and their semantic annotation.	
Component	**Description**
BPaaS Design Tool	The BPaaS Design Tool has been created on the base of the CloudSocket metamodel and provides the possibility to model domain-specific business processes, execution workflows, decision models and key performance indicators.  BPaaS Design Package Repository BP WF KPI DM Sending Christmas Greetings BP WF KPI DM Sending Easter Greetings
Access	SaaS Deployment: https://www.cloudsocket.eu/ADONISNP36/ (user credentials on demand) Experimentation Version Download: https://www.adoxx.org/live/web/cloudsocket-developer-space/downloads
License	Closed source
Further Details	Demonstration: Section 4.2.3 Technical Details: Section 4.3.2.1
Manual	https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Design+Environment+Components
Lead Partner	BOC

Table 1 – Prototype Components of Design Environment

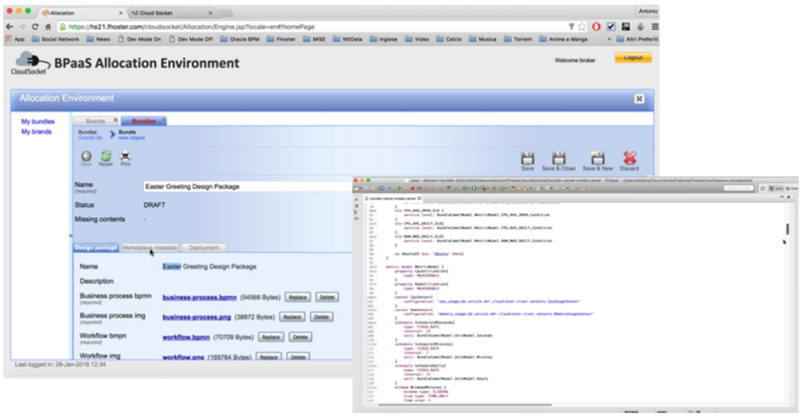
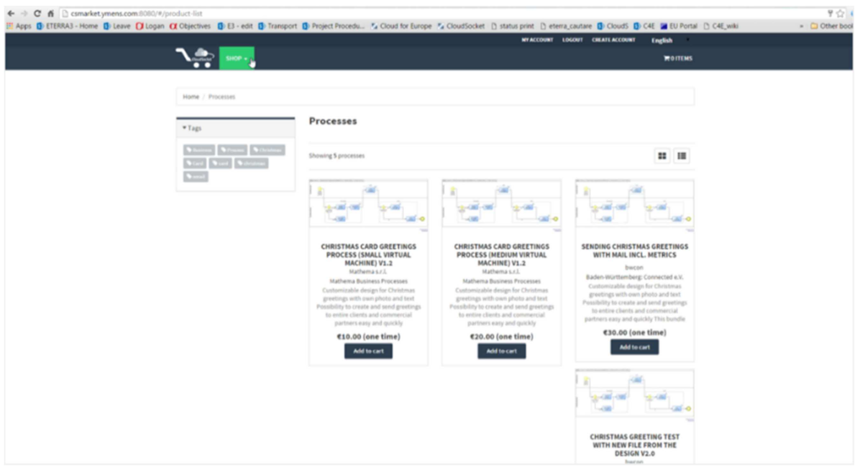
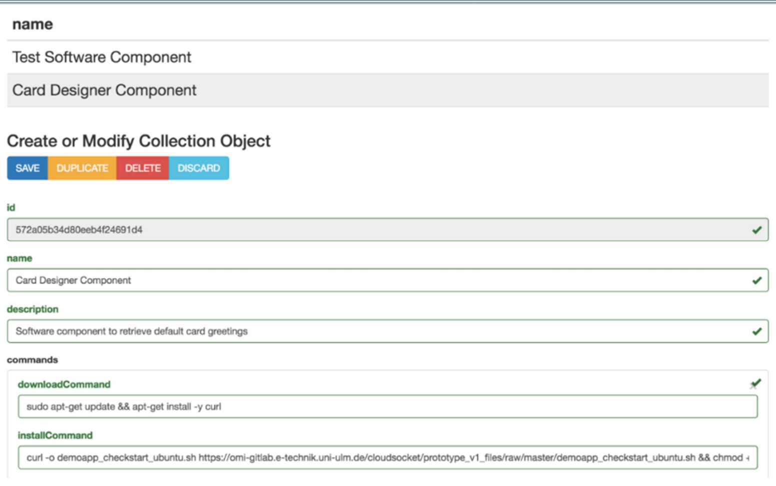
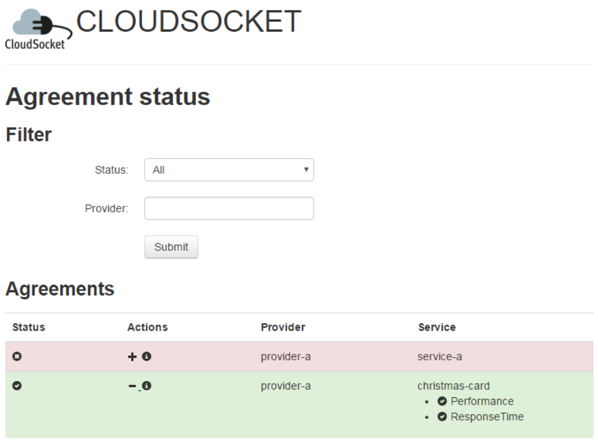
BPaaS Allocation Environment	
The goal of the BPaaS Allocation Environment is to configure allocation directives and rules for an executable workflow model to be deployed and executed in the cloud. An executable workflow model, as produced by the BPaaS Design Environment, does not contain information in terms of which concrete services can be exploited to realise the functionality of the workflow tasks. The respective selection of services per workflow task is supported by the BPaaS Allocation Environment. Similarly, driven by the same set of requirements, the same environment can also be used to address the selection of IaaS offerings to support the deployment and provisioning of (internal) BPaaS software components. Apart from these basic allocation decisions, the BPaaS Allocation Environment covers the specification of adaptation rules that drive the adaptation behaviour of a BPaaS as well as the specification of SLAs and marketing meta-data (e.g., pricing) for a certain BPaaS. In the end, the resulting product is a BPaaS bundle that can be published in the Marketplace.	
Component	**Description**
Allocation Tool	It is responsible for selecting a BPaaS Design Package (previously created via the Design Environment) and creating a BPaaS Bundle ready to be published in the Marketplace and deployed in the Execution Environment.  BPaaS Allocation Environment CloudSocket Welcome broker Logout Allocation Environment My bundles My brands Brands Bundles Save Save & Close Save & New Discard Design package Marketplace metadata Deployment Replace Delete https://hs21.fhoster.com/cloudsocket/Allocation/Engine.jsp?locale=en#!homePage Name: Easter Greeting Design Package Status: DRAFT Missing contents: - Name: Easter Greeting Design Package Description Business process bpmn: business-process.bpmn (54568 Bytes) Business process img: business-process.png (38872 Bytes) Workflow bmpn: workflow.bpmn (70709 Bytes) Workflow img: workflow.png (169764 Bytes) Last logged in: 29-Jan-2016 12:34
Access	SaaS Deployment: https://hs21.fhoster.com/cloudsocket/Allocation_prototype/Engine.jsp (user credentials on demand)
License	Proprietary
Further Details	Demonstration: Section 4.2.4 Technical Details: Section 4.3.3
Manual	https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Allocation+Environment+Components
Lead Partner	FHOSTER

Table 2 - Prototype Components of Allocation Environment

BPaaS Marketplace and Execution Environment	
The BPaaS Execution Environment deploys and executes a BPaaS bundle, once this has been purchased by a customer at the BPaaS Marketplace. The BPaaS deployment proceeds according to the deployment plan included in the bundle, along with additional configuration activities taken to enable the proper deployment of the workflow into a workflow engine and of the monitoring infrastructure. Once a BPaaS is successfully deployed, it can be run and managed by the BPaaS Customer. In addition, it is automatically monitored in a cross-layer manner and adapted, when needed, in order to keep up with the SLOs promised in the enclosed SLA of the BPaaS bundle.	
BPaaS Marketplace	
Component	**Description**
yCONNECT	It is an online frontstore through which customers discover, analyse and purchase BPaaS bundles by also initialising the respective BPaaS deployment in the cloud environment. Therefore, it is responsible for linking the Allocation to the Execution Environment, giving the client the opportunity to buy and configure the BPaaS bundles received from the Allocation and to send the configured bundles to the Execution for provisioning.  MY ACCOUNT LOGOUT CREATE ACCOUNT English SHOP 0 ITEMS Home / Processes Tags Add to cart Processes Showing 5 processes CHRISTMAS CARD GREETINGS PROCESS (SMALL VIRTUAL MACHINE) V1.2 Mathema s.r.l. Mathema Business Processes Customizable design for Christmas greetings with own photo and text Possibility to create and send greetings to entire clients and commercial partners easy and quickly €10.00 (one time) CHRISTMAS CARD GREETINGS PROCESS (MEDIUM VIRTUAL MACHINE) V1.2 Mathema s.r.l. Mathema Business Processes Customizable design for Christmas greetings with own photo and text Possibility to create and send greetings to entire clients and commercial partners easy and quickly €20.00 (one time) SENDING CHRISTMAS GREETINGS WITH MAIL INCL. METRICS bwcon Baden-Württemberg: Connected e.V. Customizable design for Christmas greetings with own photo and text Possibility to create and send greetings to entire clients and commercial partners easy and quickly This bundle €30.00 (one time) CHRISTMAS GREETING TEST WITH NEW FILE FROM THE DESIGN V2.0
Access	SaaS Deployment: http://csmarket.ymens.com:8080/ (user credentials on demand)
License	Proprietary
Further Details	Demonstration: 3.2.2 and 4.2.5 Technical Details: Section 3.3.1
Manual	https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Marketplace+Component
Lead Partner	YMENS
Repository Manager	It is responsible for managing the information related to different entities such external services, software components, and cloud providers. It is a transversal component allowing the population, browsing and search of this information using standard web technologies.

	 name Test Software Component Card Designer Component Create or Modify Collection Object SAVE DUPLICATE DELETE DISCARD id 572a05b34d80eeb4f24691d4 name Card Designer Component description Software component to retrieve default card greetings commands downloadCommand sudo apt-get update && apt-get install -y curl installCommand curl -o demoapp_checkstart_ubuntu.sh https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/prototype_v1_files/raw/master/demoapp_checkstart_ubuntu.sh && chmod +
Access	SaaS Deployment: http://134.60.64.221/ (user credentials on demand) Download: as docker images • mongodb: https://hub.docker.com/_/mongo/ • restheart: https://hub.docker.com/r/softinstigate/restheart/ Restheart SchemaForm UI: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/restheart-schemaform-ui Registry Client Library: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/registry-client.
License	GNU AGPL v3.0 [3]
Further Details	Demonstration: Section 4.2.5 Technical Details: Section 3.3.1
Manual	https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Repository+Manager+Component
Lead Partner	FHOSTER, ATOS
BPaaS Execution Environment	
Component	**Description**
Workflow Engine	It is responsible for managing the deployment, execution and management of the different workflow instances of a purchased BPaaS workflow at the execution phase. 134.60.64.132/activiti-webapp-explorer2/ Aplicaciones SecurityDashboard SPARQL sfa DocumentacionTecn Activity RSpec Semantica BlogTecnologia YourBPM GWT maven Viajes Jaxb Otros marcadores CloudSocket ID de usuario Contraseña Iniciar sesión
Access	http://134.60.64.132/activiti-webapp-explorer2/ (deployed as part of a bundle and user

	credentials on demand) Download Engine: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/workflow-engine. Download Workflow Parser: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/workflow-parser.
License	Apache License Version 2.0 [7]
Further Details	Demonstration: Section 3.2.3 Technical Details: Section 3.3.2.1
Manual	https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Workflow+Engine+Component
Lead Partner	ATOS
Cloud Provider Engine	It is responsible for the complete deployment and lifecycle management of all the required components of the BPaaS bundle, including software components and VMs across multiple clouds, with transactional semantics (at least for the deployment part).
Access	Colosseum: http://134.60.64.155:9000 Entrypoint: http://134.60.64.155:9012/job Download: https://github.com/cloudiator/ Download EntryPoint Wrapper: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/execution-environment-simple-entrypoint
License	Apache License Version 2.0 [7]
Further Details	Demonstration: Section 3.2.3 Technical Details: Section 3.3.2.3
Manual	https://github.com/cloudiator https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Execution+Environment+Entrypoint
Lead Partner	UULM
Monitoring Engine	It is responsible to monitor a BPaaS and correlate/aggregate monitoring data from different levels, from atomic services or cloud components up to the level of workflows.
Access	http://134.60.64.155:8080 Download: https://github.com/cloudiator/visor.git
License	Apache License Version 2.0 [7]
Further Details	Demonstration: Section 3.2.3 Technical Details: Section 3.3.2.2
Manual	https://github.com/cloudiator/visor

Lead Partner	UULM, FORTH
Adaptation Engine	It is responsible for reconfiguring the BPaaS possibly across different levels (via e.g., service substitution, workflow recomposition, horizontal and vertical scaling) to resolve the problematic situations identified by triggered adaptation rules.
Access	http://134.60.64.155:9000/api/composedMonitor http://134.60.64.155:9000/api/componentHorizontalOutScalingAction http://134.60.64.155:9000/api/componentHorizontalInScalingAction Download: https://github.com/cloudiator/axe-aggregator
License	Apache License Version 2.0 [7]
Further Details	Demonstration: Section 3.2.3 Technical Details: Section 3.3.2.4
Manual	https://github.com/cloudiator/visor
Lead Partner	UULM, FORTH
SLA Engine	The SLA Engine represents the component responsible for generating, storing and observing the formal documents describing electronic service level agreements between the parties involved in a BPaaS offering (CloudSocket broker, cloud service providers supporting the BPaaS functionality), including of course the BPaaS customer  CLOUDSOCKET **Agreement status** **Filter** Status: All Provider: Submit **Agreements** Status | Actions | Provider | Service ✗ | + ℹ | provider-a | service-a ✓ | − ℹ | provider-a | christmas-card: ✓ Performance, ✓ ResponseTime
Access	SaaS Deployment for SLA Dashboard: http://134.60.64.232:8000 (user credentials on demand) Core SLA Engine: http://134.60.64.232:8080 Download: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/sla-framework
License	Apache License Version 2.0 [7]
Further Details	Demonstration: Section 3.2.3 Technical Details: Section 3.3.2.5

Manual	https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/SLA+Engine+Component
Lead Partner	ATOS

Table 3 - Prototype Components of Marketplace and Execution Environment

BPaaS Evaluation Environment	
The BPaaS Evaluation Environment has the overall goal to evaluate a BPaaS in order to provide optimization suggestions to its designer. This evaluation comes in various forms: (a) the assessment of KPIs, (b) the derivation of best deployments for the BPaaS, (c) the production of adaptation event patterns and rules and (d) the discovery of bottlenecks and problematic business model parts. Thus, the externally seen functionality of the BPaaS Evaluation Environment maps to initiating the performance of analysis tasks as well as the retrieval and graphical presentation of the various evaluation/analysis results produced according to suitable graphic metaphors by a business dashboard.	
Component	**Description**
Semantic Repository	A semantic repository enabling performing different types of analysis.
Access	http://134.60.64.222:8080/rest-test-swagger-0.0.1-SNAPSHOT/ Download: https://github.com/openlink/virtuoso-opensource.
License	GPL v2 [27]
Further Details	Demonstration: Section 4.2.2 Technical Details: Section 4.3.1.3
Manual	http://docs.openlinksw.com/virtuoso/.
Lead Partner	FORTH
Conceptual Analytics Engine	Provides an API through which KPI assessment can be performed on top of the semantic repository
Access	http://134.60.64.222:8080/rest-test-swagger-0.0.1-SNAPSHOT/ Download: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/evaluation_skb/repository/archive.zip?ref=master.
License	Mozilla Public Licence (MPL) 2.0 [28]
Further Details	Demonstration: Section 4.2.2 Technical Details: Section 4.3.1.2
Manual	https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Conceptual+Analytics+Engine
Lead Partner	FORTH

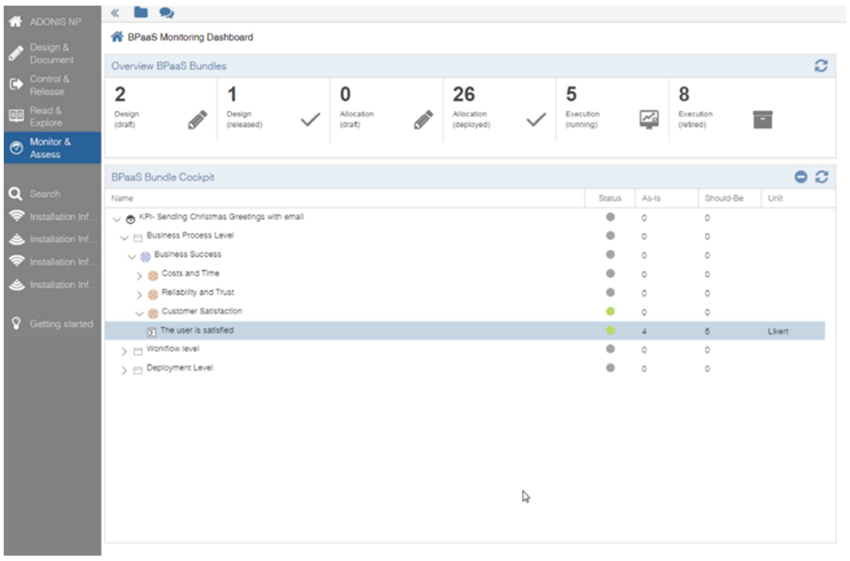
Hybrid Business Dashboard	Enables the visualisation of the analysis information via the use of suitable metaphors. Guides the user in properly performing the different types of analysis 
Access	SaaS Deployment: https://www.cloudsocket.eu/ADONISNP36/ (user credentials on demand)
License	Closed source
Further Details	Demonstration: Section 4.2.2 Technical Details: Section 4.3.1.1
Manual	N/A
Lead Partner	BOC

Table 4 - Components of Evaluation Environment

BPaaS Monitoring Dashboard

Overview BPaaS Bundles

2	1	0	26	5	8
Design (draft)	Design (released)	Allocation (draft)	Allocation (deployed)	Execution (running)	Execution (retired)

BPaaS Bundle Cockpit

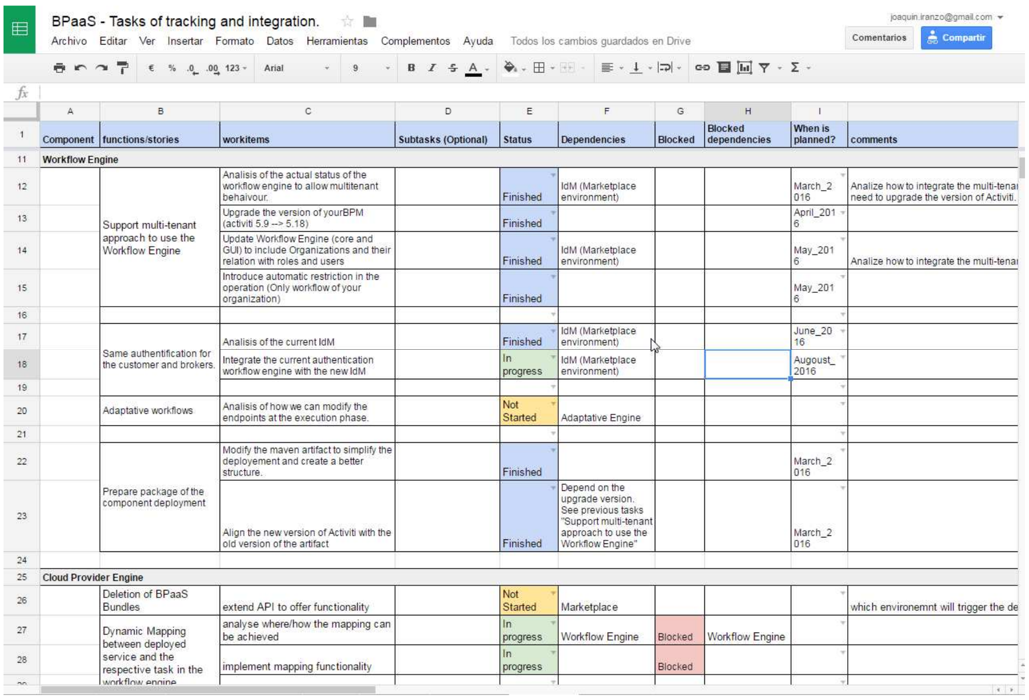
Name	Status	As-Is	Should-Be	Unit
KPI- Sending Christmas Greetings with email	●	0	0	
Business Process Level	●	0	0	
Business Success	●	0	0	
Costs and Time	●	0	0	
Reliability and Trust	●	0	0	
Customer Satisfaction	🟢	0	0	
The user is satisfied	🟢	4	5	Likert
Workflow level	●	0	0	
Deployment Level	●	0	0	

2.1 Implementation Approach

The different environments are aligned with the respective tasks in WP4: i) T4.2 is responsible for the Design and Evaluation Environment, ii) T4.4 for the Allocation Environment, and iii) T4.3 for the Marketplace and Execution Environment. Despite the fact that these environments are independent, the project has started to integrate and align them from the very beginning of the development phase. This allows to harmonize the different interfaces and interactions, which have been clearly defined in D4.1 [1], allowing to have a complete lifecycle including all the BPaaS (management) phases.

This approach has been incremental by following the four following steps: i) detailing the interfaces and identifying the basic functionalities for the different environments and components; ii) passing internal testing for internal components or between different (internal) components of the same environment; iii) inter-integration between environment pairs; iv) final integration of all components and covering all phases and environments.

This integration has been followed up through short periods; identifying the risk and problems as soon as possible in order to take corrective actions, if needed. A shared excel file has been introduced to review the details of the work items and the respective functionality realisation status and their planning, besides the dependencies and blocked actions. Every environment has been covered by different sheets where the components report the status of their work items. Morover, this file also defines the different sprints and their integration at the different levels. WP4 has realized periodic meetings every two weeks to analyse the integration, the work items per component and to identify the new parallel discussions, such as the definition of a registry-based approach, the CAMEL integration/extension, and the metric definition.



BPaaS - Tasks of tracking and integration.

	A	B	C	D	E	F	G	H	I	
1	Component	functions/stories	workitems	Subtasks (Optional)	Status	Dependencies	Blocked	Blocked dependencies	When is planned?	comments
11	Workflow Engine									
12		Support multi-tenant approach to use the Workflow Engine	Analisis of the actual status of the workflow engine to allow multitenant behaivour.		Finished	IdM (Marketplace environment)			March_2016	Analize how to integrate the multi-tenai need to upgrade the version of Activiti.
13			Upgrade the version of yourBPM (activiti 5.9 --> 5.18)		Finished				April_2016	
14			Update Workflow Engine (core and GUI) to include Organizations and their relation with roles and users		Finished	IdM (Marketplace environment)			May_2016	Analize how to integrate the multi-tenai
15			Introduce automatic restriction in the operation (Only workflow of your organization)		Finished				May_2016	
16										
17		Same authentification for the customer and brokers.	Analisis of the current IdM		Finished	IdM (Marketplace environment)			June_2016	
18			Integrate the current authentication workflow engine with the new IdM		In progress	IdM (Marketplace environment)			Augoust_2016	
19										
20		Adaptative workflows	Analisis of how we can modify the endpoints at the execution phase.		Not Started	Adaptative Engine				
21										
22		Prepare package of the component deployment	Modify the maven artifact to simplify the deployement and create a better structure.		Finished				March_2016	
23			Align the new version of Activiti with the old version of the artifact		Finished	Depend on the upgrade version. See previous tasks "Support multi-tenant approach to use the Workflow Engine"			March_2016	
24										
25	Cloud Provider Engine									
26		Deletion of BPaaS Bundles	extend API to offer functionality		Not Started	Marketplace				which environemnt will trigger the de
27		Dynamic Mapping between deployed service and the respective task in the workflow engine	analyse where/how the mapping can be achieved		In progress	Workflow Engine	Blocked	Workflow Engine		
28			implement mapping functionality		In progress		Blocked			

Figure 2 - Work item definition table

This simple approach allows to integrate the different environments at their early states and it is completely aligned with the Task T4.5 CloudSocket Integration and Consolidation, which will introduce more efficient tools for managing the continuous integration for all the levels; spanning not only the development, for example a ticketing system to cover the development, but also integration, bugs, and management of different environments (test, integration and production).

This document describes and summarizes the first prototype that covers all the phases of a BPaaS Bundle defined in D4.1, through respective environments and components. So, the first prototype comprises the:

- description of the components [19],
- access to the software. If it is open-source, we link directly to the Source Code [20]. If it is proprietary, then it is provided the access of the component as a SaaS,
- available instances for the prototype as SaaS, deployed onto the UULM and YMENS infrastructure,
- summary documentation for the components drawn from their complete specification in the component wiki [18].

The project team decided to introduce a simple business process which includes all the basic concepts allowing to harmonize and integrate the environments accordingly. This process concerns the sending of the Christmas greeting cards automatically via email. This process is shortly analysed in the following subsection.

2.1.1 BPaaS Sample

The project has used the business process for sending Christmas cards to illustrate the idea of a BPaaS. Although it is obvious that such a simple business process is embedded in a Customer Relationship Management software that may be provided as a SaaS solution, we use this well-known example to introduce the vision of BPaaS and enable the integration of the different BPaaS Environments.

First, the business process flow describes the activities of Christmas card distribution, such as using pre-selected images or creating own images, entering the text, uploading the recipient email-list or sending the email. Those actions, when executed in the cloud, may cause particular issues, which have been identified by respective business requirements, related to image copyrights, legal compliance of text and of storing private data – such as email addresses – and, finally, the IT resources allocation, in case all emails are to be sent at the same time.

Such issues or requirements can drive the conduction of Cloud-specific extensions on the business processes that are necessary to configure the technical behaviour of: (a) the workflow – e.g. by introducing service tasks that check the copyright of an image vs. manual acceptance of terms - and (b) the IT-infrastructure – e.g. by data processing of private data within Europe.

The domain specific business processes are transferred into executable workflows by considering current IT-cloud offerings and enabling either automatic alignment between the business and IT level or a manual one via the interaction of the respective roles (business process and workflow designers). For example, in manual alignment, the integrator can gather more information in order to identify potential cloud services to cover the defined features and the business analyst can support the integrators in finding the best option together, in order to avoid entering unnecessary loops, or worse, having misunderstandings that might lead to an incorrect definition of the business process.

Once an executable workflow is generated, it needs to be allocated accordingly. Two different types of allocations actually take place: i) each service task in the workflow is mapped to a respective atomic service able to realise its functionality; ii) for each internal service mapping to a workflow service task, we need to discover the best possible infrastructure (IaaS or PaaS) services able to support its deployment and provisioning. Apart from this basic allocation decisions, additional information needs to be captured in order to produce a respective BPaaS bundle. This information includes: i) Service Level Agreements (SLA) that specify the service level to be delivered by the BPaaS along with penalties to be applied if this level is deteriorated; ii) adaptation rules that drive the BPaaS runtime behaviour in order to still keep up with the service level promised or violate it in the least possible way; iii) marketing data covering categories, tasks, and pricing for the BPaaS bundle.

Such a BPaaS is published and offered in a marketplace similar to a SaaS marketplace. After this BPaaS is purchased, it is automatically deployed in a multi-cloud infrastructure and its operation is monitored with the main goal to react as quickly as possible to respective problematic situations. Semantic abstraction, conceptual analytics and human interaction enable the abstraction back from BPaaS logs to high-level business information visualised in business dashboards, indicating that all data has been stored in Europe, no data violation took place, and cloud-bursting had been performed within the limits of additional IT costs. This dashboard enables a learning cycle as well as to optimise the BPaaS such that its performance is improved and possibilities for increased broker gains are examined.

2.2 Architecture

The CloudSocket prototype architecture was defined in D4.1 [1], which created a technical and detailed common understanding, hence enabling a coordinated initial implementation of the BPaaS environments that compose the so-called CloudSocket. Additionally, it specified the functional capabilities, involved roles, competencies and data interchange formats of the five BPaaS environments in order to enable the combination or exchange of those environments to create personalized adaptations with the BPaaS environments developed in this project or by exchanging environments with alternative implementations.

Figure 3 introduces the four major building blocks as well as their relationships and the data exchanges. Each of the four building blocks supports one phase of the BPMS paradigm when applied for business process management in the cloud. The BPaaS offerings are provided to the customer via the Marketplace, whereas the BPaaS Execution Environment enables their operation in the cloud. This is the main reason that these latter two environments look consolidated in the architectural image. The conceptual challenge of bridging domain specific business processes to executable workflows that are in production in the cloud and are constantly improved via evaluation information, is performed by the other BPaaS Environments.

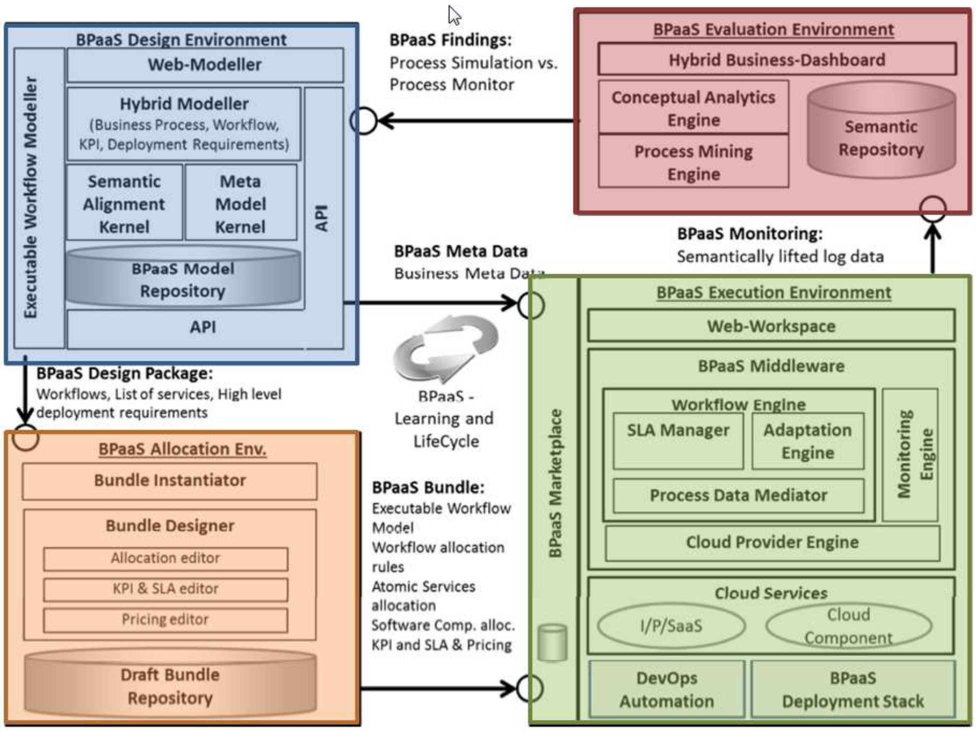


Figure 3 - CloudSocket Architecture

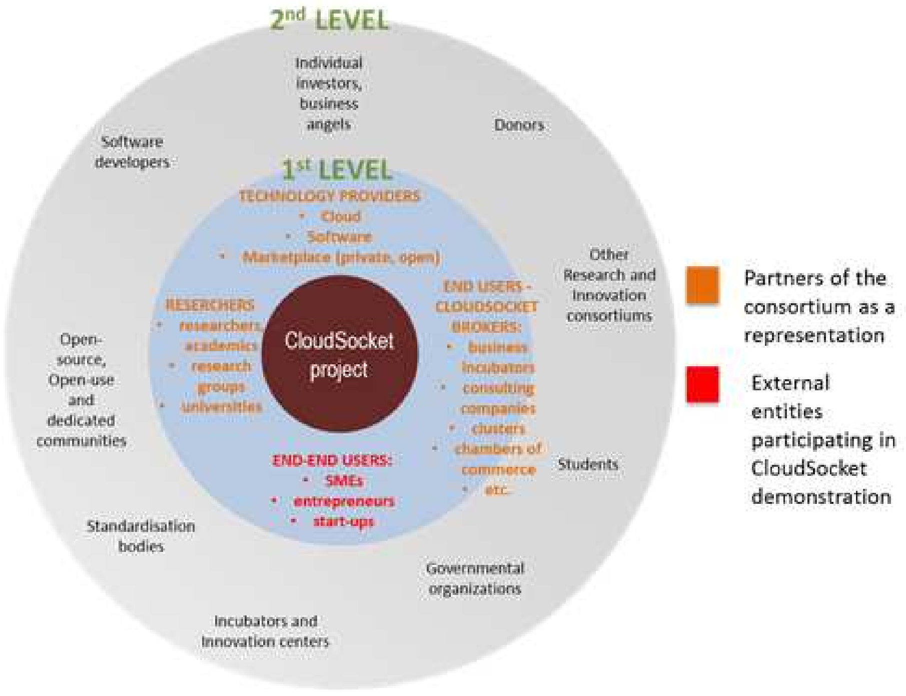
The BPaaS Environments that support the BPaaS lifecyle are: i) the BPaaS Design Environment enabling to model business processes, business requirements (KPIS), business decisions, workflows and semantic annotations, ii) the BPaaS Allocation Environment able to link deployable workflows to concrete services, iii) the BPaaS Execution Environment enabling to deploy, execute, monitor and adapt the BPaaS workflow, iv) the BPaaS Evaluation Environment which lifts monitoring information and logs to key performance indicators at the business level as well as provides additional types of BPaaS analysis, and v) the BPaaS Marketplace being the main place where customers can browse, search, select, and purchase BPaaS bundles as well as receive billing reports for the BPaaS bundles that they have bought. More details can be found in D4.1 [1].

2.3 Actors and Perspectives

The project has considered a classification of the stakeholders, which are all the individuals, groups, units or communities which (a) could be interested in project development and exploitation or (b) just follow the project results and especially those activities of CloudSocket that could have a direct or indirect impact on them, as described in the internal deliverable (D8.1- First Explotation and business plan). These stakeholders have been split into 2 main levels: i) the different groups of project partners which indeed are involved in RDI work and CloudSocket concept development as well as are directly linked to project success and further exploitation, ii) all further groups of stakeholders which are not directly in charge of the project execution but are somehow interested in its results. Therefore, the first prototype is aligned with this first level, where the following stakeholders are considered (see Figure 4):

- End-End User can be Small and Medium Enterprises (SMEs), founders and start-ups, both IT and not IT-based, which in case of CloudSocket project represent potential broker customers. These end-end users are identified in the sequel of this deliverable as BPaaS Customers as they represent potential purchasers of the BPaaS offerings provided by the CloudSocket Brokers.
- End-Users of the platform also known as CloudSocket Brokers. Such brokers do not operate only as a third-party business that is an intermediary between the purchaser of a cloud computing service (SMEs and start-ups) and the provider of that service (Marketplace with its multi-clouds offer) but can also act as a consultant to support SMEs in transforming their business processes into the cloud, after assessing their readiness to move to the cloud. Moreover, such brokers can realize the whole lifecycle of cloud-based business processes thus saving for SMEs/startups the work of attempting to perform the business-to-IT alignment themselves as well as the investment of resources and time in order to support this lifecycle.
- Technology providers are all the project partners, which provide their software components/products such as base CloudSocket functionality realization or add-on functionalities or other (commercial) organisations which offer replacements of software components that have been developed by the members of the CloudSocket consortium. We foresee that such technology providers might also offer the whole CloudSocket prototype to brokers in order to enable the respective management of the BPaaS to be generated and offered to BPaaS Customers. In some cases, such providers may also offer particular environments, like the Marketplace, to be exploited by brokers, leading to a more loosely coupled instantiation of a running CloudSocket platform comprising different environments that are maintained by different operators.
- Researchers are university representatives (researchers, academics) and research groups (universities, institutes) focusing on research and development of future results and new sciences, elaborating their teaching courses, building network and research communities, consolidating and conceptualizing of abstracts or general ideas. This stakeholder kind performs research and development tasks which can

result in add-on or component replacement prototypes that could be embraced in existing CloudSocket product variants.



2nd LEVEL
Individual investors, business angels
Donors
Software developers
Other Research and Innovation consortiums
Open-source, Open-use and dedicated communities
Students
Standardisation bodies
Governmental organizations
Incubators and Innovation centers
1st LEVEL
TECHNOLOGY PROVIDERS
• Cloud
• Software
• Marketplace (private, open)
RESERCHERS
• researchers, academics
• research groups
• universities
END USERS - CLOUDSOCKET BROKERS:
• business incubators
• consulting companies
• clusters
• chambers of commerce
• etc.
END-END USERS:
• SMEs
• entrepreneurs
• start-ups
CloudSocket project
Partners of the consortium as a representation
External entities participating in CloudSocket demonstration

Figure 4 - CloudSocket ecosystem

The first prototype focuses at this first level of stakeholders, and the demonstration covers the two main actors, the BPaaS Customers and the CloudSocket Brokers; integrating all the necessary environments and components to cover the complete BPaaS lifecycle:

- BPaaS Customers map to the End-end users in the stakeholders' classification. They are different organizations, which are interested to purchase and use the published BPaaS bundles offered by the different brokers.
- CloudSocket Brokers are the End Users of the platform in the stakeholders' classification. They are directly the brokers that identify a business process and create the respective BPaaS bundles of this process to be published in their marketplace in the demonstration.

Therefore, the two main actors and their perspectives have been considered to show the demonstration in the following sections:

- BPaaS Customer Perspective: Involves entities such as SMEs, founders or start-ups that want to reduce their costs and create added value for their business processes by moving some parts of them (or fully) in the cloud. They can search, review and purchase the different BPaaS bundles, which have been published previously by the CloudSocket Brokers. Afterwards, the purchased BPaaS Bundle is deployed automatically in the cloud and it is ready to be used by the customer.

- CloudSocket Broker Perspective: The CloudSocket Brokers want to create a BPaaS bundle in order to expose them to possible customers that might be interested in them. They need to interact with the Design and Allocation Environments to create the bundle. Afterwards, they can publish the bundles in the marketplace such that these bundles are exposed and available to their customers. Finally, the brokers can analyze the results of the Evaluation Environment to optimise existing BPaaS or create new ones by identifying the respective needs to be covered.

These two perspectives are explicated in the next two sections by relying on respective demonstrations of the CloudSocket prototype.

3 PERSPECTIVE: BPAAS CUSTOMER

3.1 Introduction

This demonstration showcases [30] how the BPaaS Customer interacts with the CloudSocket BPaaS Marketplace to search, review and buy a fitting BPaaS Bundle. Once the respective bundle is purchased, the Execution Environment deploys all necessary BPaaS components in the cloud and, upon success, it informs the BPaaS customer that the BPaaS is ready to be used.

An explanation is given over all the steps followed by the Execution Environment in order to deploy and make available the BPaaS purchased.

The demonstration is concluded with the BPaaS customer accessing the Workflow Engine in order to create and execute instances of the BPaaS workflow deployed.

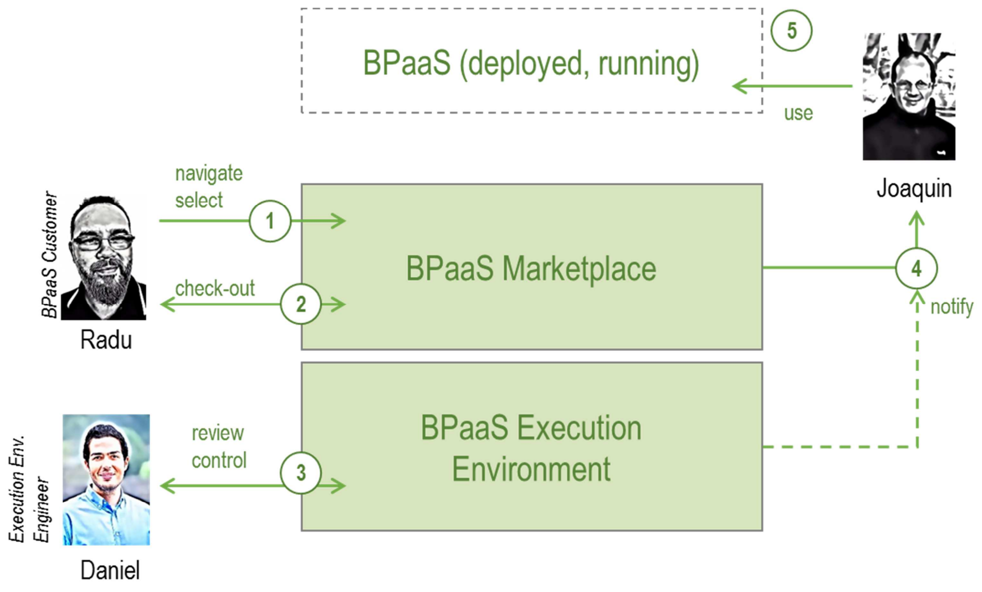


Figure 5 - BPaaS Customer perspective

3.2 Demonstration

3.2.1 Involved roles

In this demonstration the involved roles, which have been defined in the CloudSocket architecture (D4.1 [1]), and/or map to the CloudSocket roles in the common understanding wiki [22], are:

- BPaaS Customer who has interest to analyse, purchase and use the different available BPaaS bundles. There are defined kinds of associated roles depending on the respective skills and responsibilities required in the interaction of the customer with the CloudSocket platform.

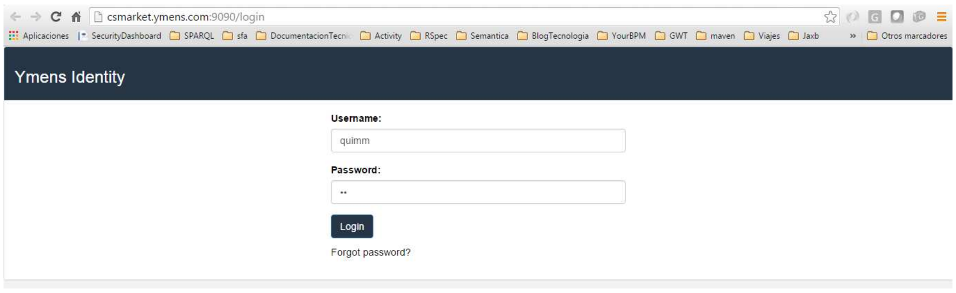
  - Business Engineer with business skills is responsible for finding and purchasing the different BPaaS bundles in the Marketplace.
  - Process Responsible with technical skills is responsible for dealing with the Execution Environment and managing the deployed BPaaS workflows (e.g., instance creation, execution, pausing, resuming and cancellation).
  - Knowledge Worker is responsible for managing the interaction with the manual workflow tasks of BPaaS bundle purchased.
- CloudSocket Operator is responsible for hosting the CloudSocket platform instance also guaranteeing its continuous operation. In addition, it also ensures a smooth communication between the different BPaaS environments. We foresee that this role could be split into 5 sub-roles mapping to the respective BPaaS environments that need to be operated. This can map to internal roles of the organisation undertaking the operation of a CloudSocket platform instance. It can also map to the case that the platform instance is collaborately operated by different operators responsible for the proper functioning of each BPaaS environment needed. In this sense, the BPaaS Execution Environment (smooth) operation could be the responsibility of an Execution Environment operator.

In the demonstration, there are different actors assigned to different roles:

- Radu acting as a user of the BPaaS customer on the marketplace undertaking the role of a Business Engineer.
- Daniel acting as the Execution Environment operator (within or independently from the CloudSocket Operator role)
- Joaquin acting as a user of the BPaaS customer responsible for running the BPaaS in the cloud (undertaking the roles of the Process Responsible to instantiate the purchased BPaaS workflow and the Knowledge Worker to deal with the different user/manual tasks involved in this workflow).

3.2.2 Purchase the Christmas Greetings Cards

Radu acting as a business engineer on behalf of his company, wants to distribute to all customers a Christmas card this year. As a founder, he is keen to have good customer relations to his early adopters and network; at the same time he is very busy to organize this activity. At an event, he learns about CloudSocket and the BPaaS idea. Interested in the topic, he logs into the CloudSocket Marketplace (Figure 6), whose URL was given by the respective broker in the event (Figure 7).

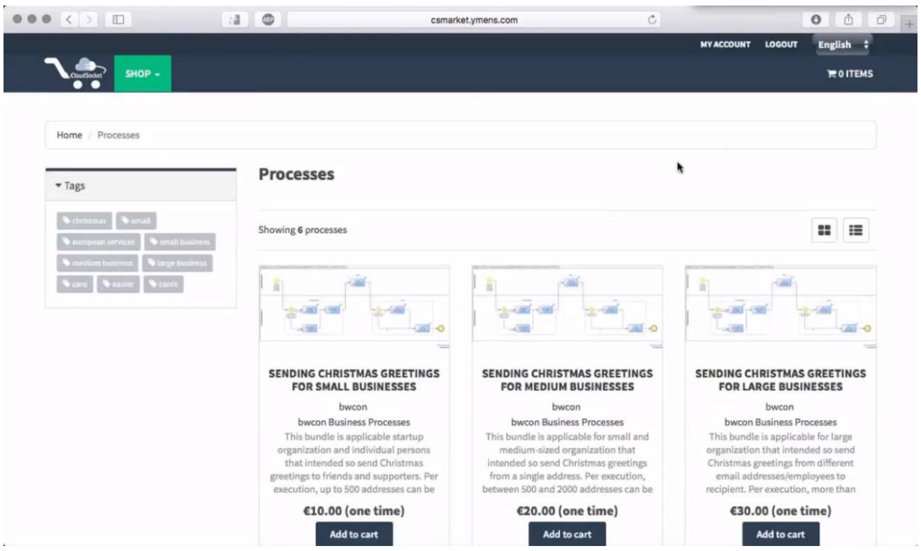

csmarket.ymens.com:9090/login

Ymens Identity

Username: quimm

Password: ••

Figure 6 – Login Marketplace



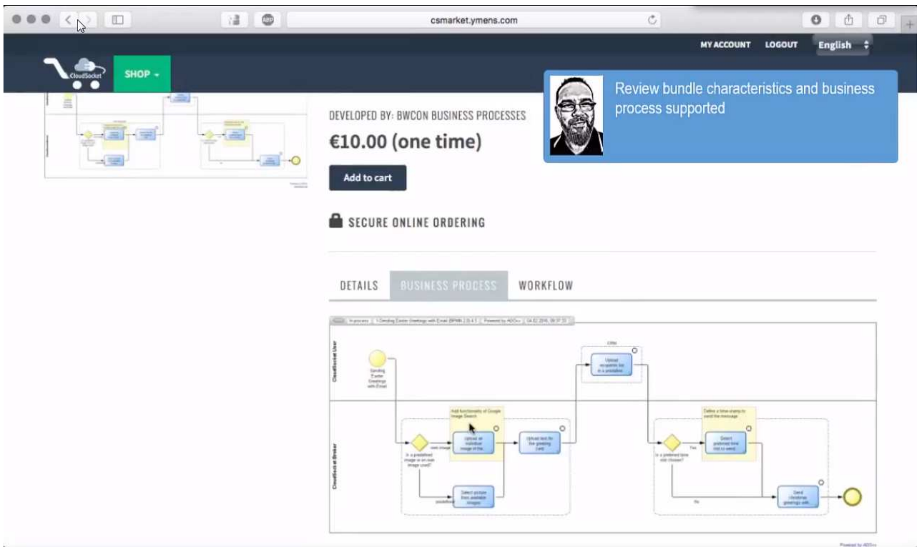
Processes

Showing 6 processes

SENDING CHRISTMAS GREETINGS FOR SMALL BUSINESSES	SENDING CHRISTMAS GREETINGS FOR MEDIUM BUSINESSES	SENDING CHRISTMAS GREETINGS FOR LARGE BUSINESSES
bwcon	bwcon	bwcon
bwcon Business Processes	bwcon Business Processes	bwcon Business Processes
This bundle is applicable startup organization and individual persons that intended so send Christmas greetings to friends and supporters. Per execution, up to 500 addresses can be	This bundle is applicable for small and medium-sized organization that intended so send Christmas greetings from a single address. Per execution, between 500 and 2000 addresses can be	This bundle is applicable for large organization that intended so send Christmas greetings from different email addresses/employees to recipient. Per execution, more than
€10.00 (one time)	€20.00 (one time)	€30.00 (one time)

Figure 7 – Browsing the offer of the broker.

After being successfully logged in (Figure 6), Radu searches for appropriate BPaaS bundles that are being offered by the broker and identifies a fitting one. As he wants to save money but still have anyone in his network messaged, he decides to go for the medium sized BPaaS bundle from those realising the required business process, which matches the required number of the recipients of the greetings to send as well as has a suitable and reasonable price. Radu has the chance to view related details of the selected BPaaS bundle in both a textual and graphical manner (Figure 8). The respective marketplace information is provided as descriptions, tags, bundle prices and (bundle) categories.



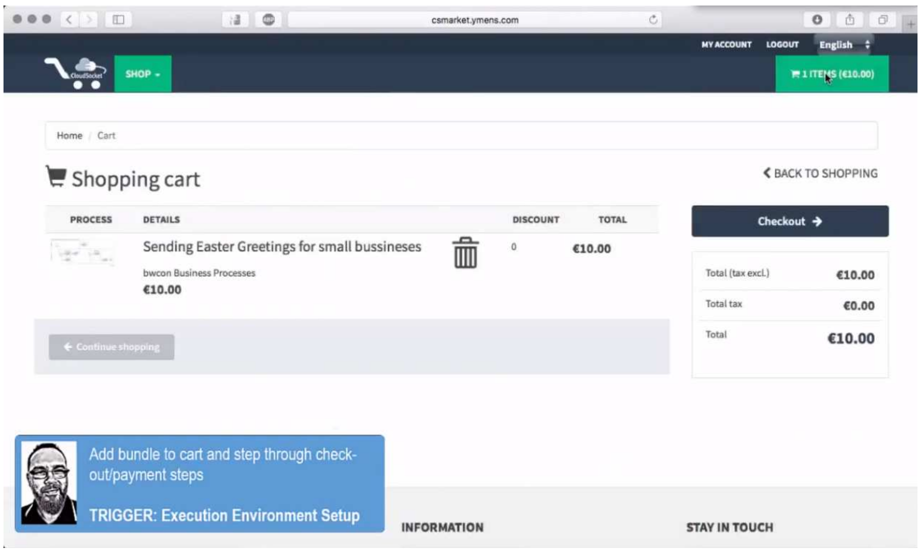
DEVELOPED BY: BWCON BUSINESS PROCESSES

€10.00 (one time)

Review bundle characteristics and business process supported

Figure 8 – View BPaaS bundle details.

Radu performs a checkout to his cart and buys the bundle (Figure 9). A notification window provides information on the deployment run that has been triggered, attempting to deploy the BPaaS bundle (along with all related assets) in the cloud (Figure 10). Radu will be notified by email as soon as the deployment has been completed.



Shopping cart

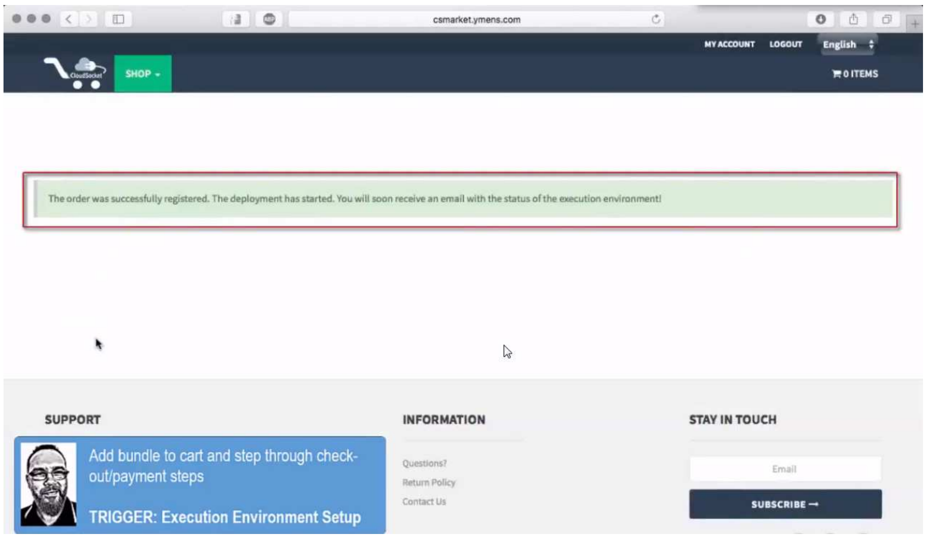
PROCESS	DETAILS	DISCOUNT	TOTAL
	Sending Easter Greetings for small bussineses bwcon Business Processes €10.00	0	€10.00

Total (tax excl.)	€10.00
Total tax	€0.00
Total	€10.00

Add bundle to cart and step through check-out/payment steps

TRIGGER: Execution Environment Setup

Figure 9 – Marketplace cart



The order was successfully registered. The deployment has started. You will soon receive an email with the status of the execution environment!

Add bundle to cart and step through check-out/payment steps

TRIGGER: Execution Environment Setup

Figure 10 – Notification message for the purchased BPaaS bundle

The following video shows how Radu buys a Christmas Greetings BPaaS bundle to satisfy his requirements: https://youtu.be/G9Qqi0p6yWg

3.2.3 Deep-dive into the infrastructure

The deployment has been triggered by the interaction of Radu with the marketplace. Parallelly, Daniel checks, as acting an Execution Environment Engineer Operator, the new deployments from the UI of the Cloud Provider Engine and intervenes if something wrong takes place.

The deployment of the bundle is performed transparently for the BPaaS Customer organization; nevertheless we take a deep dive into the Execution Environment to understand the different steps of the setup happening on the backend.

The BPaaS Execution Environment will deploy the Software components over an infrastructure automatically (in our case OpenStack) and also setup related services (e.g., monitoring) in the infrastructure. Figure 11 gives a simplified overview of all technical steps executed during the BPaaS deployment, while the following two paragraphs shortly analyse the main functionality of these steps.

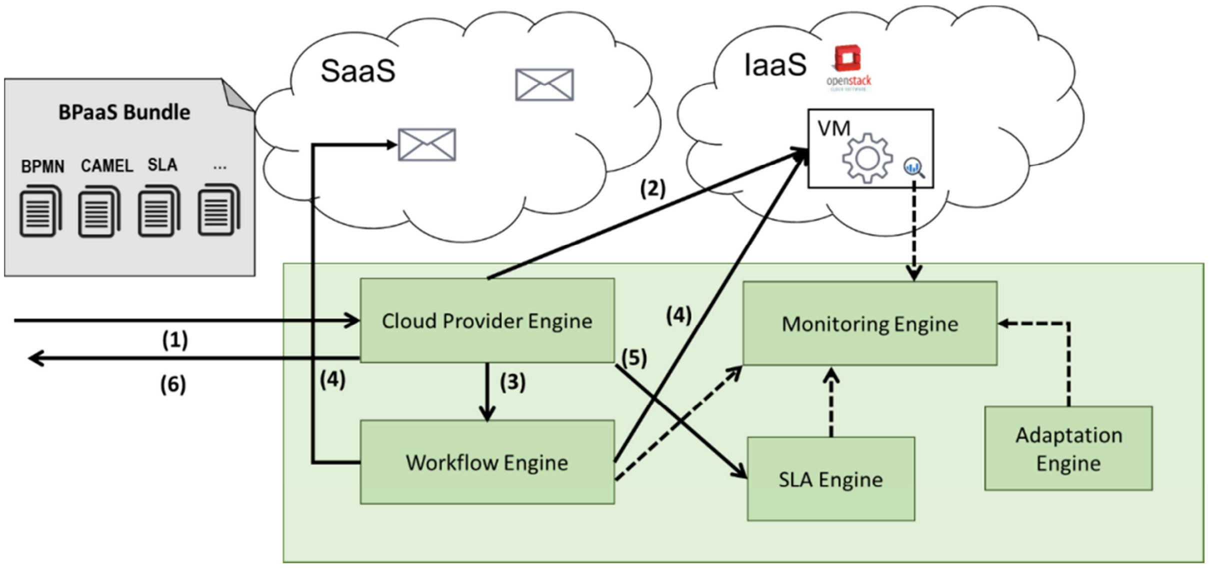
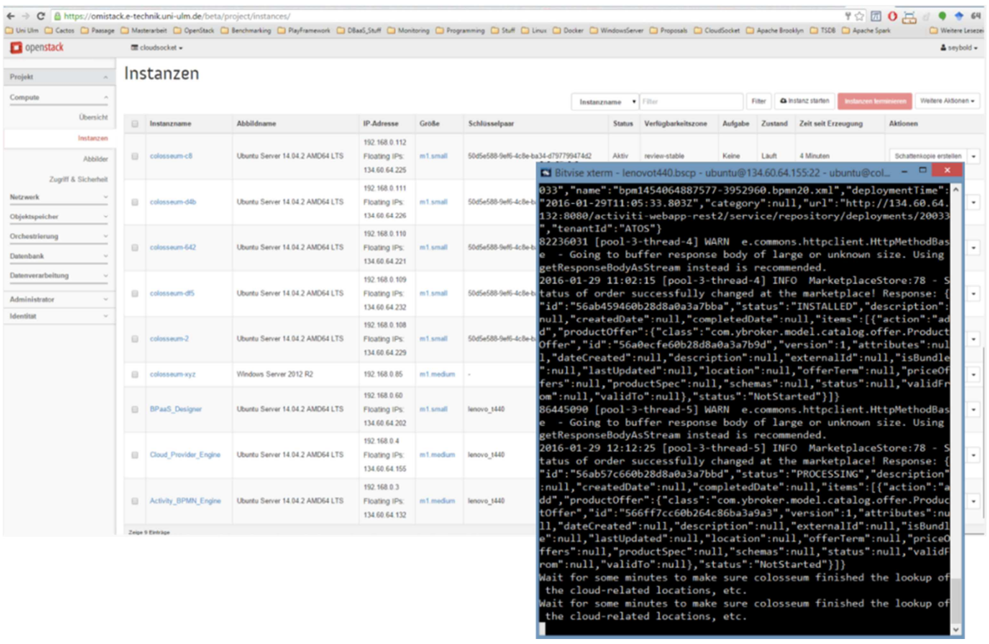


Figure 11 - BPaaS Deployment

(1) The cloud provider engine receives the BPaaS bundle from the Marketplace, when Radu has finished the order process. The bundle mainly contains the BPMN file, the CAMEL file and the SLA template along with other information. The Cloud Provider Engine processes the CAMEL file. *(2)* Based on the CAMEL description, the Cloud Provider Engine will access the specified cloud provider and provision the required virtual machines, install the services, configure the monitoring and receive the actual endpoints for the (internal) service instances. Figure 12 shows a sample screenshot mapping this deployment process to the selected UULM cloud infrastructure, which has been selected for the demonstration. The OpenStack dashboard in Figure 12 shows the VMs that were created by the Cloud Provider Engine.

As soon as all required services are successfully setup by the Cloud Provider Engine, (3) the BPMN file with the actual service endpoints is passed to the Workflow Engine in order to update the BPMN workflow file and deploy it. (4) Now, the Workflow Engine can manage the deployed workflows, which include these actual endpoints services; and create its instances, which are able to access to all the specified services for the deployed BPaaS Bundle during their execution.In step (5) the SLA template is passed to the SLA Engine in order to configure the BPaaS Bundle specific SLAs (6) Finally, after all deployment actions for the BPaaS bundle have been deemed successful, Radu gets notified that the bundle was successfully deployed and receives the required details to execute the workflow via the Workflow Engine (see Section 3.2.4).

With the completed BPaaS Bundle deployment, the Monitoring Engine provides the BPaaS monitoring data to the SLA Engine and the Adaptation Engine. Based on this monitoring data the SLA Engine evaluates the bundle specific SLAs and the Adaption Engine is able to trigger the adaptation actions that are specified in CAMEL.



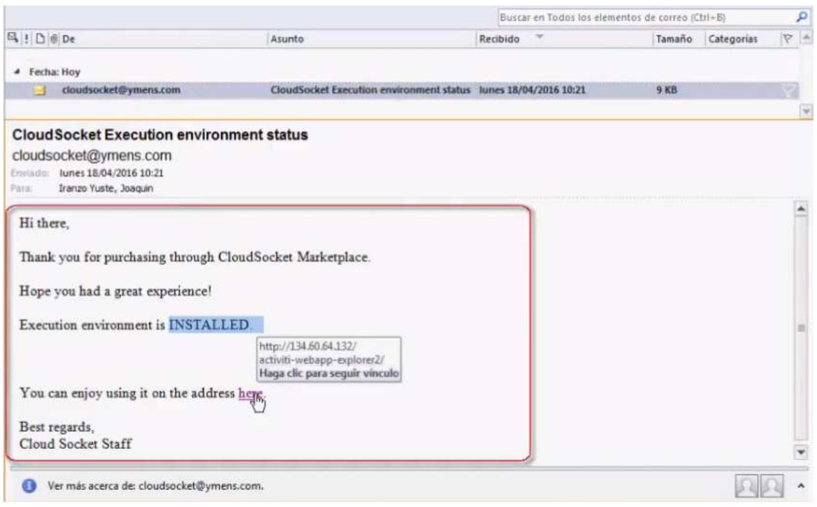
```
033","name":"bpm1454064887577-3952960.bpmn20.xml","deploymentTime":
"2016-01-29T11:05:33.803Z","category":null,"url":"http://134.60.64.
132:8080/activiti-webapp-rest2/service/repository/deployments/20033
","tenantId":"ATOS"}
82236031 [pool-3-thread-4] WARN  e.commons.httpclient.HttpMethodBas
e  - Going to buffer response body of large or unknown size. Using
getResponseBodyAsStream instead is recommended.
2016-01-29 11:02:15 [pool-3-thread-4] INFO  MarketplaceStore:78 - S
tatus of order successfully changed at the marketplace! Response: {
"id":"56ab459460b28d8a0a3a7bba","status":"INSTALLED","description":
null,"createdDate":null,"completedDate":null,"items":[{"action":"ad
d","productOffer":{"class":"com.ybroker.model.catalog.offer.Product
Offer","id":"56a0ecfe60b28d8a0a3a7b9d","version":1,"attributes":nul
l,"dateCreated":null,"description":null,"externalId":null,"isBundle
":null,"lastUpdated":null,"location":null,"offerTerm":null,"priceOf
fers":null,"productSpec":null,"schemas":null,"status":null,"validFr
om":null,"validTo":null},"status":"NotStarted"}]}
86445090 [pool-3-thread-5] WARN  e.commons.httpclient.HttpMethodBas
e  - Going to buffer response body of large or unknown size. Using
getResponseBodyAsStream instead is recommended.
2016-01-29 12:12:25 [pool-3-thread-5] INFO  MarketplaceStore:78 - S
tatus of order successfully changed at the marketplace! Response: {
"id":"56ab57c660b28d8a0a3a7bbd","status":"PROCESSING","description"
:null,"createdDate":null,"completedDate":null,"items":[{"action":"a
dd","productOffer":{"class":"com.ybroker.model.catalog.offer.Produc
tOffer","id":"566ff7cc60b264c86ba3a9a3","version":1,"attributes":nu
ll,"dateCreated":null,"description":null,"externalId":null,"isBundl
e":null,"lastUpdated":null,"location":null,"offerTerm":null,"priceO
ffers":null,"productSpec":null,"schemas":null,"status":null,"validF
rom":null,"validTo":null},"status":"NotStarted"}]}
Wait for some minutes to make sure colosseum finished the lookup of
 the cloud-related locations, etc.
Wait for some minutes to make sure colosseum finished the lookup of
 the cloud-related locations, etc.
```

Figure 12 - OpenStack dashboard and console VM

Now we can have a detailed look at the Execution Environment to see what has been triggered after Radu purchase of the BPaaS bundle in the following video https://youtu.be/pTH1mTSI5qg

3.2.4 Preparation of the Christmas Cards

Radu forwards the confirmation email to Joaquin in order to take care of the respective workflow execution; both belong to the same organization. Then, Joaquin receives it with the confirmation and the link to execute the BPaaS bundle (see Figure 13). Acting as a process responsible, he logs in the Workflow Engine and creates a new instance of the workflow deployed for the BPaaS bundle that has been purchased previously (see Figure 14).



De	Asunto	Recibido	Tamaño	Categorías
cloudsocket@ymens.com	CloudSocket Execution environment status	lunes 18/04/2016 10:21	9 KB	

CloudSocket Execution environment status
cloudsocket@ymens.com
Enviado: lunes 18/04/2016 10:21
Para: Iranzo Yuste, Joaquin

Hi there,

Thank you for purchasing through CloudSocket Marketplace.

Hope you had a great experience!

Execution environment is INSTALLED.

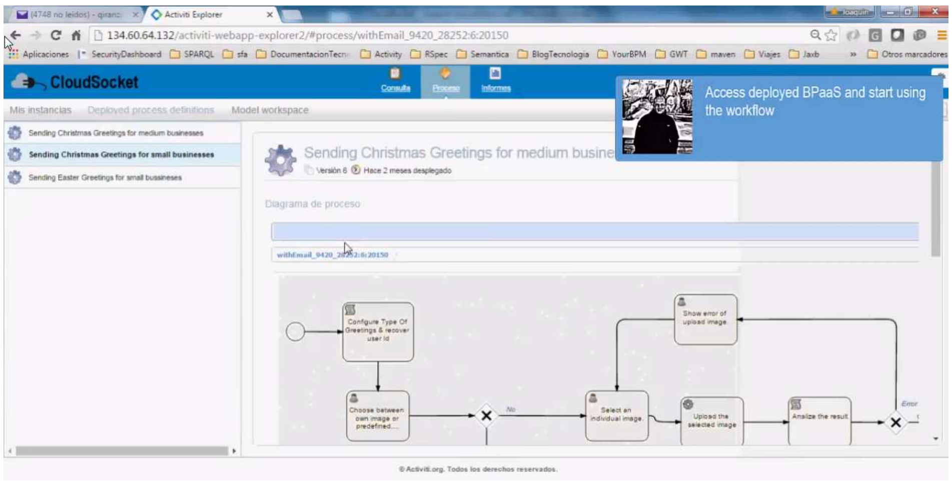
You can enjoy using it on the address here.

Best regards,
Cloud Socket Staff

http://134.60.64.132/activiti-webapp-explorer2/
Haga clic para seguir vínculo

Ver más acerca de: cloudsocket@ymens.com.

Figure 13 – Email for the notification of purchased BPaaS bundle.

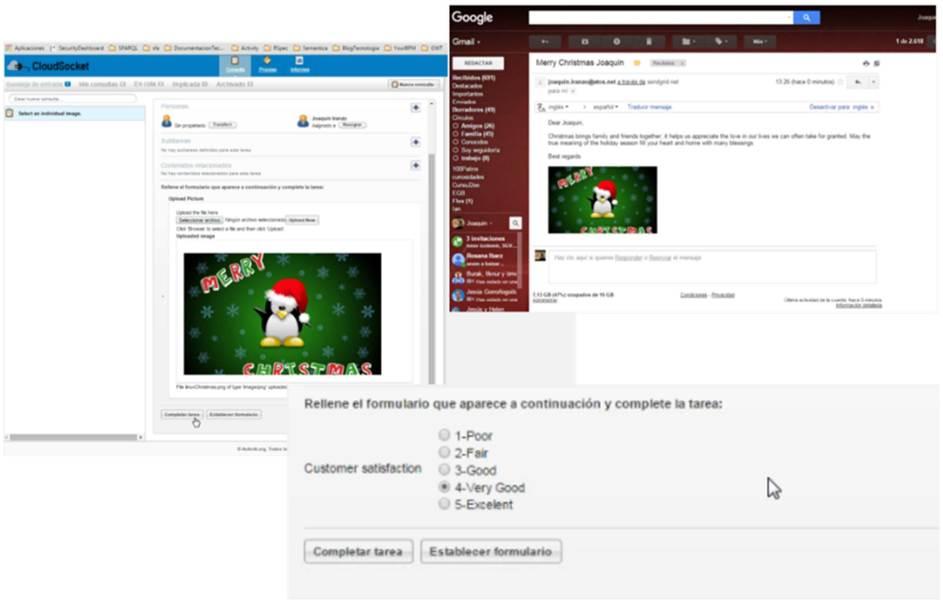

Access deployed BPaaS and start using the workflow

CloudSocket
Consulta
Proceso
Informes
Mis instancias
Deployed process definitions
Model workspace
Sending Christmas Greetings for medium businesses
Sending Christmas Greetings for small businesses
Sending Easter Greetings for small bussineses
Sending Christmas Greetings for medium businesses
Versión 6
Hace 2 meses desplegado
Diagrama de proceso
Configure Type Of Greetings & recover user Id
Choose between own image or predefined...
No
Select an individual image
Show error of upload image
Upload the selected image
Analize the result
Error
© Activiti.org. Todos los derechos reservados.

Figure 14 – Creation of the instance for the purchased BPaaS bundle

The workflow instance is started and Joaquin, acting as the Knowledge Worker role, receives different user tasks, which are fulfilled by specifying the necessary inputs (image, text, recipient list) (see Figure 15). The workflow executes and automatically sends the custom Christmas cards to the specified recipients.

Joaquin is satisfied that he was able to run this process without additional effort, thus fulfilling the main goal of this organization, and concludes the execution by providing (customer satisfaction) feedback (see Figure 15).


Rellene el formulario que aparece a continuación y complete la tarea:

Customer satisfaction
- ○ 1-Poor
- ○ 2-Fair
- ○ 3-Good
- ● 4-Very Good
- ○ 5-Excelent

Completar tarea | Establecer formulario

Figure 15 - Workflow Engine GUI and received email

How Joaquin interacts with the Workflow Engine to prepare the Christmas card and send it to his customers is showed in the following video https://youtu.be/wTJdHFSlMwU

3.3 Environments

The involved environments are detailed and outlined by the red board in Figure 16:

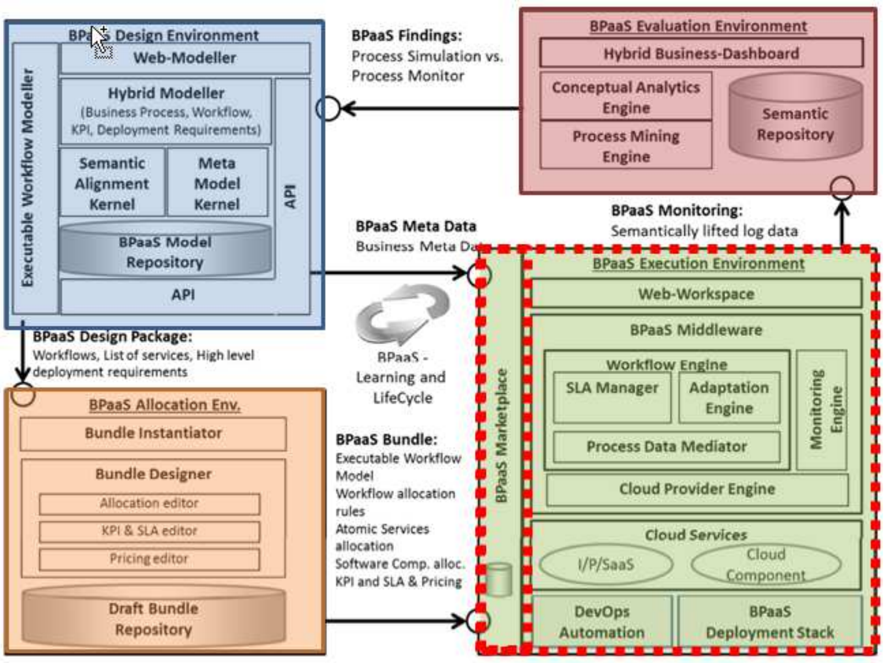


Figure 16 - CloudSocket architecture with the environments involved in the BPaaS customer perspective outlined with a red rectangle

The following sections, organised according to the environment that they concern, analyse in more detail the components that belong to the involved environments.

3.3.1 BPaaS Marketplace

The Marketplace's role is to link the BPaaS Allocation to the Execution Environment, giving the client the opportunity to buy and configure the BPaaS bundles received from the Allocation and to send the configured bundles to the Execution Environment for provisioning.

This BPaaS Marketplace provides the following high-level features: i) BPaaS & SaaS Product Catalogue; ii) Decision Support System for BPaaS procurement; iii) Customer & User Registration & On-boarding; iv) Identity Provisioning & Identity Lifecycle Management; v) Cloud Service Provider registration of atomic cloud services; (vi) Registry Services; vii) Authorization (at service level) & Authentication (at user level).

In order to support the aforementioned functionalities, the BPaaS Marketplace comprises two main components. These components are the following:

- Marketplace (yCONNECT*)* allows the customers to discover, analyse and purchase a BPaaS bundle in the cloud environment. Therefore, this is the actual Marketplace which enables the customers to browse, analyze and buy the BPaaS bundles.
- Repository Manager is responsible for managing the information related to different entities, such external services, software components, cloud providers and so on. It is a transversal component, with

respect to the rest of the Environments, allowing the population, browsing and searching of this information using standard web technologies.

Figure 17 shows the detailed scheme of the CloudSocket Reference Architecture and the location of the different components.

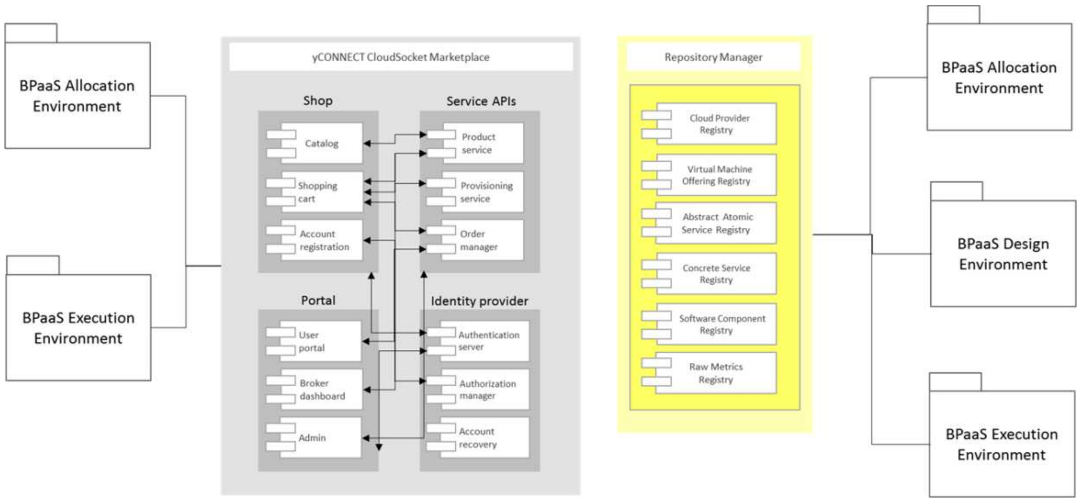


Figure 17 - Internal architecture for the BPaaS Marketplace

The BPaaS Marketplace components in each level interact with each other in order to properly deliver and visualise the functionalities of the environment. More information about how these interactions are performed and what are the respective scenarios covered can be found at the following URL: https://www.cloudsocket.eu/uml/5-Marketplace/remotedocu/modeldocu/27012016151357/modelContentHTML/ in which the corresponding UML diagrams [2] can be viewed. This information was also covered in the D4.1 deliverable [1].

3.3.1.1 Marketplace

For BPaaS Customers, as SME or other organizations, the CloudSocket Marketplace offers a place for registration. Registration process is ensured by the Account Registration component. Only registered users are allowed to buy BPaaS bundles.

Through the Catalog component, BPaaS Customers browse and view details of published bundles. By using the Shopping Cart component, the BPaaS Customer can purchase the bundles.

The Marketplace will expose the following interfaces:

- A public graphical user interface of a BPaaS Shop.
- A graphical user interface for authentication and authorization through IdM.
- A graphical user interface for registered users which provides checkout capabilities for BPaaS as well as the non-shop related capabilities mapping to the Portal

The main functionalities are:

- Register users and companies through Registration component.
- List available BPaaSs through Catalog component.
- Checkout BPaaSs through Shopping cart, Order manager and Provisioning service components.
- Manage various user related information through User portal component.

The following table indicates the details of the component.

Type of ownership	Extension
Original tool	GRADY (Grails Rapid Application Development for Ymens)
Planned OS License	Proprietary. Component available only as service
Reference community	YMENS R&D staff
Lead Partner	YMENS

Table 5 - Details of the Marketplace component

Comprises

- Shop – main module that manages shopping experience
- Portal – module that manages non-shop related interactions of the users (purchase history, account details, access provisioned items)
- Service API – provides data for Web UI and core workflow functionality (Execution Environment – provision endpoint)
- Identity provider – ensures M2M and U2M authentication and authorization flows

Depends on

- Allocation Environment – to provide BPaaSs to be listed in the Marketplace
- Execution Environment – to provision in cloud BPaaSs purchased by the user

Architecture design

The general architecture of this component can be viewed in grey, in Figure 17.

The Figure 18 introduces the deployment SOA model of the Marketplace.

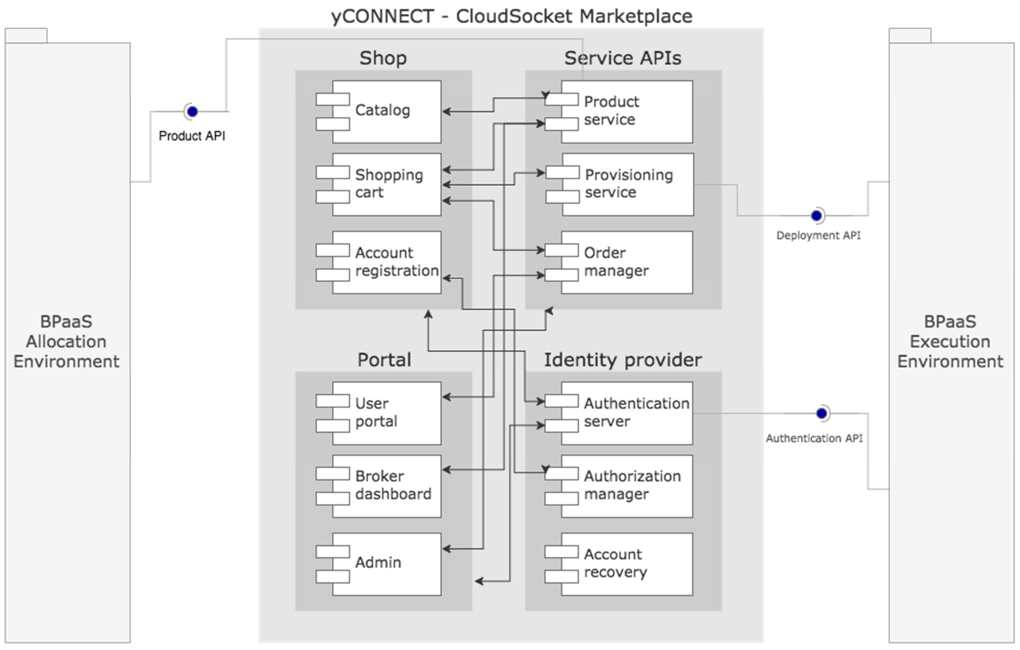


Figure 18 - Component diagram of the Marketplace

Deploy Diagram

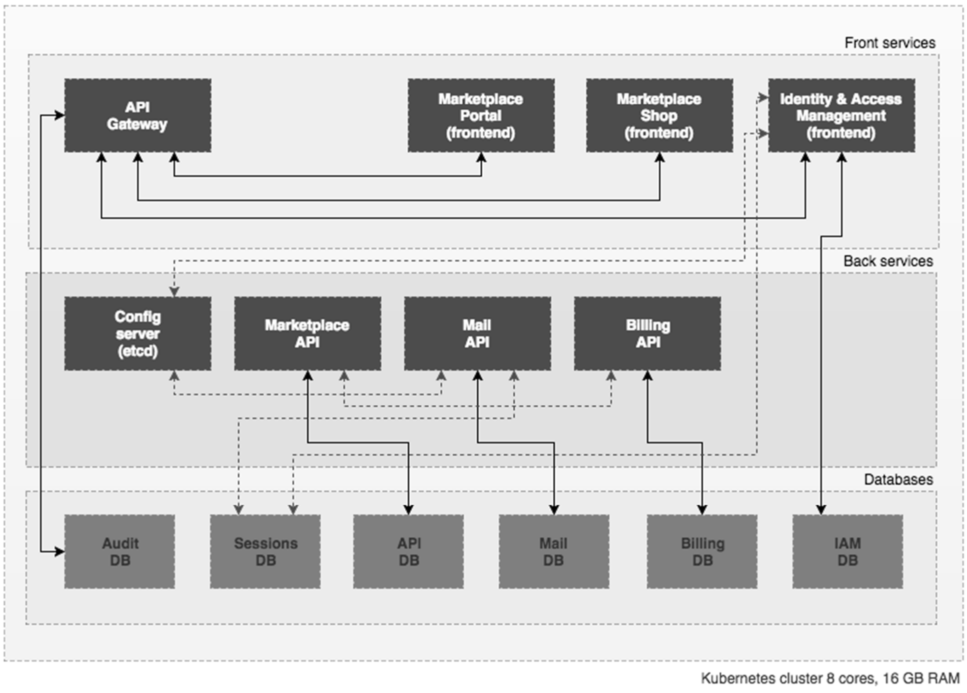


Figure 19 - Deploy diagram of the Marketplace

Functionalities

Table 6 indicates the covered functionalities and their status:

Functionality	Description	Completed for release	Integrated (which level)
Shop	The module provides BPaaS listings into the shop environment and allows registered users to checkout items	Yes (1st release)	Complete lifecycle.
Service APIs	The module ensures API connectivity of the Marketplace in the CloudSocket value stream (Product API – used by Allocation Environment; Provisioning API and Order Management – used by Execution Environment)	Yes (1st release)	Complete lifecycle
Portal	The module enables users access to purchased BPaaSs as well as shopping history visualisation and account information management	Partially (1st release)	
Identity provider	This module provides authentication and authorization services for the Marketplace as well as for the entire CloudSocket value chain	Yes (1st release)	Complete M2M with Allocation Environment Complete M2M with Execution Environment

			In progress U2M with Execution Environment

Table 6 - Functionalities of the Marketplace

Table 6 highlights these functionalities while details about the workitems and the respective functionality realisation status can be found at the follow-up excel file, which has been referenced in Section 2.1. Details about the interactions required to realise them can be found at D4.1 [1] and especially the MP UC2 - Publish BPaaS Bundle to Product Catalogue, MP-UC2-Purchase BPaaS Bundle, MP-UC3-Register New Customer User and MP-UC-4-Onboard Cloud Service Provider along with respective UML diagrams [2]

Manuals

The manual, API description, and handbooks are detailed in the alive wiki documentation:
https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Marketplace+Component

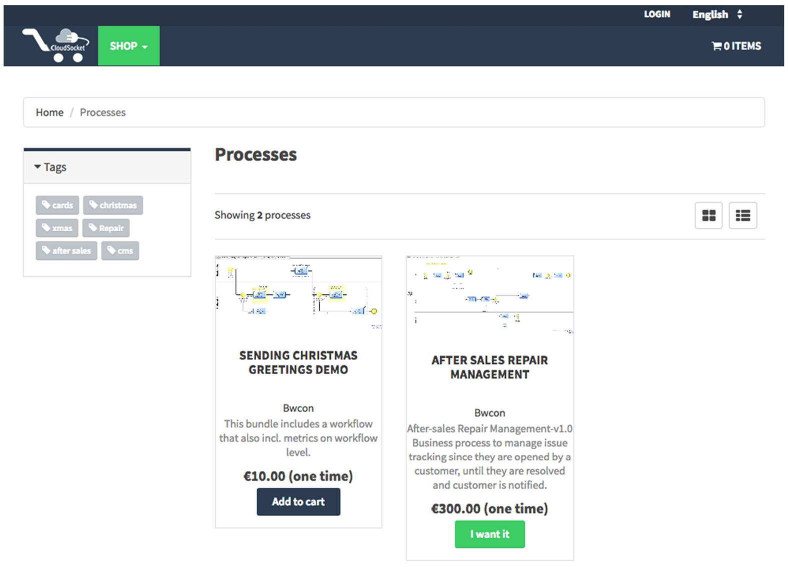
Download

yCONNECT CloudSocket Marketplace is built on top of GRADY (Grails Rapid Application Development for Ymens) platform that is a collection of open tools for proprietary development.

Grails, the open source framework and the baseplate for GRADY is available at: https://grails.org/download.html

Instances

The yCONNECT CloudSocket Marketplace is available at: http://csmarket.ymens.com:8080/ .



Processes

Showing 2 processes

SENDING CHRISTMAS GREETINGS DEMO

Bwcon

This bundle includes a workflow that also incl. metrics on workflow level.

€10.00 (one time)

AFTER SALES REPAIR MANAGEMENT

Bwcon

After-sales Repair Management-v1.0 Business process to manage issue tracking since they are opened by a customer, until they are resolved and customer is notified.

€300.00 (one time)

Figure 20 - Marketplace web page.

3.3.1.2 Repository Manager

Summary

Some components of the CloudSocket Environments need to access a set of information related to different entities, such as external services, software components, and cloud providers. For this reason, the CloudSocket project needs a specific transversal component allowing the population, browsing and searching of this information using standard web technologies.

The Repository Manager has the responsibility to satisfy those functionalities and has a key role into the CloudSocket architecture because it's exploited by all the CloudSocket Environments to access this information, which for now, there is only one structural level of registries is offered. For the next release, the consortium might consider leveraging upon this structural level in order to create a semantic one on top of it so as to be also aligned with the research results in WP3. This will allow the posing of more advanced and semantic queries which can enhance the different scenarios/tasks that can be performed by the respective exploiting components as well as enable better query response accuracy levels.

The repository manager offers the information needed by the CloudSocket Environments by the means of a number of registries. The main registries identified for the CloudSocket context are the followings:

- Cloud Provider Registry: it contains the information about a cloud provider like the name, the type of offering that it provides (i.e. Platform as a Service or Infrastructure as a Service) and technical details, specified using the CAMEL language, which enable the Cloud Provider Engine to have the exact knowledge of how to access and exploit the (cloud) services of this Cloud Provider.
- Virtual Machine Offering Registry: it contains the information about a Virtual Machine offering provided by a specific Cloud Provider. Such information includes the number of cores, the size of the main memory and disk storage as well as the price of the given offering.
- Abstract Service Registry: it contains the information about abstract services mapping to an abstract functionality; these services are of course not real in the sense that they are not developed, deployed and executed and thus available in a certain endpoint. Such information includes the abstract service name, short textual description, and interface. It also includes the specification of semantic annotations oriented towards semantically explicating the exact service functionality as well as the respective input and output that is provided by each method of this service. Such information can enable a semantic discovery of services which has been proven to reach higher discovery accuracy levels.
- Concrete Service Registry: it contains the information about concrete atomic services already up and running. Such information includes the name, the description, the interface definition, the interface protocol (i.e. SOAP or REST) and a list of instances for this service which are associated to a pair of location and endpoint URL (thus enabling us to know exactly where each instance of a certain atomic service is available).
- Software Component Registry: it contains the information about software components developed. Such information includes the name, the description, the set of commands to install the software component to specific operating system images, the communication port(s) exposed, the minimum hardware requirements for the proper functioning of this component and the dependencies it has with other components.
- Metrics Registry: it contains the information about the metrics handled by the CloudSocket platform. Such information includes the name of the metric, its short description, the property that is being measured, the definition of the sensor which captures the measurements for a raw metric, the derivation formula for composite metrics, the unit and the type of the metric as well as default measurement schedule and window information.

The Table 7 shows which registries information is accessed by which CloudSocket environment:

	Design Environment	**Allocation Environment**	**Execution Environment**	**Evaluation Environment**
Cloud Provider Registry	--	X	--	--
Virtual Machine Offering Registry	--	X	--	X
Abstract Atomic Service Registry	X	X	X	--
Concrete Service Registry	X	X	X	X
Software Component Registry	X	X	X	X
Raw Metrics Registry	X	X	--	X

Table 7 - Registries information accessed by Environments

The following table indicates the details of the component.

Type of ownership	New development
Original tool	MongoDB[4] and Restheart [5]
Planned OS License	GNU AGPL v3.0. [3]
Reference community	MongoDB, Restheart
Lead Partner	FHOSTER, ATOS

Table 8 - Details of the Repository Manager

Comprises

- Data Layer
- Functional Layer
- User Interface Layer

Architecture design

The general architecture of this component can be viewed in yellow in Figure 17, while Figure 21 shows the Repository Manager's internal architecture.

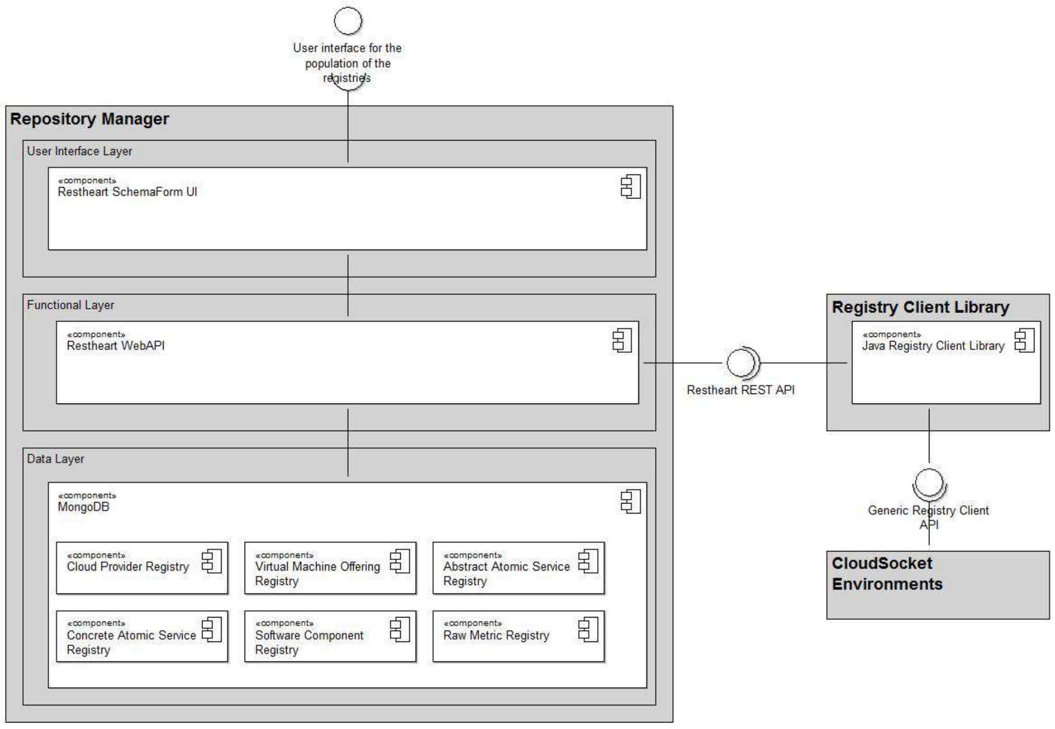

User interface for the population of the registries
Repository Manager
User Interface Layer
«component» Restheart SchemaForm UI
Functional Layer
«component» Restheart WebAPI
Data Layer
«component» MongoDB
«component» Cloud Provider Registry
«component» Virtual Machine Offering Registry
«component» Abstract Atomic Service Registry
«component» Concrete Atomic Service Registry
«component» Software Component Registry
«component» Raw Metric Registry
Restheart REST API
Registry Client Library
«component» Java Registry Client Library
Generic Registry Client API
CloudSocket Environments

Figure 21 – Internal architecture of the Repository Manager

The Repository Manager follows a classic three-tier architecture pattern in which the user interfaces (presentation), functional process logic, computer data storage and data access are developed and maintained as independent modules. The three layers are described in detail below:

- In the Data Layer, there exists a sole component mapping to the MongoDB NoSQL database: The latter is a free and open-source cross-platform document-oriented database. It is classified as a NoSQL database. MongoDB [4] avoids the traditional table-based relational database structure in favour of JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster. It is used to implement the persistency layer to store the registry JSON documents. Each JSON document of the registry describes a different resource. The resources can refer to each other. For example, we are able to describe a Virtual Machine offering as a JSON document and associate it using an object id reference to a specific Cloud Provider.

- Functional Layer: There lies the Restheart WebAPI which is a Java open source Web API server built on top of the MongoDB database [4]. RESTHeart [5] exposes a RESTful application programming interface with CRUD operations, following the Hypertext Application Language (HAL) standard. RESTHeart naturally fits an architecture where there is the need to invoke document-oriented data services on top of MongoDB [4] via HTTP. The main features of the framework are the followings:

    - Lightweight Server: the API is ready to use and does not require any coding;
    - Built on standards like HTTP, JSON, RESTful, HAL, json-schema;
    - Pluggable Authentication and Authorization with ready to use Identity Managers and role based Access Manager. This feature will be used in order to integrate the Repository Manager with the Identity Manager provided by the Marketplace;
    - Data operations API enabling document management and including the following operations: create, read, update, delete and query documents;
    - Data validation with json-schema.

- User Interface Layer: There lies, the Restheart SchemaForm UI which is a custom AngularJS application, developed by FHOSTER for the CloudSocket project, based on two main angular-js modules:
    - angular-restheart: it is the client provided by RESTHeart project to facilitate the access to the webapi from an angular-js client;
    - angular-schemaform: it is a set of AngularJS directives (and a couple of services) to generate Bootstrap 3 ready forms from a JSON Schema. The main features of this angular-js module are:
        - Validates the form using a JSON Schema;
        - Fine tunes presentation with a form definition, changes field types, changes order and so on;
        - Lots of basic form types out of the box;
        - Supports array with drag'n'drop or in tabs;
        - Easily extended with custom form field types.

  The web application has all the features useful to access the content of the registries and modify the content of each object. It provides mechanisms for validation of the data and a user-friendly interface that simplifies the population process of the registries.
  It allows also to create new kind of registries by providing the JSON Schema and a form definition compliant with the angular-schemaform framework which instructs it on how to render the form of the registry object.

Functionalities

Table 9 indicates the covered functionalities and their status:

Functionality	Description	Completed for release	Integrated (which level)
Registry object persistence	The MongoDB [4] database allows to store the registry documents into a NoSQL database.	Yes (1st release)	Complete lifecycle.
Create\Edit\Delete\Update registry object	Restheart SchemaForm UI allows CRUD operations on each registry.	Yes (1st release)	Complete lifecycle.
Browsing registry object	Registry Client Library is a Java client which allows the browsing of registries object.	Yes (1st release)	Complete lifecycle
Document Semantic Enrichment	Restheart SchemaForm UI will allow to enrich registry document with some semantic information in order to improve the discovery of the registries object.	No (2nd release)	

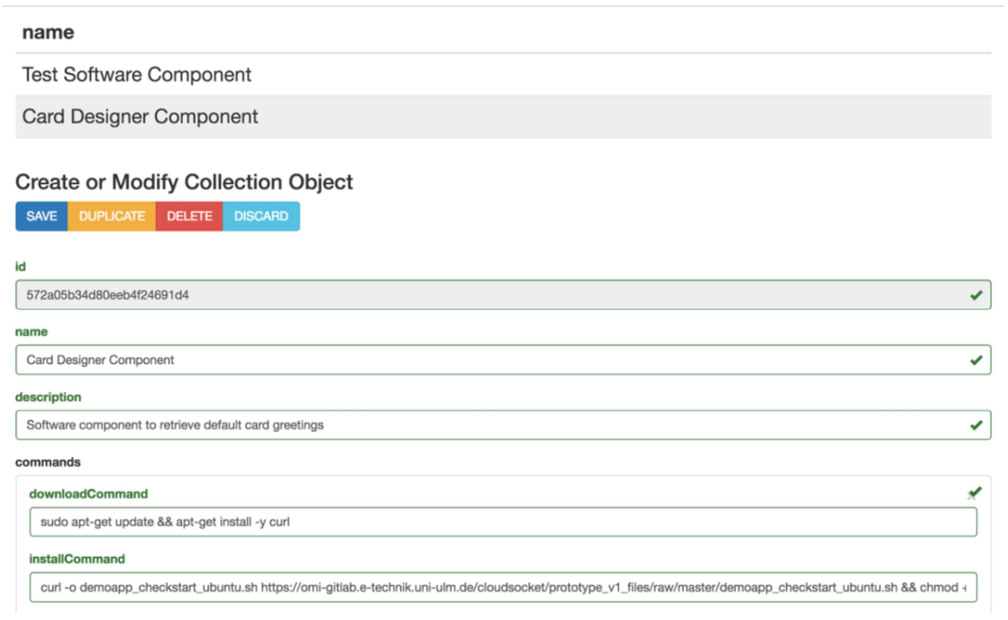
Table 9 - Functionalities of Repository Manager

Manuals

The manual, API description, handbooks are detailed in the alive wiki documentation: https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Repository+Manager+Component

The Repository Manager can be installed entirely as Docker [6] containers since all the components are available as Docker images.

In Figure 22 below there is a screenshot of the user interface, showing the Software Component Registry form.



name
Test Software Component
Card Designer Component

Create or Modify Collection Object

id

572a05b34d80eeb4f24691d4

name

Card Designer Component

description

Software component to retrieve default card greetings

commands

downloadCommand

```
sudo apt-get update && apt-get install -y curl
```

installCommand

```
curl -o demoapp_checkstart_ubuntu.sh https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/prototype_v1_files/raw/master/demoapp_checkstart_ubuntu.sh && chmod +
```

Figure 22 - User interface of the Software Component Registry

Since the registries are accessed by some of the CloudSocket environments, to avoid that each environment has its own client to access the registry contents, a common Registry Client Module has been built and is available as a Java library. The client has all the CRUD features to access the registries, it handles the authentication and all the functionalities for filtering and paginating the registry objects.

Download

The docker images can be found at the following url:

- mongodb: https://hub.docker.com/_/mongo/
- restheart: https://hub.docker.com/r/softinstigate/restheart/

The Restheart SchemaForm UI project can be found at the following url: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/restheart-schemaform-ui.

The Registry Client Library can be found at the following url: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/registry-client.

Instances

The prototype instance of the Repository Manager can be found at the following URL: http://134.60.64.221/.

3.3.2 BPaaS Execution Environment

The CloudSocket Execution Environment enables managing, monitoring and adapting the execution of the BPaaS bundles generated during the allocation phase, which have been published via the Marketplace. The environment covers the whole steps of execution phase: i) deploy and configure all components required to execute the bundles, which comprise the workflow, SLA, adaptation rules, and details of the third-party services involved; ii) allow to manage the BPaaS Customer's workflow instances, when a BPaaS workflow bundle has been deployed; iii) visualize the conformance levels to associated agreements and the respective monitoring data; iv) generate and manage the violations incurred as well as trigger the respective BPaaS bundle adaptation rules, allowing the environment to adapt the BPaaS instances to maintain the promised service level.

In order to support the aforementioned capabilities, the BPaaS Execution Environment comprises several components. These components are the following:

- Workflow Engine is responsible for deploying the executable workflows and executing the different workflow instances at the execution phase.
- Cloud Provider Engine is responsible for the complete deployment and lifecycle management of all the required components of the BPaaS bundle, including software components and VMs across multiple clouds, with transactional semantics (at least for the deployment part).
- Monitoring Engine is responsible for cross-level monitoring a BPaaS and correlate/aggregate monitoring data from different levels, from infrastructure and software services up to the level of the workflow.
- Adaptation Engine is responsible for the reconfiguration of the BPaaS deployment possibly in a cross-level manner (e.g., different services, service configurations and workflow structure modification) to resolve the problematic situations identified by adaptation rules.
- *SLA Engine* is responsible for generating, storing and observing the formal documents describing electronic service-level agreements (SLAs) between the parties involved in CloudSocket.
- *Process Data Mediator* is offered as a service to the Adaptation Engine in order to cover the cases where services are substituted and their replacements need to communicate properly with the rest of the services in the execution order of the current BPaaS workflow.

Figure 23 shows the detailed scheme of the CloudSocket Reference Architecture and the location of the different components.

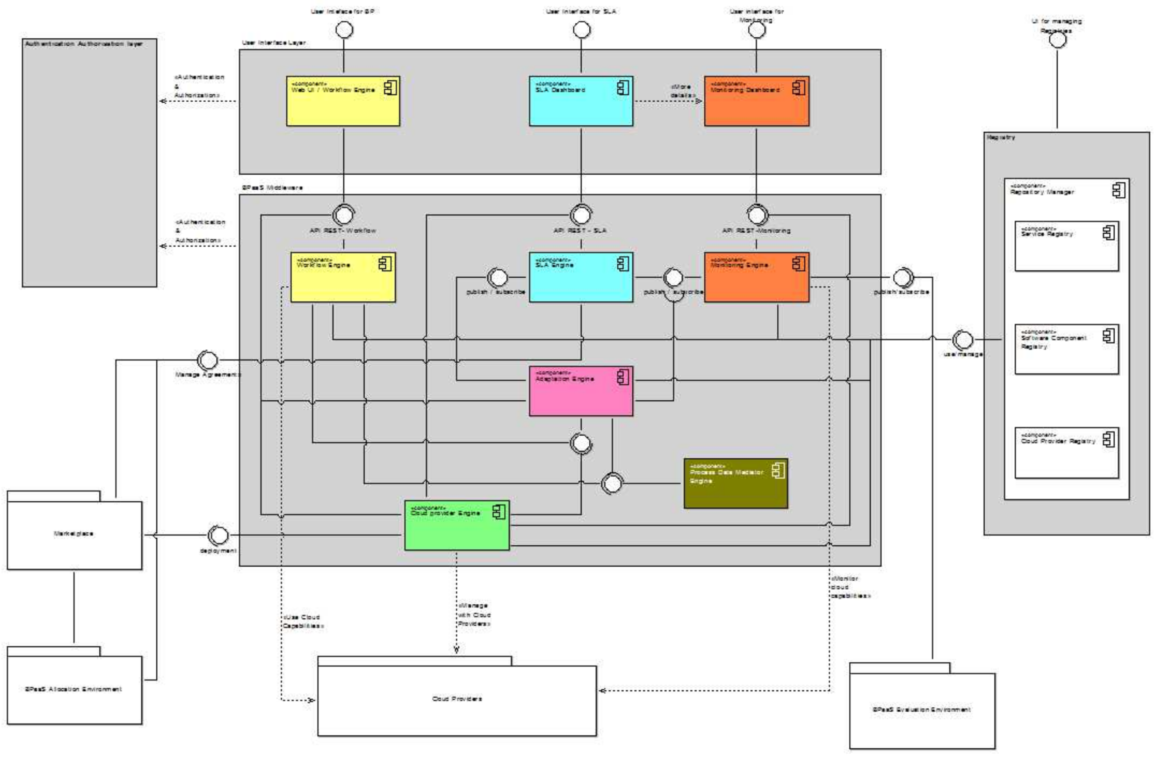

User interface for BP
User interface for SLA
User interface for Monitoring
UI for managing Registries
Authentication Authorisation layer
User Interface Layer
«Authentication & Authorization»
«component» Web UI / Workflow Engine
«component» SLA Dashboard
«More details»
«component» Monitoring Dashboard
BPaaS Middleware
«Authentication & Authorization»
API REST- Workflow
API REST - SLA
API REST -Monitoring
«component» Workflow Engine
«component» SLA Engine
«component» Monitoring Engine
publish / subscribe
publish / subscribe
publish/subscribe
use/manage
Manage Agreements
«component» Adaptation Engine
«component» Process Data Mediator Engine
«component» Cloud provider Engine
Marketplace
deployment
BPaaS Allocation Environment
«Use Cloud Capabilities»
«Manage with Cloud Providers»
«Monitor cloud capabilities»
Cloud Providers
BPaaS Evaluation Environment
Registry
«component» Repository Manager
«component» Service Registry
«component» Software Component Registry
«component» Cloud Provider Registry

Figure 23 - Internal architecture for the Execution Environment.

The architecture differentiates between two main levels: (a) one covering the components that have an interaction with the actors through a GUI, such as Web interface for the Workflow Engine, SLA dashboard and Monitoring dashboard and (b) another including the core engines to cover these functionalities. There is another hidden level, the data one which is covered internally by the internal architecture of each component but due to complexity reasons it is not shown in Figure 23.

The components in each level interact with each other in order to properly deliver and visualise the functionalities of the environment. More information about how these interactions are performed and what are the respective scenarios covered can be found at the following URL: https://www.cloudsocket.eu/uml/3-ExecutionEnvironment/remotedocu/modeldocu/27012016104208/modelContentHTML in which the corresponding UML diagrams [2] can be viewed. This information was also covered in the D4.1 deliverable [1].

In the following, we describe only those components that are part of this first release of the CloudSocket prototype. These components are the Cloud Provider Engine, Workflow Engine, SLA Engine, Monitoring Engine and Adaptation Engine. In this respect, it becomes apparent that the Process Data Mediator Engine has not been implemented yet and it will be incorporated in the next and final release of the CloudSocket prototype. This means that the automatic mediation to substitute different services with different interfaces is not yet supported by the BPaaS Execution Environment.

3.3.2.1 Workflow Engine

Summary

It is responsible for managing the deployment, execution and management of the different workflow instances at the execution phase. It is multi-tenant leading to executing a workflow instance on behalf of one organisation by also taking care of the corresponding workflow and organisation data level.

This component exposes two interfaces (see also the manual section):

- A graphical user interface for interacting with the different actors involved in the lifecycle of a workflow.
- A REST API interface allowing programmatic access to the different types of functionalities offered.

The main functionalities are:

- Deploy/redeploy a workflow in the workflow-engine, instantiate it and execute it.
- Manage (start, suspend, resume, stop…) and monitor the workflow instances, according to the workflow description in BPMN.
- Interact with manual tasks of the workflow.
- Manage the workflow engine environment such as the defined roles, users and tenants, the database and the running instances of the workflows.

Moreover, the executable workflows are designed by the integrated Editor Workflow based on the abstract workflows. Their design has to be aligned with the workflow engine, since the standard BPMN2.0 specification does not cover the complete definition of workflow execution details but it is allowed to be extended in order to cover it. As the deployment will be executed on the cloud, it is mandatory to define dynamically the self-contained executable workflows, which are included in the BPaaS bundle, following these standards.

The main roles to interact with the functionalities are:

- CloudSocket Customer, who wants to use the purchased BPaaS bundles, which has been deployed automatically in the cloud. The customer can interact via the use of different roles: i) Knowledge Worker is responsible for managing the manual task of the workflows; ii) Process Responsible (technical skills) is in charge of dealing with the workflow engine, managing the workflow instances and following their status.
- Broker, who is responsible for the configuration of the Workflow Engine, for performing tasks on behalf of customers (e.g., workflow instances actions), for inspecting/monitoring the status of all workflows deployed and especially the status of respective SLAs.
- Platform operator, who may be the broker itself, and is responsible for checking if the Workflow Engine works as expected.

Additionally, there is a role that covers the functionalities for creating the executable workflow, which will be included in the BPaaS bundle, during the design phase. Therefore, this role is not directly related with the execution phase, but the internal Workflow Designer component is completely integrated with the whole Workflow Engine component.

- Workflow Designer, which is responsible to design the executable workflows using the executable workflow editor in order to include the necessary information to transform the abstract workflows (independent of the workflow engine) to the executable workflows (associated to a specific workflow engine).

All the actors will interact directly with the graphical user interface and the system will manage automatically the authorization of the appropriate functionalities (see the user guide in the Manual section).

The following table indicates the details of the component.

Type of ownership	Extension
Original tool	FIWARE and Fed4FIRE project
Planned OS License	Apache License Version 2.0.[7]
Reference community	Activiti community
Lead Partner	ATOS

Table 10 - Details of the Workflow Engine

Comprises

- Web UI of workflow Engine (Graphical User interface)
- Workflow Engine (core functionalities)

Depends on

- IdM Marketplace
- Repository Manager

Architecture design

The general architecture of this component can be viewed by checking the components coloured in yellow, in Figure 23.

Figure 24 introduces in more detail the different internal components of the Workflow Engine. The components are split into two main parts: i) one part is responsible for exposing the interfaces with the external actors (other components, BPaaS Customers or Brokers); ii) the other is responsible for managing the business logic and the data layer.

The two layers allow decoupling the components between the presentation layer, which is directly related to the exposed interfaces (Graphical user interfaces and REST API) and the backend, which is responsible to provide all the functionalities and their persistence. Thus, the interface layer will focus on the look and feel or how to expose these interfaces. Moreover it will manage the necessary calls to the backend layer, which will execute the actions.

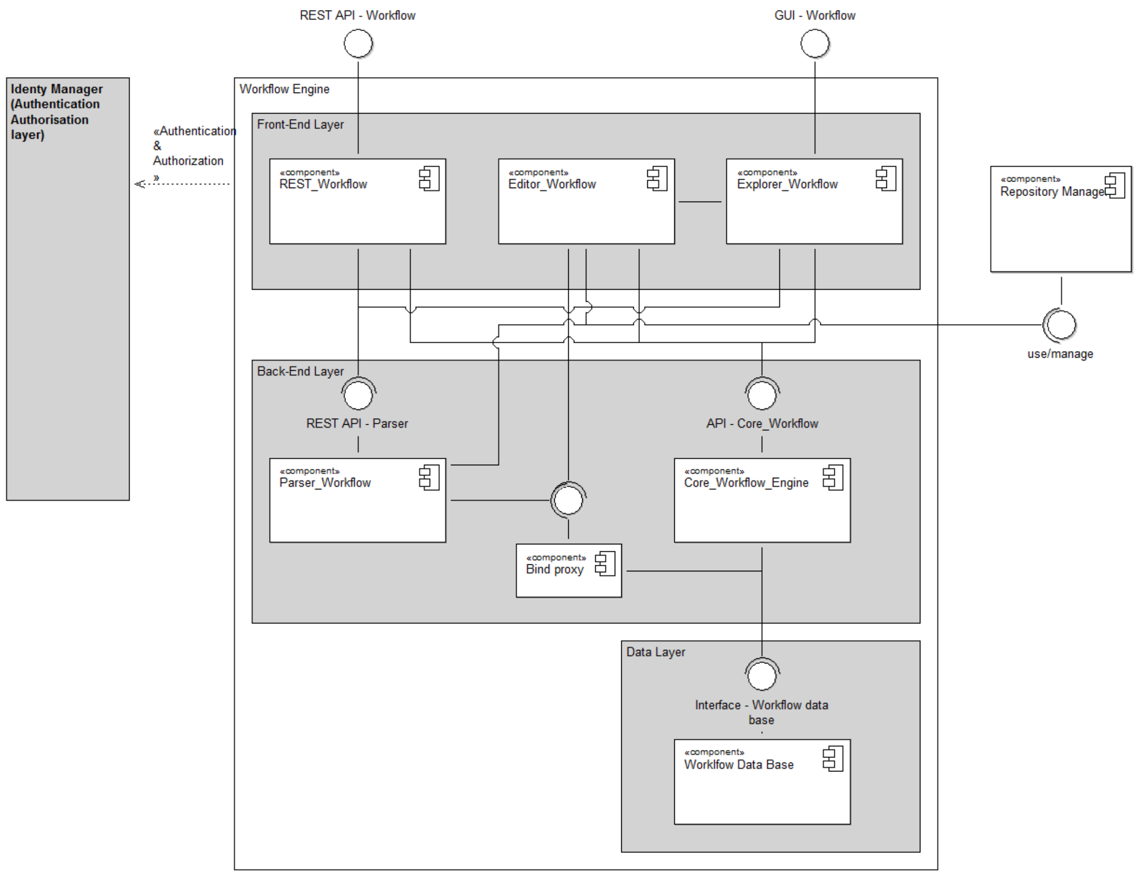

REST API - Workflow
GUI - Workflow
Identy Manager (Authentication Authorisation layer)
«Authentication & Authorization »
Workflow Engine
Front-End Layer
«component» REST_Workflow
«component» Editor_Workflow
«component» Explorer_Workflow
«component» Repository Manager
use/manage
Back-End Layer
REST API - Parser
API - Core_Workflow
«component» Parser_Workflow
«component» Core_Workflow_Engine
«component» Bind proxy
Data Layer
Interface - Workflow data base
«component» Worklfow Data Base

Figure 24 - Internal architecture for the Workflow Engine.

Front End layer:

- REST Workflow module is responsible for exposing all the functionalities of the Workflow Engine through a REST API, allowing other components to interact with the engine programmatically, without human interaction. The main interactions are with the Cloud provider Engine (Section 3.3.2.3), which is responsible to orchestrate the deployment of the BPaaS bundle in the cloud environment, and the Adaptation Engine (Section 3.3.2.4), which is responsible to modify the environments and the currently running workflow instance in order to cover the new conditions.
- Editor Workflow module is responsible for the editing of the executable workflows at the design phase. A graphical user interface is used for editing, modifying and generating the executable workflows, while the involved actors are the workflow designer, who can be contracted by the brokers to take care of the workflow design task. However, the module cannot interact directly with the customers and their companies; it is thus exploitable only by brokers. Nevertheless, this editor is necessary to align the definition of the concrete workflows with the technology adopted in the workflow engine in order to deploy and execute them correctly in the cloud environment.
- Explorer Workflow module is responsible for exposing a graphical user interface in order to interact with the human actors, such as brokers and customers. Hence, through this exposed dashboard, the different actors can manage the complete lifecycle of the deployed workflows, allowing interacting with the platform in an easy way, increasing the quality of experience (QoE).

Back End Layer

- Core Workflow Engine module is responsible for managing all the functionalities of the Workflow Engine. Hence, the complexity of the business logic of all these functionalities is delegated to this module and an interface is exposed to interact with them. It is also responsible to interact with the data layer and to persist the workflow, its instances and the rest of the entities.
- Workflow Parser module is responsible for managing the functionalities related to workflow parsing, such as the introduction of the real endpoints of the software components and the automatic generation of the service task tags to invoke WS and RESTFul services.
- Bind Proxy is a small module to facilitate the interaction with the database. It is responsible for managing the binding between service tasks and the associated services. It has been introduced to avoid using the Core Workflow Engine module as a proxy, to only connect with the database in order to manage the binding actions. In this manner, using this module, the Workflow Parser is decoupled from the Core Workflow Engine.

Data layer

- Workflow Data Base module is responsible for persisting all the data in order to support all the functionalities. The model definition is based on an entity-relationship schema to represent all the entities, including tenants, roles, workflows, instances, jobs… It exposes a standard interface to connect with the data base; nevertheless it is not its responsibility to create an Object-relational mapping to interact with the entities such as JPA in java or SQLAlchemy [10] in Python [11], since that is delegated to the Core Workflow Engine and the Bind Proxy to provide it.

All the components expose different types of interfaces in order to cover nonfunctional requirements, such as scalability, modularity and replaceability. Nevertheless, some of them are more coupled than others, since they work to provide features together.

- The REST Workflow, Explorer Workflow and Core Workflow Engine components are working jointly in order to arrange the workflow execution. Therefore, it is possible to replace these components with implementations such as Camunda [12] or BonitaSoft [13]; and re-implementing the adaptation and extension developed in these components based on Activiti [14].
- The Editor Workflow is partially independent of the base line workflow engine as its main duty is to create the executable workflow to be deployed in the specific engine, which might include concrete extensions on the standard BPMN. In this sense, this component could also be extended in order to work for other workflow engines. In addition, it could also be replaced by another workflow editing component which could be coupled to a specific workflow engine like Activiti or the one which totally replaces the agglomeration of the previously referred components in the previous bullet.
- The Parser Workflow Manager and Bind Proxy have isolated functionalities and expose a Restful interface. Hence, it is feasible to substitute them by other components, if they maintain the exposed APIs.
- Finally, the Workflow Engine use a persistence framework with support for custom SQL, stored procedures and advanced mappings, this allow to substitute the Workflow Data Base module by other relational databases as long as the new option follows the standard connectors, since this persistence framework creates an abstraction layer between the entities and their persistence in the data base

Functionalities

Table 11 indicates the covered functionalities and their status:

Functionality	Description	Completed for release	Integrated (which level)
Deployment of a BPaaS bundle workflow	The component will allow deploying the executable workflow included in the BPaaS bundle.	Yes (1st release)	Complete lifecycle.
Workflow Instance Management	Manage and follow the status of workflow instances, according to the workflow description in the BPaaS bundle.	Yes (1st release)	Complete lifecycle
Manage the workflow execution.	When workflow instances are running, all kinds of actions, such as service tasks (invoke remote services) and user tasks (interact with the customer), need to be properly managed according to the workflow definition.	Yes (1st release)	Complete lifecycle
End-Point adaptations for the deployment phase.	Adaptation of the real endpoints for the atomic services and software components when the workflow is deployed	Yes (1st release)	Complete lifecycle.
Multi-Tenant	The component has to be deployed for multiples tenants using the same workflow engine instance. This allows introducing a SaaS-based solution for CloudSocket.	Yes (1st release)	Complete lifecycle.
Manage the workflow instance engine environment	The component has to manage and follow up the complete workflow instances and the deployed BPaaS bundles for the different tenants	Yes (1st release)	Complete lifecycle
Design of Executable workflows	The component has to assist the creation of the executable workflows, including the calls to the associated services.	Yes (1st release)	Complete lifecycle Integrated, but it needs to be improved to be aligned with the registry and the automatic code generation (2nd release).
Assist in the creation of the executables workflows.	The component has to be integrated with the Registry in order to facilitate the creation of the executable workflows, besides generating automatic code based on this registry data.	Partially (1st release)	Complete lifecycle. Only integrated with the software component registry.
Monitor at the workflow level	The component has to provide the feedback for the defined metrics at the level of the workflow to the monitor engine.	Partially (1st release)	Complete Lifecycle. The component provides some simple metrics in order to be integrated with the monitoring.
Centralized authentication and authorization	The component will delegate the authentication and authorization to an external Identity Manager, following the standards (OAuth2 and SCIM).	No (2nd release)	Working on the adaptation of the component to support the standards. Now, the component uses internal authentication and

			authorization solution.
End-Point adaptations for the execution phase	Adaptation of the real endpoints for the atomic services and software components when the workflow is executed and the Adaptation Engine decides to modify a workflow execution to handle undesired situations.	No (2nd release)	
Discover of the most appropriated services to be included in the BPaaS bundle	The component will use more complex searching mechanisms in order to assist in the workflow design via employing a semantic approach.	No (2nd release)	

Table 11 Functionalities of the Workflow Engine

Table 11 highlights these functionalities while the details about the workitems and the respective functionality realisation status can be found at the follow-up Excel file, which has been referenced in Section 2.1. Details about the interactions required to realise them can be found at D4.1 [1] and especially the EE-UC-1-Deployment of BPaaS, EE-UC-2 – Execution of the BPaaS and EE-UC-4 – Workflow Environment Management along with respective UML diagrams [2].

Manuals

The installation manual, the API description, unit test and handbooks are detailed in the alive wiki documentation https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Workflow+Engine+Component

The component is developed in java based on Activiti 5.18; several frameworks are used such as Vaadin [8], Spring [9] and mybatis [15]. A MySQL database is used [16].

As a build automation tool, the components have used maven [26], which describes how they are built, and their dependencies. This allows building and generating automatically the artifacts for the different environments (development and production) and their configuration for example with Eclipse. Besides, the use of this artifact will facilitate the continuous integration, which will be covered in Task T4.5 CloudSocket Integration and Consolidation.

The installation manual takes advantage of these tools, facilitating the installation at the different environments and describing the necessary steps to install the component and its modules. Figure 25 indicates the generated artefacts and their dependencies.

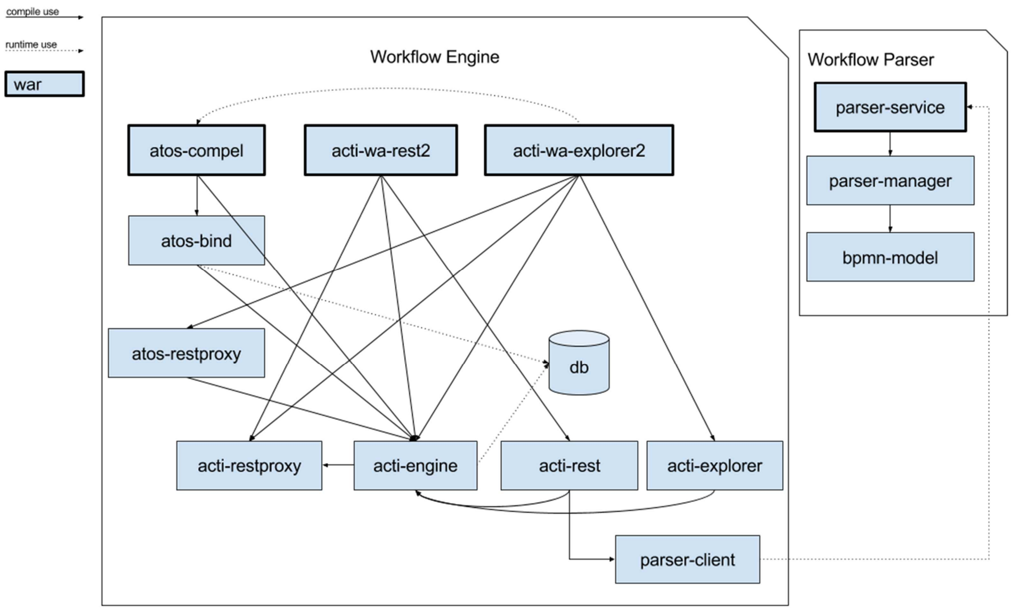


Figure 25- Artefacts of the Workflow Engine.

The handbook is split in two parts to cover the exposed interfaces:

- RESTful Interface is described to indicate the header, the method, the json structure of the request and the response. Besides, it is included the associated unit tests.
- User manual is responsible to explain the covered functionalities through the screenshots and description of the interface.

Download

The source code has been published in the CloudSocket GitLab repository [20]. The developers can clone the repository for the different components and follow the installation manual.

As it was indicated in the Architecture section, the REST Workflow, Explorer Workflow, Bind Proxy and Core Workflow Engine components are working together and the code is available at the link https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/workflow-engine. The Workflow_Parser component is available at the link https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/workflow-parser.

The schema of the data base is created automatically during the first installation; hence it is not necessary to have the dump file or the script creation.

The download section of the CloudSocket portal [23] contains the software components for the Execution Environment.

Instances

There is an available instance deployed on the UULM infrastructure to integrate the first prototype: http://134.60.64.132/activiti-webapp-explorer2/. This instance is published as software as a service for multitenancy in the cloud and it is completely integrated with the rest of components of the Execution Environment

Figure 26 - Instance of the Workflow Engine.

3.3.2.2 Monitoring Engine

Summary

It is responsible to monitor a BPaaS and correlate/aggregate monitoring data from different levels, from infrastructure and software services up to the level of workflows. It exposes an API in order to allow components, such as the SLA Manager and the BPaaS Evaluation Environment, to draw the information monitored by subscribing to particular metrics and perform respective related tasks. In the case of the SLA Manager, this will involve performing an SLO evaluation, while, in the case of the BPaaS Evaluation Environment, this will involve performing background analysis of the monitored information in order to discover interesting patterns in the context of one or more business processes. The component will cover both raw metrics (direct measurements provided by deployed sensors or external measurement systems like PaaSs) and aggregated metrics (formulas to exploit metrics are already implemented and produce the respective aggregated measurements). This component will also handle the monitoring of contextual information which will be handed over (through exposing a particular API) to the Adaptation Engine to enable it to completely assess adaptation rules.

As metrics are involved in SLO and contextual conditions, it is essential that they need to be defined beforehand in order to allow the Monitoring Engine to measure them and thus enable the evaluation of such conditions. The BPaaS Evaluation Environment has an interface to this component via the publish-subscribe mechanism in order to have the respective support to its analysis activities, especially with respect to the assessment of KPIs.

This component will expose:

- A REST interface to configure sensors
  - Input: Description of metrics (Sensor configuration for raw metrics and metric formula description to allow aggregation); Reference to user-defined sensors as software components in the respective registry (which includes how to download the implementation, for example as a rar file);
- Java-based interface to be implemented for user-defined sensors.
- A Time-Series Database system to store the measurements
  - Output: Measurement DB (which can be realized by a time series database or a semantic database or even both); Context DB (storage & update of contextual information); Notification system (measurements as notification to subscribers).

The main functionalities are:

- Provide a framework for sensors providing measurements of:
    - Raw and composite metrics
    - Platform-specific metrics (domain-independent metrics)
    - BPaaS-specific metrics (domain-dependent metrics)
- Allow implementation and description of user-defined metrics
- Monitor and aggregate metrics
- Metric specification modification leading to changing the monitoring infrastructure/environment, and new metrics definition for a BPaaS
- Inform interested parties about fresh metric values through the publish/subscribe mechanism
- Inform interested parties about historical metric values through the use of the REST-API
- Inform interested parties about contextual information

The following table indicates the details of the component.

Type of ownership	Extension
Original tool	Visor (of Cloudiator framework)
Planned OS License	Apache License Version 2.0. [7]
Reference community	Communities around Cloud Monitoring, Cloudiator developers, mainly UULM
Lead Partners	UULM, FORTH

Table 12 - Functionalities of the Monitoring Engine

Comprises

- REST interface, also wrapped and used by the Cloud Provider Engine - see Section 3.3.2.3 (User interface)
    - Based on the Colosseum API of Cloudiator and currently also KairosDB as TSDB
- Sensor and measurement system (core functionalities)
    - Based on Visor and Colosseum of the Cloudiator tool suite

Depends on

- Loosely depending on CAMEL due to the definition of metrics
- Cloud Provider Engine for deploying part of the monitoring engine (e.g., sensors, collectors, TSDBs) in respective clouds whose services are to be monitored

Architecture design

The component is part of the open-source software Colosseum of the Cloudiator framework, which is used to realize the Cloud Provider Engine. We will go into detail in the architecture section of the Cloud Provider Engine - see Section 3.3.2.3.

Replaceability: The monitoring is very generic and can be fed with measurements of external software, so it is open to use additional software for this task. The whole monitoring engine could be substituted by another one provided that the respective exposed interfaces are supported. However, please bear in mind that currently this engine depends on the Cloud Provider Engine for the deployment of some of its parts. In this sense, in order to completely substitute this engine, we also need to modify the way the monitoring deployment is performed by the Cloud Provider Engine.

Functionalities

Functionality	Description	Completed for release	Integrated (which level)
Measure Raw Metrics	Ability to measure raw metrics and store them in a (time series) database	Yes (1st release)	Complete lifecycle.
Distribute TSDB	Distribute the TSDBs in a hierarchical and adjustable manner	Partly (1st release)	Already able to distribute raw monitoring data, but missing distribution of composed metrics across the clouds.
Sensor Interface	Provide an interface to be implemented by BPaaS-specific/broker-defined sensors	Yes (1st release)	Complete lifecycle.
Remote Dynamic Sensors	Provide a way to dynamically integrate new sensors in the running instance remotely	No (2nd release)	-
Aggregate Metrics	Definition of aggregated metrics	Partly (1st release)	Basic composite metrics are possible, while more complex metric composition scenarios are planned to be supported (e.g., including ordered and dynamic compositions).
Higher Level Sensors	All levels of BPaaS-specific metrics also covering PaaS, SaaS and WfaaS	No (2nd release)	Currently only loosely coupled BPaaS-specific metrics by the Workflow Engine.
Metric Subscription	Provide a way to subscribe to metrics	Partly (1st release)	So-called Monitor Subscriptions already possible but not yet integrated.
Metric Report	Push interface to integrate third-party monitoring tools	Partly (1st release)	The monitoring agent Visor already provides a push interface, but not yet integrated.
Interfacing with the SLA Engine	Adjust the reporting tool to create measurements in a usable format for the SLA Engine	No (2nd release)	-
IdM Integration	Integrate with the IdM	No (2nd release)	-
Multiple Report Modules	Allow to report to multiple different receiving units in one instance of the monitoring agent. A receiving unit is currently the	No (2nd release)	-

	TSDB but could be simultaneously be a log system in order to e.g., realize fault-tolerance		

Table 13 - Functionalities of the Monitoring Engine

Table 13 highlights these functionalities while details about the workitems and the respective functionality realisation status can be found at the follow-up excel file, which has been referenced in Section 2.1. Details about the interactions required to realise them can be found at D4.1 [1] and especially the EE-UC-1-Deployment of BPaaS, EE-UC-2 – Execution of the BPaaS and EE-UC-3 – Monitoring of Agreement Status along with respective UML diagrams [2].

Manuals

We refer for this component to the general documentation of the main project under:
https://github.com/cloudiator/visor

We do not have a separate Wiki page for this, since we wanted to have the interface compatible with the main project. Also the sensors are configured through the main REST interface of the Cloud Provider Engine (Colosseum subcomponent).

Download

The official version can be downloaded from: https://www.cloudsocket.eu/download

The monitoring agent that gets deployed on the VMs as well as the basic sensors can be downloaded from github. It features also an own branch for the CloudSocket-specific features, which are not needed for the general purpose of this project: https://github.com/cloudiator/visor.git

The components are released here under the Apache 2.0 license.

Instances

The instances depend on the current deployment. Each deployed virtual machine will be delivered with an instance of the monitoring agent. The monitoring agents gather the measurements. The REST interface of the Cloud Provider Engine is used to configure the Monitoring Engine. An instance of the Monitoring Engine is deployed in a VM at the OpenStack cloud of UULM and is available at the following URL: http://134.60.64.155:8080

3.3.2.3 Cloud Provider Engine

Summary

This component is responsible for the complete deployment and lifecycle management of all the required components of the BPaaS, including software components and VMs across multiple clouds, with transactional semantics (at least for the deployment part). These capabilities will be managed by different subcomponents of the Cloud Provider Engine to provide a modular, flexible and scalable architecture. To exhibit these capabilities, the Cloud Provider Engine will build upon existing functionalities offered through the interfaces exposed by the cloud providers.

The BPaaS deployment with transactional semantics will be managed by the Deployment Engine, responsible to deploy, configure and operate all appropriate software and VM components before deploying a workflow in the Workflow Engine. In essence, the Deployment Engine will be responsible for executing the deployment plan included in the BPaaS bundle by orchestrating all necessary steps. This deployment plan will not only cover component deployment but also agreement registration and validation, monitoring infrastructure deployment and configuration, and workflow deployment in the Workflow Engine in order to guarantee that in the end the workflow of the BPaaS bundle will be ready for execution. If something goes wrong, then the transactional/failure semantics, which is defined as part of the deployment plan, will dictate what actions will have to be performed to remedy for this, which could involve rolling back the system or compensating previous deployment actions and performing new ones with the same goal.

The Cloud Provider Engine will expose two interfaces: (a) an interface to enable the performance of re-deployment actions in order to interact with the Adaptation Engine component, and (b) an interface to interact internally with the Deployment Engine for managing deployment transactionally.

The Deployment Engine sub-component will comprise different plug-ins to connect to the different clouds (by also exploiting previously generated end-user cloud credentials), allowing to interact with these clouds to execute a common action as, e.g., the concrete deployment actions for a VM will be different depending on the cloud provider, but the actual abstract action is the same: deploy a VM. Hence, this sub-component will allow providing an abstraction over the different specificities of cloud providers with respect to cloud management actions and it will be responsible for transforming abstract management actions to cloud-specific ones.

This component will expose:

- A REST interface to configure the deployment onto different cloud providers
    - Input: Deployable workflow (BPaaS bundle integrated via CAMEL adapter):
        - BPMN file (workflow definition)
        - SLA agreement definition (including Service Level Objectives with conditions over quality metrics)
        - List of adaptation and alternative allocation rules to be applied, whereas the allocation rules are currently hardwired into the CAMEL deployment plan.
        - Semantic Metadata (describing metrics or adaptation actions) to allow taking decisions in the execution phase.
    - Deployment plan along with transactional semantics, further deployment actions to be executed in the context of adaptation rules which are dictated by the Adaptation Engine (as part of the BPaaS bundle)
    - Output: The status and result of the current deployment plan can be viewed constantly on run-time.
- A REST interface to configure the service instances during run-time

The main functionalities are:

- Manage the complete BPaaS deployment.
- Manage and abstract from current IaaS capabilities and later also from PaaS capabilities.
- Manage the relationships with the different cloud providers.
- Offer scaling & migration capabilities to the Adaptation Engine

The following table indicates the details of the component.

Type of ownership	Extension & Adaptation *(for some components also Creation)*
Original tool	Colosseum, Sword and Lance (of Cloudiator framework)
Planned OS License	Apache License Version 2.0. [7]
Reference community	Cloudiator developers, mainly UULM
Lead Partner	UULM

Table 14 - Description of the Cloud Provider Engine

Comprises

- REST interface, wrapped by the Execution Environment Entrypoint
- Cloudiator framework
- Execution Environment Entrypoint
- CAMEL Adapter

Depends on

- Workflow Engine
- Monitoring Engine
- Adaptation Engine
- SLA Engine

Architecture design

Technically speaking, the Cloud Provider Engine relies on the Cloudiator framework. This consists among others on the Colosseum, Lance, Sword, Axe and Visor components.

The main part is the Colosseum, which is the entry point of the framework and engages the other components. Cloudiator works in two domains: the home domain and the remote domain. The home domain is the central point where this so-called cloud orchestration tool is run from. Figure 27 shows the home domain, in which Colosseum, Axe and Sword run. Colosseum has components to: (a) discover the offers of cloud providers; (b) store the offers and metadata in registries; (c) manage the deployment of application components and (d) justify about the best-suited configurations (via the use of a broker). Axe consists of the Scaling Engine, which evaluates the monitoring data from the Time-Series Databases and reacts on described situations (incarnated in adaptation rules), e.g. by enacting scaling actions. Sword is the abstraction layer from any Cloud Provider. Currently only abstraction on IaaS level is supported, but it is work-in-progress to also support the PaaS level.

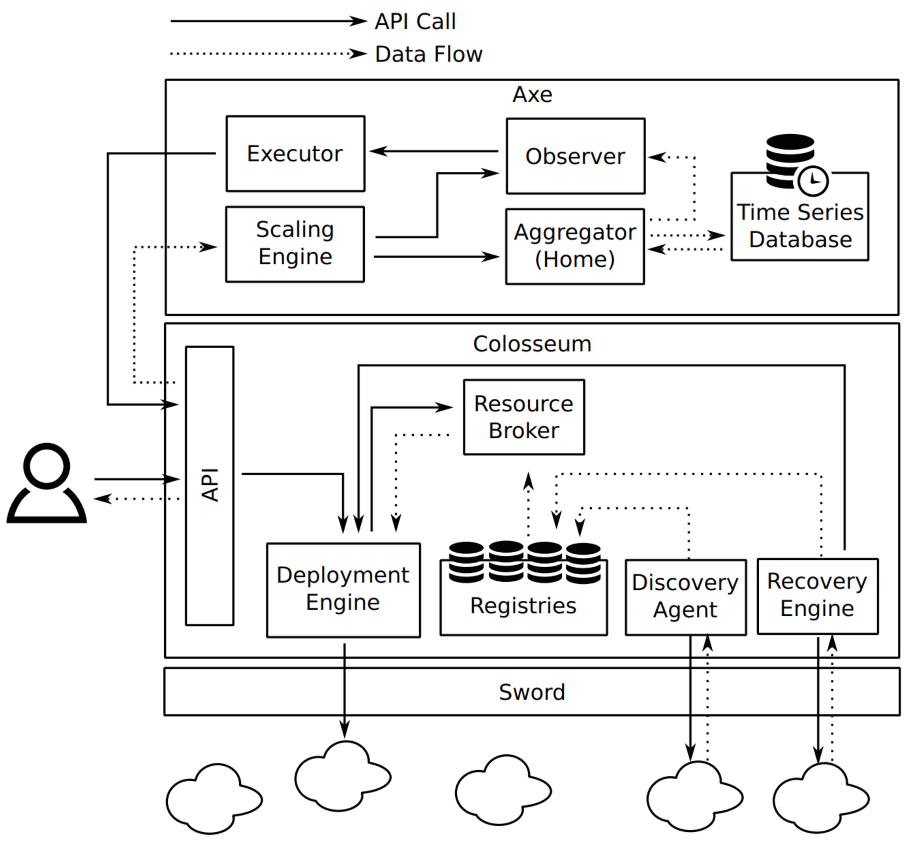


Figure 27 - Internal architecture for the Cloudiator framework of the home domain.

The remote domain is deployed along with the deployment of any virtual machine. Figure 28 shows the respective components involved. The Lifecycle Agent packs application components into a container and takes care of executing life-cycle actions. The Monitor Agent (Visor) senses the platform and application, and pushes the data into a local Time-Series Database (TSDB). The Aggregator (Axe) can aggregate and reason over existing metrics (from local or remote TSDB). Finally, it executes adaptation actions or stores aggregated values into a local TSDB.

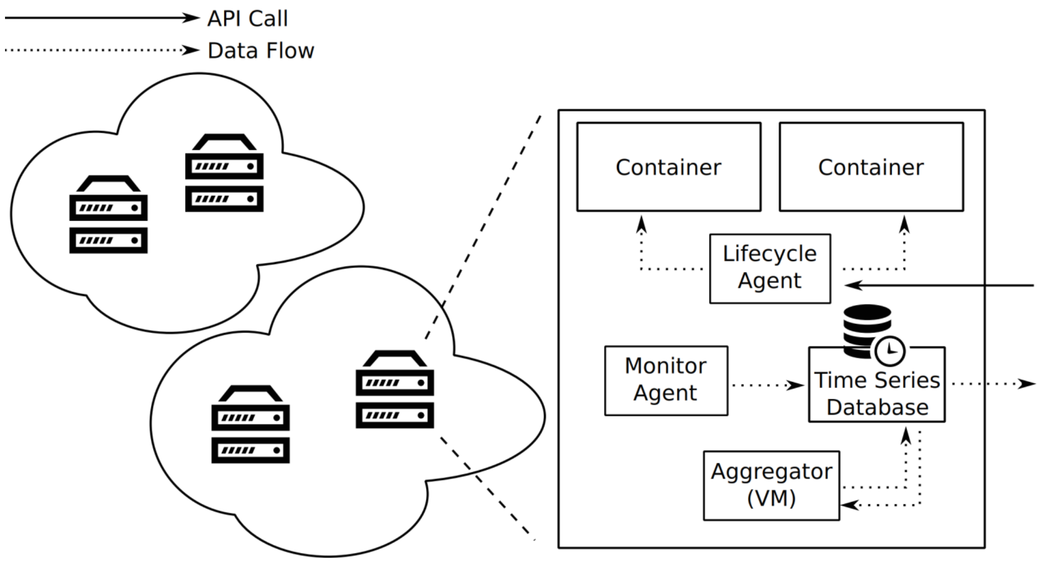


Figure 28 - Internal architecture for the Cloudiator framework of the remote domain.

Functionalities

Functionality	Description	Completed for release	Integrated (which level)
Deploy BPaaS bundle	Use the deployment plan of the BPaaS bundle to instantiate the cloud services at different cloud service providers	Yes (1st release)	An enhanced configuration interface is under development
Mange multi-cloud deployments on IaaS level	Deploy services across different cloud providers by abstracting the underlying cloud provider	Yes (1st release)	Complete lifecycle.
Manage Cloud Providers on higher service levels	Manage Cloud Provisioning on PaaS and SaaS level. For each level an abstraction layer should be provided.	Partly (1st release)	A proposal for the PaaS integration in CAMEL is in the making of the research documented in D3.3
Complex Life Cycle Actions	In addition to the generic imperative life-cycle actions, the Cloud Provider Engine should be open to more complex actions and a user-defined execution of them	No (2nd release)	-
Integration of well-known DevOps tools	The application component configuration should additionaly specified via Chef recipes or Puppet modules	No (2nd release)	-
Simplified access to the log files	The user should have an easy and supportive way to access the logs	Partly (1st release)	The raw output was added to the entry point, but not for each component instance and VM.
Web GUI	In addition to the REST interface, a GUI should support the user	Partly (1st release)	The web-based GUI for the Cloud Provider Engine is not yet finished.
Integration of IdM	Switch from user/pw to token-based authentication against cloud providers	No (2nd release)	-
Interfaces for Adaptation Engine	Extended API to offer adaptation actions (on each level) and their analysis on VM, platform and service level	Partly (1st release)	Simple horizontal scaling is already integrated, complex adaptation plans missing in the Cloud Provider Engine.

Table 15 - Functionalities of the Cloud Provider Engine

Table 15 highlights these functionalities while details about the workitems and the respective functionality realisation status can be found at the follow-up excel file, which has been referenced in Section 2.1. Details about the interactions required to realise them can be found at D4.1 [1] and especially the EE-UC-1-Deployment of BPaaS along with respective UML diagrams [2].

Manuals

We refer for this component to the general documentation of the main project under: https://github.com/cloudiator

We do not have a separate Wiki page for this, since we wanted to have the interface compatible with the main project. Also the sensors are configured through the main REST interface of the Cloud Provider Engine (Colosseum subcomponent).

The REST interface for the BPaaS bundles is provided in the Entrypoint: https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Execution+Environment+Entrypoint

Download

The sources are hosted on github. All projects are referenced on the main page. Where applicable, the repositories feature an own branch (namely cs-master) for the CloudSocket-specific features, which are not needed for the general purpose of the Cloudiator project: https://github.com/cloudiator

The components are released under the Apache 2.0 license.

The Entry-point, as a wrapper for the Cloudiator toolset, can be found here: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/execution-environment-simple-entrypoint

Instances

The colosseum component is deployed in a VM at the Openstack cloud of UULM and is available at the following URL: http://134.60.64.155:9000

The Entrypoint is deployed in the same VM and available at the following URL: http://134.60.64.155:9012/job

3.3.2.4 Adaptation Engine

Summary

This component is responsible for the reconfiguration of the BPaaS deployment (different services, modified service configuration and workflow structure) to resolve the problematic situations identified by adaptation rules. Different types of adaptations will be performed at different levels of abstraction. In particular, adaptation actions on VMs (deploy, migrate), services (substitute, rebind), workflow tasks (e.g., re-execute, map to different service compositions) and the workflows themselves (recompose workflow) shall be offered.

The management of the adaptation rules will be covered by a subcomponent called Rule Engine, which is responsible to take decisions based on the environment variables and the performance levels encountered. So, it should interact with the Monitoring Engine and the SLA Engine to collect the needed data required for the evaluation of the rules exploited. The required data include SLO violations communicated by the SLA Engine as well as contextual information produced by the Monitoring Engine (in which the Rule Engine needs to subscribe). Such data are required in order to assess SLO and contextual conditions which constitute the left part of adaptation rules. In case that one or more rules are triggered, the respective adaptation actions will be executed - based on the settings from the BPaaS Allocation Environment - to maintain the quality of the service promised and of the experience of the stakeholders by, e.g., migrating services to other providers and deploying software components over different clouds.

The need for adaptation will be indicated by the Rule Engine as it possesses the knowledge of the adaptation rules (covering scalability and the fault-tolerance) and be supported by the Adaptation Engine which includes an adaptation library of actions that can be exploited to perform the different types of adaptation needed at the different levels. The Adaptation Engine will need to perform either one or more adaptation actions. In the first case, it will be responsible for just executing or delegating this action to another component (e.g., Cloud Provider Engine). In the second case, the actions to be executed will be described in a form of a workflow which will also dictate the sequence in which the adaptation actions have to be run; hence, there is a need for a (possibly internal to the Adaptation Engine) workflow engine able to execute the adaptation workflows.

This component will expose:

- A REST interface to manage the rules:
  - Input: Violations, Metrics, description of adaptation rules (including event patterns that lead to their triggering) to adapt a BPaaS, policies for adaptation (e.g., max amount of VMs or service instances, cost limits), which are specified in the Allocation environment. The following functionalities are covered:
    - register adaptation rules,
    - modify adaptation rules on demand,
    - get adaptation history for a certain BPaaS,
    - allow executing normal as well as personalized/customized adaptations.
  - Referencing of the actual adaptation workflow (by the Rule Engine which is a sub-component of the Adaptation Engine) in rules stored in the Rule Engine, part of the Adaptation Engine, to allow its instantiation when a respective adaptation need arises.
  - Keep track of the result of the executed adaptation

The main functionalities are:

- Evaluate adaptation rules which involves assessing contextual and SLO conditions (where the latter are already evaluated and sent in the form of SLO violations to the Adaptation Engine)
- Manage the execution of an adaptation strategy (in the context of the triggering of a specific rule)
- Adapt the BPaaS according to the adaptation strategy provided.

The following table indicates the details of the component.

Type of ownership	Extension & Adaptation
Original tool	Axe (of Cloudiator framework)
Planned OS License	Apache License Version 2.0. [7]
Reference community	Cloudiator developers, mainly UULM
Lead Partner	UULM, FORTH

Table 16 - Description of the Adaption Engine

Comprises

- Rule Engine (i.e. the Scaling Engine of Colosseum)
- Distributed aggregators

Depends on

- Cloud Provider Engine
- Monitoring Engine

Architecture design

The Adaptation Engine highly relies on the Axe component and the Scaling Engine of the Colosseum component of the Cloudiator framework, which is used for the Cloud Provider Engine. You can see the architecture of Axe in the Cloudiator framework in Figure 27.

Functionalities

Functionality	**Description**	**Completed for release**	**Integrated (which level)**
Basic Scaling Functionality	Provide a way to describe simple threshold-based rules to trigger horizontal scaling actions.	Yes (1st release)	Complete lifecycle.
Event / Measurement Registration	Register to Monitoring & SLA Engines to obtain events for the evaluation of adaptation rules	Partly (1st release)	Registration is not completed, but events can be attached as sensors and the adaptation/scaling rules can be specified in CAMEL.
Obtain adaptation rules	Expose interface to retrieve adaptation rules from the Cloud Provider Engine	No (2nd release)	-
Realise adaptation functionality	Map events to adaptation rules and start their execution	Partly (1st release)	Currently available for simple adaptation/scaling actions.
Adaptation plan	Provide a sophisticated way to define the workflow of an adaptation action and its interdependence	Partly (1st release)	The current definition effort for the integration in CAMEL is described in D3.3.
Migration of stateful services	Ability to migrate stateful service to another cloud provider	No (2nd release)	-

Table 17 - Functionalities of the Adaption Engine

Table 17 highlights these functionalities while details about the workitems and the respective functionality realisation status can be found at the follow-up excel file, which has been referenced in Section 2.1. Details about the interactions required to realise them can be found at D4.1 [1] and especially the EE-UC-1-Deployment of BPaaS and EE-UC-2 – Execution of the BPaaS along with respective UML diagrams [2].

Manuals

We refer for this component to the general documentation of the main project under:

https://github.com/cloudiator/visor

We do not have a separate Wiki page for this, since we wanted to have the interface compatible with the main project. Also the Adaptation Engine is configured through the main REST interface of the Cloud Provider Engine (Colosseum subcomponent).

Download

The official version can be downloaded from: https://www.cloudsocket.eu/download

The sole axe-aggregator, which gets deployed on the VMs as well as the basic sensors can be downloaded from github. It features also an own branch for the CloudSocket-specific features, which are not needed for the general purpose of this project: https://github.com/cloudiator/axe-aggregator

The components are released here under the Apache 2.0 license.

Instances

As the adaptation engine is subcomponent of Cloudiator, it is deployed on the same VM at the Openstack cloud of UULM. The respective adaptation actions can be set via the following API calls:

http://134.60.64.155:9000/api/composedMonitor

http://134.60.64.155:9000/api/componentHorizontalOutScalingAction

http://134.60.64.155:9000/api/componentHorizontalInScalingAction

3.3.2.5 SLA Engine

Summary

The SLA Engine represents the component responsible for generating, storing and observing the formal documents describing electronic SLAs between the parties involved in CloudSocket: BPaaS bundle customers, brokers and cloud providers.

The component follows the WS-Agreement (WSAG) specification. This means that the documents representing templates and agreements are valid according to the schema defined in this specification. The specification is extended where needed to cover some specific CloudSocket needs.

This component will expose:

- A graphical user interface for checking the status of agreements,
- A REST API interface allowing programmatic access to the different types of functionalities offered.

The main functionalities are:

- Generation of WS-Agreement templates and agreements.
- Management of SLA related entities: templates, agreements, providers, violations, penalties
- Assessment of SLOs and corresponding penalties when an SLO is violated.
- Notification of detected violations and incurred penalties to interested parties.

The following table indicates the details of the component.

Type of ownership	Extension & Adaptation
Original tool	SLA Framework asset
Planned OS License	Apache License Version 2.0. [7]

Reference community	Atos Research & Innovation (ARI) developers
Lead Partner	ATOS

Table 18 - SLA Engine details.

Comprises

- SLA Core (core functionalities)
- SLA Dashboard (Graphical User Interface)

Depends on

- Monitoring Engine
- Repository Manager Engine

Architecture design

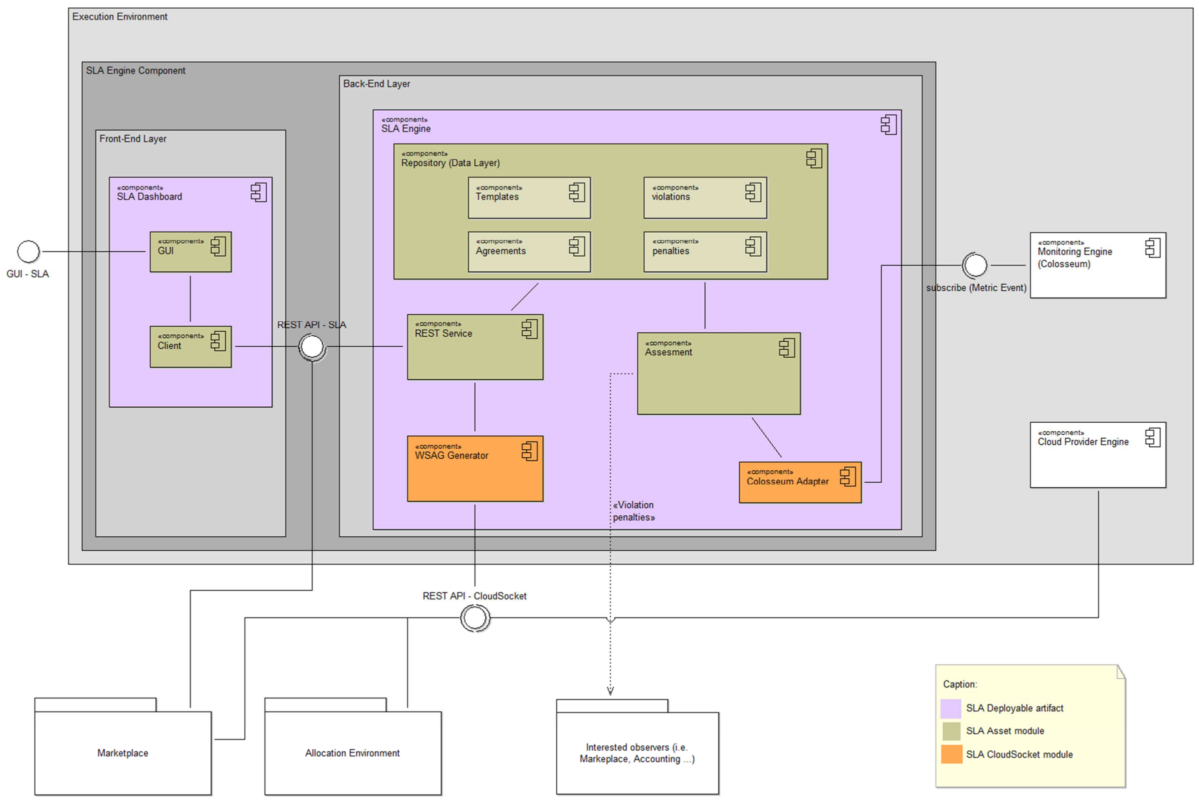


Figure 29 - SLA Engine architecture

The general architecture of this component can be viewed in Figure 29. The SLA Management in CloudSocket is composed of two main components:

- SLA Core. It relies on the Atos SLA Framework asset, with some CloudSocket-related modules. It contains all the basic functionality. The main subcomponents are:
  - Repository. Database of WSAG related entities.

    - Assessment. It is in charge of the evaluation process of the SLA agreements. It notifies raised violations and penalties to interested observers (i.e., an Accounting component).
    - REST Service. It is the REST interface of the SLA Core to the rest of the CloudSocket platform.
    - WSAG Generator. It generates WSAG-compliant templates and agreements according to the requirements of allocation and marketplace, respectively.
    - Colosseum Adapter. It is in charge of: i) subscribing into Colosseum for appropriate metric events; ii) receiving events and translate them into the internal SLA Core representation.
- SLA Dashboard. This component is a frontend application where BPaaS Customers and brokers can check the status of agreements and the penalties that have been applied.

The related components with the SLA Engine in CloudSocket are:

- Monitoring Engine. It reports the metric data and violations of metric conditions to the SLA Core.
- Allocation Environment. The Allocation Environment contacts the SLA Generator to obtain a template that describes the service level of a BPaaS Bundle.
- Marketplace. The Marketplace contacts the SLA Generator to obtain the agreement (the document that acts as a contract between the customer and the broker) when a customer purchases a BPaaS Bundle.
- Repository Manager. Provides service information to be bound to concrete workflows making them executable.
- Cloud Provider Engine. Provides the SLA to be stored and managed by the SLA Engine

Functionalities

Functionality	Description	Completed for release	Integrated (which level)
Generation of WS-Agreement template	Generates the customer-broker SLA template that describes the service levels to be satisfied by a BPaaS bundle.	Yes (1st release)	Ongoing
Generation of WS-Agreement models	Generates the SLA agreement according to the predefined template. An agreement is generated for each purchased BPaaS bundle.	No (2nd release)	
Integration with Monitoring Engine	Subscribes to Monitoring Engine in order to receive data about a BPaaS bundle execution.	Yes (1st release)	Ongoing
Cost model	Incorporates the CloudSocket cost model into the SLA documents. Penalties associated to violations map to this cost model.	No (2nd release)	
SLA composition	Generates and assess hierarchical SLAs where the parent SLA reflects the customer-broker relationship and the children SLAs reflect the “broker-cloud provider” or “user-cloud provider” relationships.	No (2nd release)	

Dashboard adaptation	Integrates the SLA Dashboard with the structure of the SLA agreements in CloudSocket.	Yes (1st release)	Finished
Integration of dashboard with IdM	Integrates the SLA Dashboard with the Identity Management service used in CloudSocket.	No (2nd release)	

Table 19 - Functionalities of the SLA Engine

Table 19 highlights these functionalities while details about the workitems and the respective functionality realisation status can be found at the follow-up Excel file, which has been referenced in Section 2.1. Details about the interactions required to realise them can be found at D4.1 [1] and especially the EE-UC-1-Deployment of BPaaS, EE-UC-2 – Execution of the BPaaS and EE-UC-3 – Monitoring of Agreement Status along with respective UML diagrams [2].

Manuals

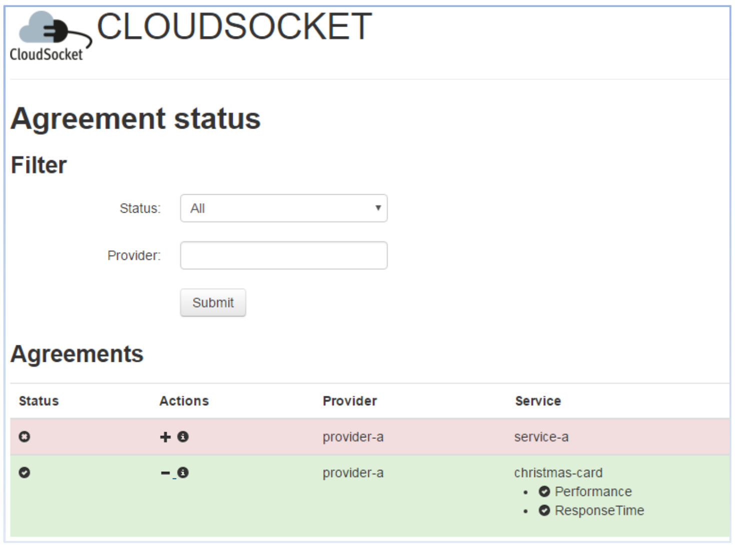
The installation manual, API description, unit test and handbooks are detailed in the alive wiki documentation: https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/SLA+Engine+Component

Download

The source code can be downloaded at https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/sla-framework

Instances

The SLA Core and SLA Dashboard components are deployed in a VM at the Openstack cloud of UULM, and are available at http://134.60.64.232:8080 and http://134.60.64.232:8000, respectively.



Agreement status

Filter

Status: All

Provider:

Agreements

Status	Actions	Provider	Service
		provider-a	service-a
		provider-a	christmas-card • Performance • ResponseTime

Figure 30 - SLA Dashboard

4 PERSPECTIVE: CLOUDSOCKET BROKER

4.1 Introduction

This demonstration showcases [30] how the Broker interacts with the CloudSocket BPaaS Design and Allocation Environments. The Broker aims to design (based on evaluation results) a new business process, defines a corresponding workflow, allocates resources accordingly and publishes a new bundle to the Marketplace. The outcome of these steps is a new BPaaS bundle available for the user to buy and use.

The starting point of this demonstration is a review of key performance indicators (KPIs) by the Broker, deciding on necessary adaptations for existing BPaaS bundles or the development of new ones. The BPaaS designer picks up on these results and creates a new design package and releases it for further processing.

The allocation expert takes over the new design package and performs allocation activities (defining deployment plans and rules, adaptation rules, SLAs as well as marketplace information such as annotations and descriptions) by finally publishing the BPaaS bundle.

To conclude the demonstration, the end-user accesses the marketplace and can now also use the newly designed BPaaS bundle.

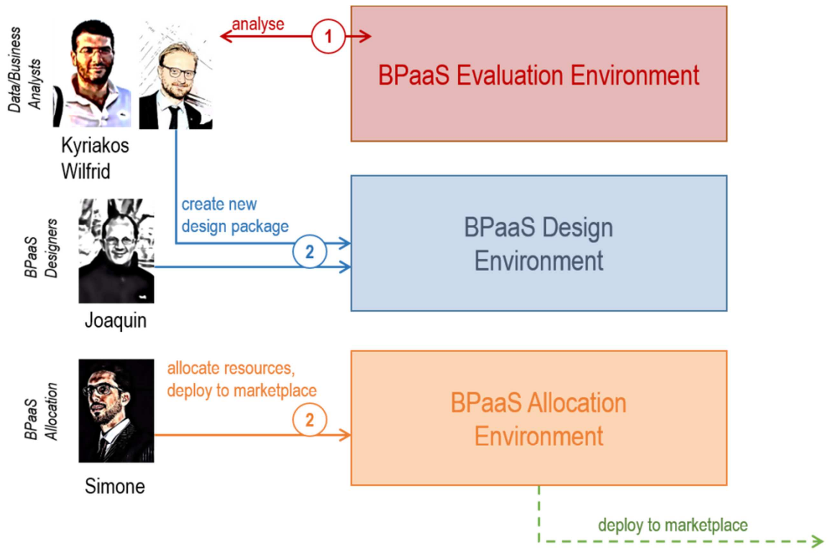


Figure 31 - Broker perspective

4.2 Demonstration

4.2.1 Involved Roles

- BPaaS Customer. Already analysed in Section 3.2.1. This role is mainly involved for demonstration purposes (e.g., to show that a BPaaS bundle has been published and is available for purchase in the Marketplace by involving a Business Engineer sub-role). The actual focus is on the next main role, i.e., the CloudSocket Broker.
- CloudSocket Broker: It is responsible for publishing the BPaaS in the Marketplace. Moreover, it is responsible for managing the discovery, orchestration, deployment and execution of BPaaS services on

the cloud, allowing the customer to use and exploit them. The broker is someone who acts as an intermediary between two or more parties during the negotiation i.e., between the BPaaS customer, the providers of cloud services and the BPaaS broker who offers the respective BPaaS bundle purchased that exploits these cloud services. The CloudSocket Broker is a physical person or an organization whose business is creating BPaaS Bundles and selling them to BPaaS Customers via the Marketplace. Different kinds of roles are defined for the broker mapping to different responsibilities that have to be taken care of in each environment and the respective skills that are needed to fulfil them. These roles are now shortly analysed below:

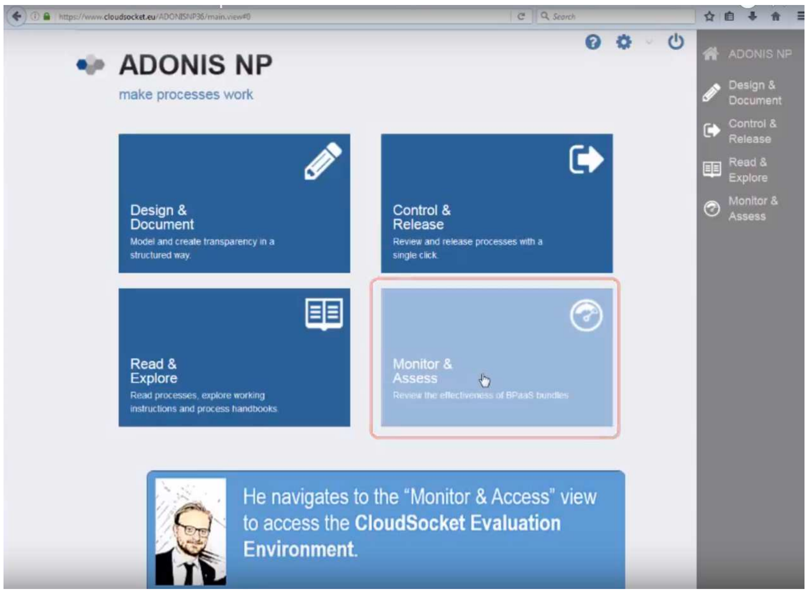
- Consultant: This is a role mapping to an internal or external entity in the CloudSocket Broker organisation. Three different types of consultants are foreseen mapping to respective sub-roles that are analysed below.
  - Business Consultant: This role is associated to respective capabilities or responsibilities to be taken care of at the business level which map to the following actual sub-roles:
    - Business Process Designer: is responsible to define a domain-specific business process model. It could also be able to provide semantic annotations for the business process modelled in order to assist in its alignment. Otherwise, such annotations are to be provided by the Ontology Expert.
    - Evaluation Expert: is responsible for defining KPIs and for monitoring their status. It is also responsible for initiating respective analysis capabilities that are offered by the BPaaS Evaluation Environment whose results can then be used to optimise the BPaaS offered.
    - Sales Consultant: is responsible to define the overall price and all the marketplace relevant information that will be shown in the Marketplace by the end-user once the bundle will be published;
  - Technical Consultant: This role is associated to respective capabilities or responsibilities to be taken care of at the technical level which map to the following actual sub-roles:
    - Workflow Designer: is responsible for the definition of executable workflows that realise the functionality and automate the domain-specific business processes that have been defined by the Business Process Designer. This role, depending on its respective skills and expertise, may also be able to undertake the semantic annotation of the executable workflow in order to assist in its allocation via the BPaaS Allocation Environment. Otherwise, such annotations are to be provided by the Ontology Expert.
    - Allocation Expert: is responsible to create bundles and take all the decisions about the allocation of the software components and atomic services, the service level agreement.
  - Ontology Expert: It is a different kind of consultant with expertise in the management of ontologies and the semantic lifting of models (business process and workflows). This role can be exploited in case the Business Process and Workflow Designers do not have the skills to semantically annotate the models that they generate.

The respective actors mapping to the above roles that are involved in the demonstration mainly from the side of the CloudSocket broker are the following:

- Wilfrid/Kyriakos as the Evaluation Experts of the CloudSocket Broker
- Wilfrid as a Business Process Designer of the CloudSocket Broker
- Joaquin as a Workflow Designer and Ontology Expert of the CloudSocket Broker
- Simone as a Sales Consultant and Allocation Expert for the CloudSocket Broker
- Joaquin acting as a BPaaS customer user undertaking the role of a business engineer to search for a new process on his mobile device

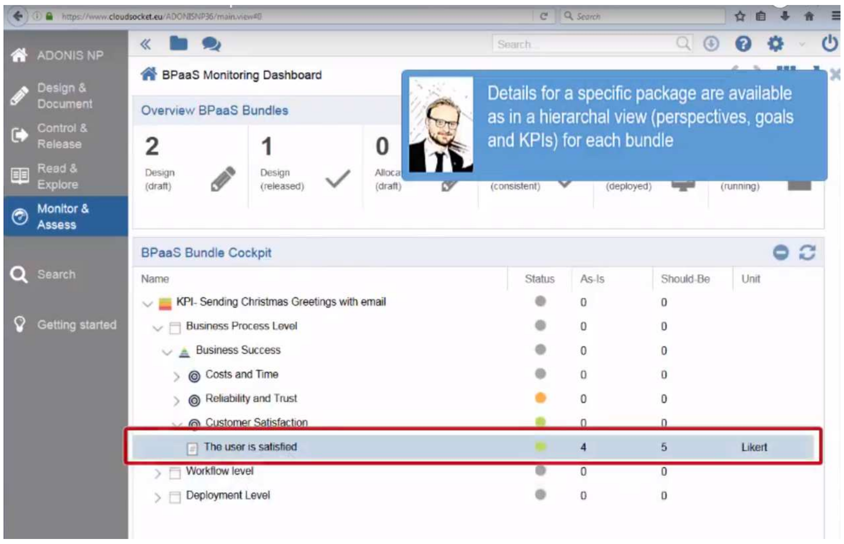
4.2.2 Assessment of indicators for running BPaaS bundles

The Broker hires evaluation experts to review currently running BPaaS bundles and assess the needs and demands based on their usage. Using the Evaluation Environment (Figure 32), the data retrieved from the monitoring interface is visually aggregated as KPIs, goals and perspectives (Figure 33).


ADONIS NP
make processes work
Design & Document
Model and create transparency in a structured way
Control & Release
Review and release processes with a single click
Read & Explore
Read processes, explore working instructions and process handbooks
Monitor & Assess
Review the effectiveness of BPaaS bundles
ADONIS NP
Design & Document
Control & Release
Read & Explore
Monitor & Assess

He navigates to the "Monitor & Access" view to access the **CloudSocket Evaluation Environment**.

Figure 32 – Monitor & Assess



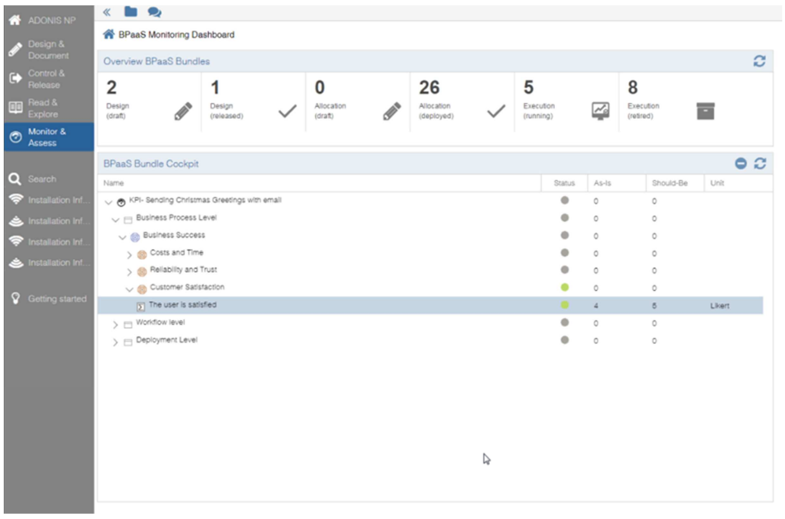
Details for a specific package are available as in a hierarchal view (perspectives, goals and KPIs) for each bundle

Name	Status	As-Is	Should-Be	Unit
KPI- Sending Christmas Greetings with email	●	0	0	
Business Process Level	●	0	0	
Business Success	●	0	0	
Costs and Time	●	0	0	
Reliability and Trust	●	0	0	
Customer Satisfaction	●	0	0	
The user is satisfied	●	4	5	Likert
Workflow level	●	0	0	
Deployment Level	●	0	0	

Figure 33 – Hierarchal view for BPaaS bundle

Based on this assessment, which outlines a quite successful BPaaS bundle, Kyriakos and Wilfrid propose to develop yet another package for not only sending Christmas greetings but modify the broker offering for the upcoming Easter break in order to cover another holiday period major event. They decide to re-use the base structure from Christmas and apply a different context on this design.

The following video shows the assessment of indicators and the decision to create a new context https://youtu.be/NvyL2H31bEQ



BPaaS Monitoring Dashboard

Overview BPaaS Bundles

2	1	0	26	5	8
Design (draft)	Design (released)	Allocation (draft)	Allocation (deployed)	Execution (running)	Execution (retired)

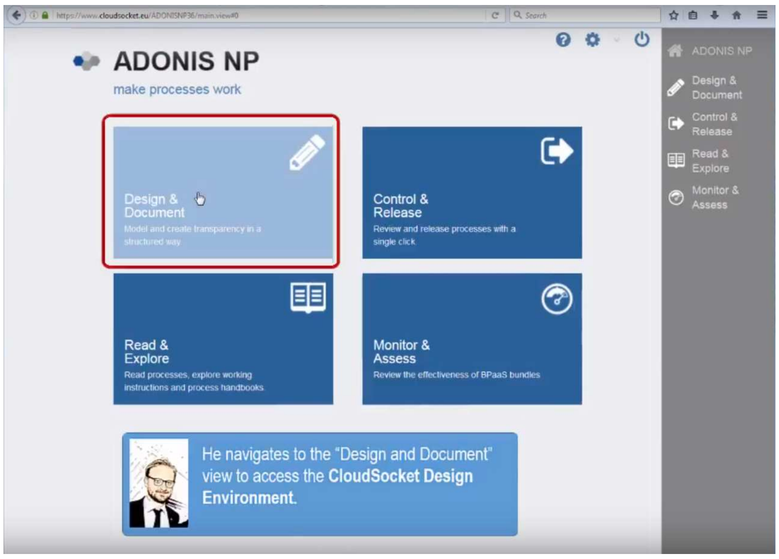
BPaaS Bundle Cockpit

Name	Status	As-Is	Should-Be	Unit
KPI- Sending Christmas Greetings with email	●	0	0	
Business Process Level	●	0	0	
Business Success	●	0	0	
Costs and Time	●	0	0	
Reliability and Trust	●	0	0	
Customer Satisfaction	🟢	0	0	
The user is satisfied	🟢	4	5	Likert
Workflow level	●	0	0	
Deployment Level	●	0	0	

Figure 34 - Analytic dashboard

4.2.3 Creation of new design package

Based on the evaluation results, the creation of a new design package is performed. The design is done using graphical models as an interaction means between business processes and workflow/technical designers Figure 35. In the demonstration, we pick-up the package made available for Christmas and modify it accordingly. This modification is done on one hand by the business process designer, modifying and changing the business process model (Figure 36), and the workflow designer on the other hand who builds an executable process out of the business process model (Figure 37).


https://www.cloudsocket.eu/ADONISNP36/main.view#0
Search
ADONIS NP
make processes work
Design & Document
Model and create transparency in a structured way
Control & Release
Review and release processes with a single click
Read & Explore
Read processes, explore working instructions and process handbooks
Monitor & Assess
Review the effectiveness of BPaaS bundles
ADONIS NP
Design & Document
Control & Release
Read & Explore
Monitor & Assess

He navigates to the "Design and Document" view to access the **CloudSocket Design Environment.**

Figure 35 – Design & Document

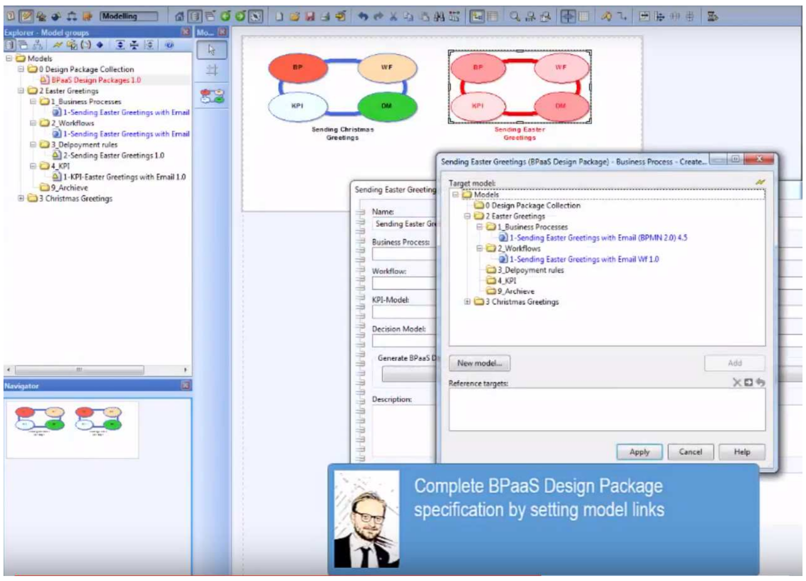

Models
0 Design Package Collection
BPaaS Design Packages 1.0
2 Easter Greetings
1_Business Processes
1-Sending Easter Greetings with Email
2_Workflows
1-Sending Easter Greetings with Email
3_Delpoyment rules
2-Sending Easter Greetings 1.0
4_KPI
1-KPI-Easter Greetings with Email 1.0
9_Archieve
3 Christmas Greetings
BP
WF
KPI
DM
Sending Christmas Greetings
Sending Easter Greetings
Sending Easter Greetings (BPaaS Design Package) - Business Process - Create...
Target model:
1-Sending Easter Greetings with Email (BPMN 2.0) 4.5
1-Sending Easter Greetings with Email WF 1.0
Name:
Business Process:
Workflow:
KPI-Model:
Decision Model:
Description:
New model...
Add
Reference targets:
Apply
Cancel
Help
Complete BPaaS Design Package specification by setting model links

Figure 36 – New Easter Greetings card (business model)

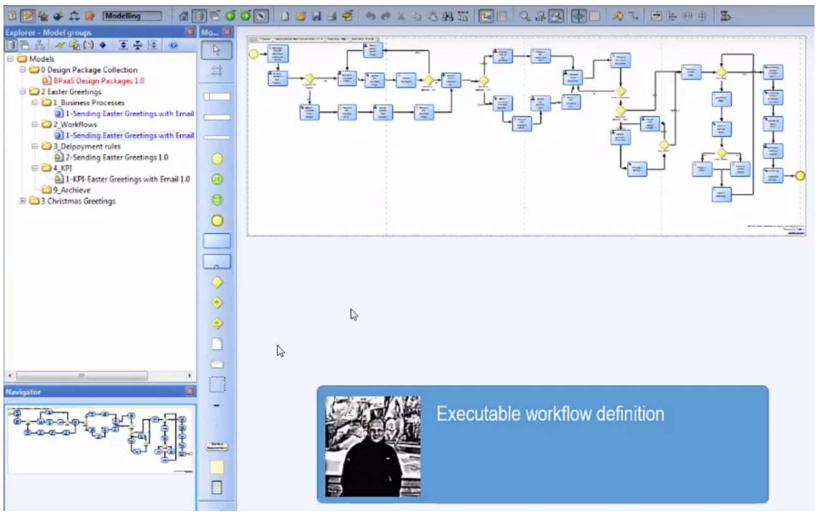
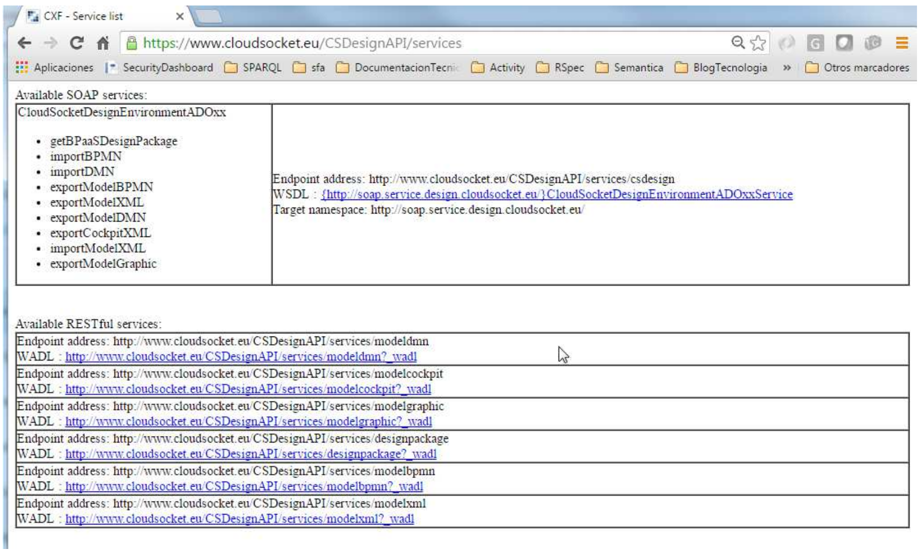

Models
0 Design Package Collection
BPaaS Design Packages 1.0
2 Easter Greetings
1_Business Processes
1-Sending Easter Greetings with Email
2_Workflows
1-Sending Easter Greetings with Email
3_Delpoyment rules
2-Sending Easter Greetings 1.0
4_KPI
1-KPI-Easter Greetings with Email 1.0
9_Archieve
3 Christmas Greetings
Executable workflow definition

Figure 37 – New Easter Greetings card (executable workflow)

The package is designed (Figure 39) as a combination of: (a) a Business Process (+ Service Requirements Specification as a result from WP3) as the business view, (b) a Workflow as the technical view, (c) Rules for deployment (DMN) and sensor definitions and (d) KPIs and goals to measure the success of the package.

In the demonstration, we focus on the design at the business level; the Christmas package is refined and annotated with necessary details to be considered at the technical level. Highlights are added to be considered at workflow and execution level.

As a final step, the package is made available by releasing it in the Modeling Layer of the Evaluation Environment. This makes the package also available via the BPaaS Design Environment API for further use in the Allocation Environment Figure 38).



CXF - Service list

https://www.cloudsocket.eu/CSDesignAPI/services

Available SOAP services:

CloudSocketDesignEnvironmentADOxx	
• getBPaaSDesignPackage • importBPMN • importDMN • exportModelBPMN • exportModelXML • exportModelDMN • exportCockpitXML • importModelXML • exportModelGraphic	Endpoint address: http://www.cloudsocket.eu/CSDesignAPI/services/csdesign WSDL : {http://soap.service.design.cloudsocket.eu/}CloudSocketDesignEnvironmentADOxxService Target namespace: http://soap.service.design.cloudsocket.eu/

Available RESTful services:

Endpoint address: http://www.cloudsocket.eu/CSDesignAPI/services/modeldmn WADL : http://www.cloudsocket.eu/CSDesignAPI/services/modeldmn?_wadl
Endpoint address: http://www.cloudsocket.eu/CSDesignAPI/services/modelcockpit WADL : http://www.cloudsocket.eu/CSDesignAPI/services/modelcockpit?_wadl
Endpoint address: http://www.cloudsocket.eu/CSDesignAPI/services/modelgraphic WADL : http://www.cloudsocket.eu/CSDesignAPI/services/modelgraphic?_wadl
Endpoint address: http://www.cloudsocket.eu/CSDesignAPI/services/designpackage WADL : http://www.cloudsocket.eu/CSDesignAPI/services/designpackage?_wadl
Endpoint address: http://www.cloudsocket.eu/CSDesignAPI/services/modelbpmn WADL : http://www.cloudsocket.eu/CSDesignAPI/services/modelbpmn?_wadl
Endpoint address: http://www.cloudsocket.eu/CSDesignAPI/services/modelxml WADL : http://www.cloudsocket.eu/CSDesignAPI/services/modelxml?_wadl

Figure 38 – BPaaS Design Environment API

The following video shows the design process and interaction to create the new BPaaS Design package https://youtu.be/IrbglZ4a-go

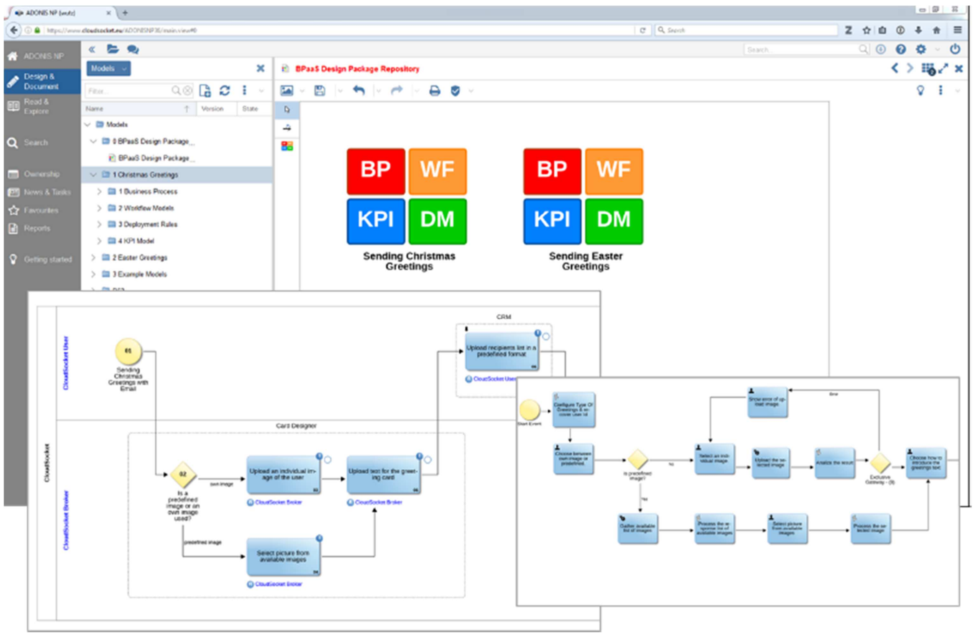
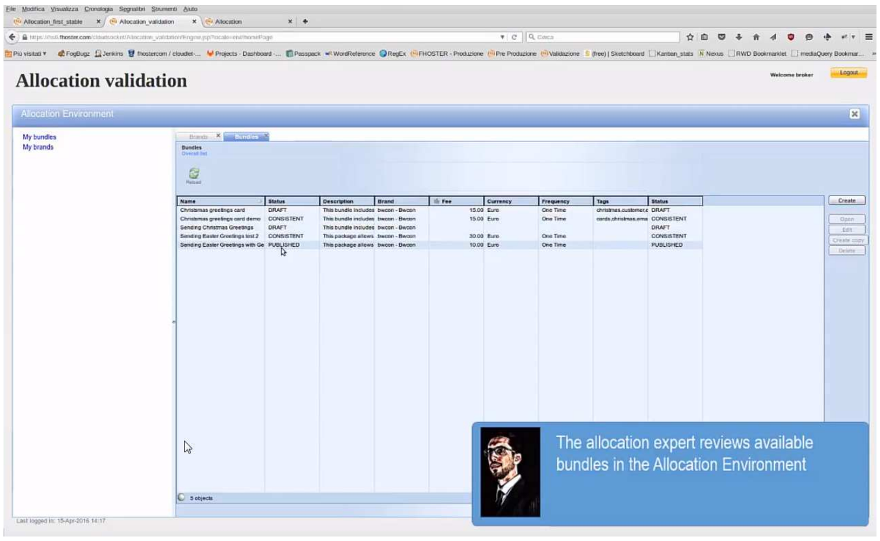


Figure 39 – BpaaS Design Package Repository

4.2.4 Allocation and deployment of the new bundle

Once the CloudSocket Broker completes a new Design Package using the Design Environment, he should be able to create one or more BPaaS bundles using the Design Package as input. The bundle contains both commercial and technical details; for this reason, the Broker can involve a sales consultant and an allocation expert in order to delegate to them the main responsibilities of the bundle creation.

In order to create a new bundle, the allocation expert accesses the Allocation Tool (Figure 40).



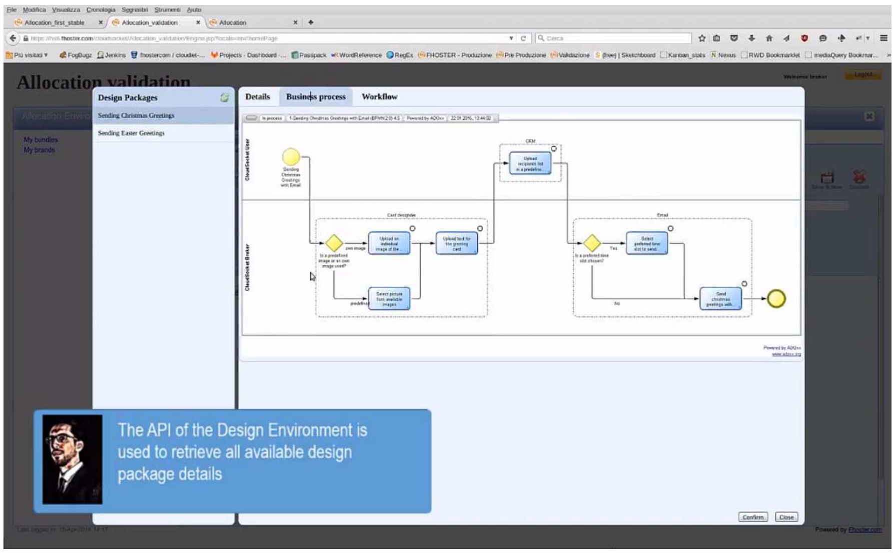
Allocation validation

Name	Status	Description	Brand	Fee	Currency	Frequency	Tags	Status
Christmas greetings card	DRAFT	This bundle includes	bwcon - Bwcon	15.00	Euro	One Time	christmas,customer,c	DRAFT
Christmas greetings card demo	CONSISTENT	This bundle includes	bwcon - Bwcon	15.00	Euro	One Time	cards,christmas,ema	CONSISTENT
Sending Christmas Greetings	DRAFT	This bundle includes	bwcon - Bwcon					DRAFT
Sending Easter Greetings test 2	CONSISTENT	This package allows	bwcon - Bwcon	30.00	Euro	One Time		CONSISTENT
Sending Easter Greetings with Ge	PUBLISHED	This package allows	bwcon - Bwcon	10.00	Euro	One Time		PUBLISHED

The allocation expert reviews available bundles in the Allocation Environment

Figure 40 – Allocation tool

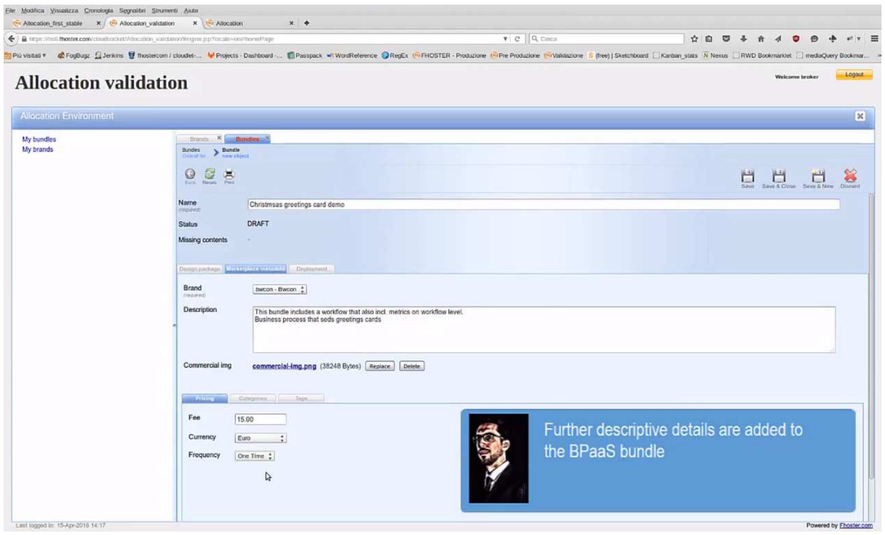
By accessing the "My bundles" menu, he can create a new empty bundle. The first step is to choose a design package for the bundle by browsing all the design packages published and exposed by the Design Environment (Figure 41). The Broker can filter the design packages and then view all the content of a specific package, which includes the description as well as the images of business process and workflow involved. After he identifies the desired package, he can confirm the selection and the Allocation Environment retrieves the whole content of the package using the API provided by the BPaaS Design Environment. This content is stored inside the new bundle and processed in order to retrieve all the information from the workflow like the software components and the concreate atomic services invoked by it.



The API of the Design Environment is used to retrieve all available design package details

Figure 41 – Available BPaaS Packages

Once the empty bundle is filled with the information of the selected design package, the sales consultant can start to define the commercial offer for that bundle. From the commercial point of view, the key information required are the bundle name, a commercial (textual) description, a brand associated to the bundle, a commercial image, a pricing offer, as well as a list of categories and tags. All those information are shown in the Marketplace once the bundle is published on it and the BPaaS Customer can access and view it in order to have a good understanding of what the bundle offers and at which price (Figure 42).


Allocation validation

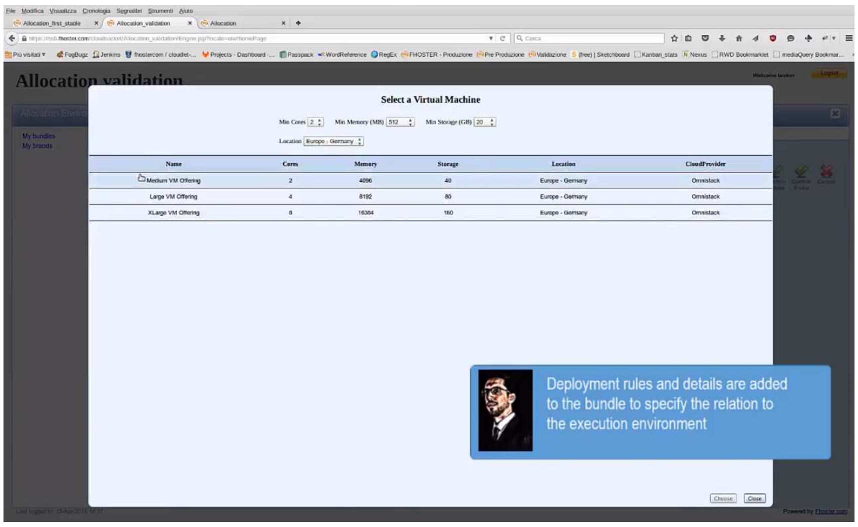
Allocation Environment

Name: Christmsas greetings card demo
Status: DRAFT
Missing contents: -
Brand: bwcon - Bwcon
Description: This bundle includes a workflow that also incl. metrics on workflow level. Business process that seds greetings cards
Commercial img: commercial-img.png (38248 Bytes)
Fee: 15.00
Currency: Euro
Frequency: One Time

Further descriptive details are added to the BPaaS bundle

Figure 42 – Definition of the BPaaS bundle

The allocation expert needs to configure the technical details of the bundle. He should allocate all the Software Components, i.e., select for each of them the proper Virtual Machine Offering which will host it. The Allocation Tool will help in such a selection by filtering the Virtual Machine Offerings space for each software component by considering the component's minimum hardware requirements and its compatible Operating System(s). As such, the Infrastructure as a Service selection/allocation scenario is covered. In order to complete the allocation phase, the allocation expert needs to allocate also all atomic services (currently these maps to the selection of one endpoint from those available). The allocation decisions are translated by the Allocation Tool into a deployment plan defined using the CAMEL language (Figure 43).


Select a Virtual Machine

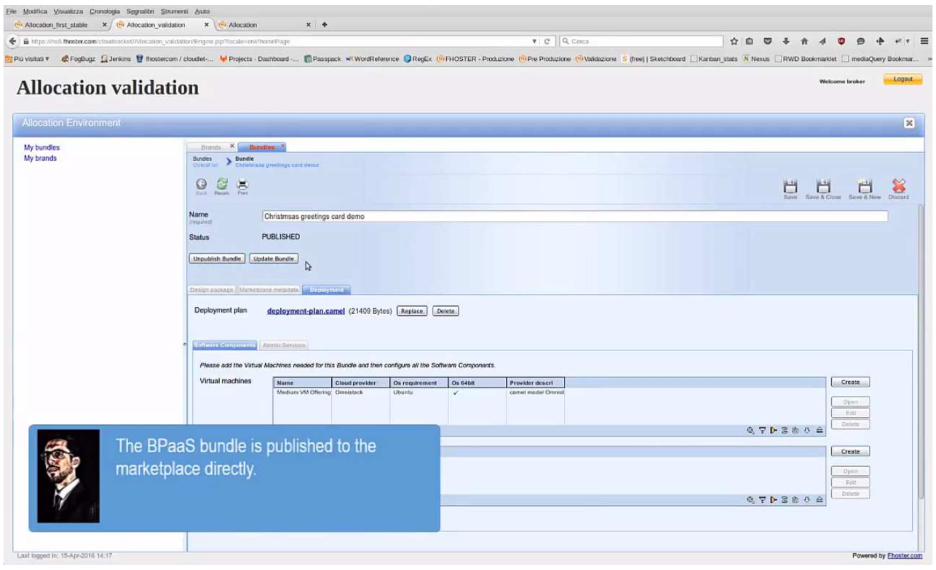
Min Cores: 2 Min Memory (MB): 512 Min Storage (GB): 20
Location: Europe - Germany

Name	Cores	Memory	Storage	Location	CloudProvider
Medium VM Offering	2	4096	40	Europe - Germany	Omnistack
Large VM Offering	4	8192	80	Europe - Germany	Omnistack
XLarge VM Offering	8	16384	160	Europe - Germany	Omnistack

Deployment rules and details are added to the bundle to specify the relation to the execution environment

Figure 43 – Definition of the technical details

The Allocation Tool helps the broker to understand what information is needed for the given bundle in order to be consistent and therefore constitutes a candidate to be published in the marketplace. For this purpose, the tool provides a missing content section which lively contains the list of all the missing information needed by the bundle to be consistent. Indeed, when a new bundle is created, it is in draft status so all possible missing contents are reported; once all contents are provided, then the bundle changes its status from draft to consistent. Only consistent bundles can be published in the marketplace. Once the bundle is published, its status changes from "consistent" to "published" (Figure 44).


Allocation validation

Allocation Environment

Name: Christmas greetings card demo

Status: PUBLISHED

Deployment plan: deployment-plan.camel (21409 Bytes)

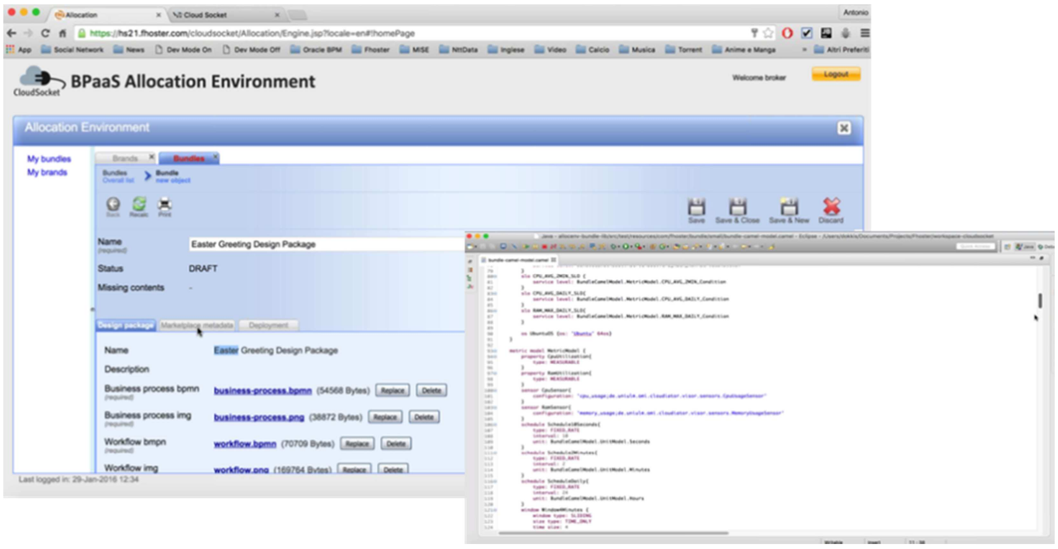
Please add the Virtual Machines needed for this Bundle and then configure all the Software Components.

Name	Cloud provider	Os requirement	Os 64bit	Provider descri
Medium VM Offering	Omnistack	Ubuntu	✓	camel model Omnist

The BPaaS bundle is published to the marketplace directly.

Figure 44 – BPaaS bundle ready to be published

The following video shows the allocation process https://youtu.be/S3vhg8mTPB0


BPaaS Allocation Environment

Allocation Environment

Name: Easter Greeting Design Package

Status: DRAFT

Missing contents: -

Name: Easter Greeting Design Package

Business process bpmn: business-process.bpmn (54568 Bytes)

Business process img: business-process.png (38872 Bytes)

Workflow bmpn: workflow.bpmn (70709 Bytes)

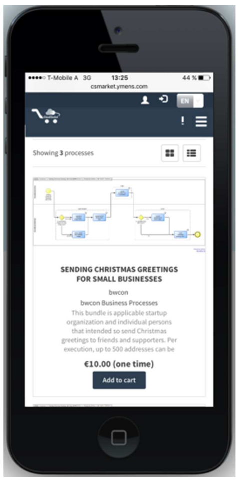
Workflow img: workflow.png (169764 Bytes)

Figure 45 - Allocation environment

4.2.5 Publication of the new bundle

Once the broker finishes the allocation phase for the new bundle, he/she is able to proceed with the publication. This phase comprises the creation of the whole content of the bundle in JSON format and the publication into the Marketplace using the REST API.

Afterwards, the new BPaaS bundle is available for the BPaaS Customer and he can browse (Figure 46) and buy it as it is described in the perspective of the BPaaS Customer Section 3.



SENDING CHRISTMAS GREETINGS FOR SMALL BUSINESSES

bwcon

bwcon Business Processes

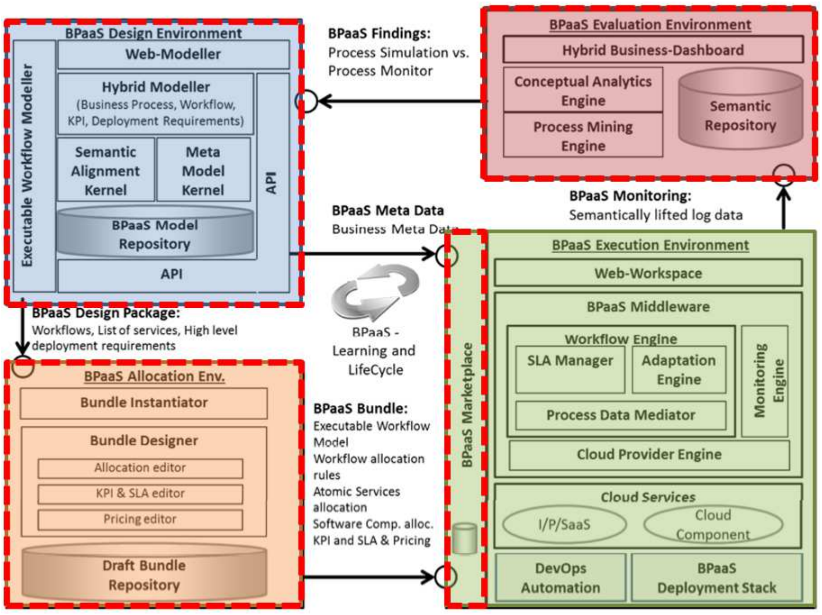
This bundle is applicable startup organization and individual persons that intended so send Christmas greetings to friends and supporters. Per execution, up to 500 addresses can be

€10.00 (one time)

Figure 46 - Marketplace for mobile device

The following video shows the publication of the new BPaaS bundle in the Marketplace and how the BPaaS Customer can review it at a mobile device https://youtu.be/yiSKcg4Esso

4.3 Environments



BPaaS Design Environment
Web-Modeller
Executable Workflow Modeller
Hybrid Modeller (Business Process, Workflow, KPI, Deployment Requirements)
Semantic Alignment Kernel
Meta Model Kernel
BPaaS Model Repository
API
API
BPaaS Findings: Process Simulation vs. Process Monitor
BPaaS Evaluation Environment
Hybrid Business-Dashboard
Conceptual Analytics Engine
Process Mining Engine
Semantic Repository
BPaaS Monitoring: Semantically lifted log data
BPaaS Meta Data
Business Meta Data
BPaaS Execution Environment
Web-Workspace
BPaaS Middleware
Workflow Engine
SLA Manager
Adaptation Engine
Process Data Mediator
Monitoring Engine
Cloud Provider Engine
Cloud Services
I/P/SaaS
Cloud Component
DevOps Automation
BPaaS Deployment Stack
BPaaS Marketplace
BPaaS - Learning and LifeCycle
BPaaS Design Package: Workflows, List of services, High level deployment requirements
BPaaS Allocation Env.
Bundle Instantiator
Bundle Designer
Allocation editor
KPI & SLA editor
Pricing editor
Draft Bundle Repository
BPaaS Bundle: Executable Workflow Model Workflow allocation rules Atomic Services allocation Software Comp. alloc. KPI and SLA & Pricing

Figure 47 - CloudSocket architecture with the environments involved in the Broker perspective outlined with a red rectangle

4.3.1 BPaaS Evaluation Environment

The CloudSocket Evaluation Environment enables conducting different types of analysis over a semantic repository as well as appropriately visualising the analysis results in a sophisticated dashboard. The following types of analysis will be supported: (a) KPI assessment; (b) KPI drill-down; (c) best deployment discovery for BPaaS; (d) detection of event patterns leading to SLO/KPI violations; (e) process mining to discover various types of discrepancies for a BPaaS workflow. This environment covers mainly the BPaaS Broker perspective as it was shown in Section 4.2.2.

In order to support the aforementioned visualisation and analysis functionality, the CloudSocket Environment comprises several components. These components are the following:

- Hybrid Business Dashboard: Enables the visualisation of the analysis information via the use of suitable metaphors. Guides the user in properly performing the different types of analysis
- Hybrid Business Process Management Tool: Integrates the findings of the different types of analysis and enables the orchestration of the communication between the user interface and the respective main analysis components.
- Conceptual Analytics Engine: Provides an API through which the (a)-(d) types of analysis can be performed. It implements the underlying functionality needed.
- Process Mining Engine: Will provide an API via which process mining analysis tasks can be performed over the semantic repository.
- Semantic Repository: It is an underlying semantic repository realised via a Triple Store which enables the posing of sophisticated semantic queries in order to assist in the realisation of the different types of analysis supported.
- Meta-Model Platform: provides access to various information that is required for performing the different types of analysis like KPIs, BPMN (for both BPaaS business processes and workflows) models and DMN specifications. It is the bridge between the Design and Evaluation Environments with respect to the static data already specified for a BPaaS at the design phase.

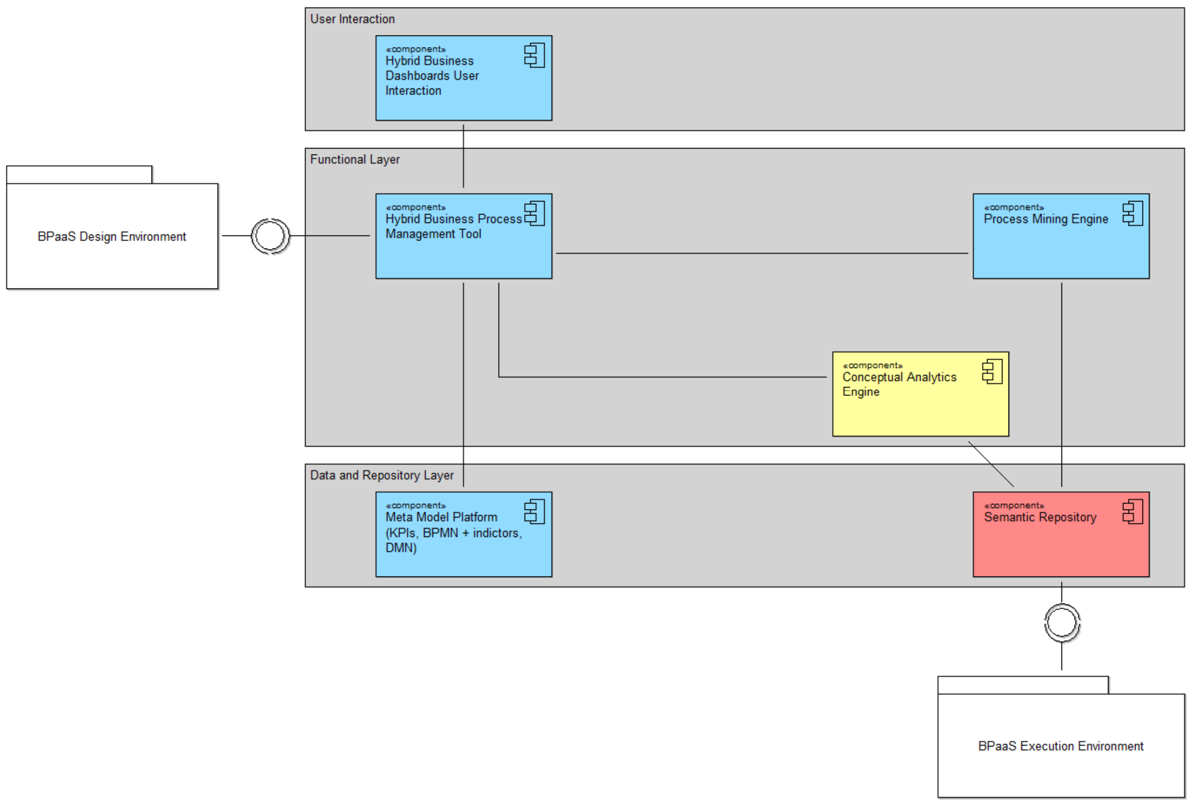

User Interaction
«component» Hybrid Business Dashboards User Interaction
Functional Layer
BPaaS Design Environment
«component» Hybrid Business Process Management Tool
«component» Process Mining Engine
«component» Conceptual Analytics Engine
Data and Repository Layer
«component» Meta Model Platform (KPIs, BPMN + indictors, DMN)
«component» Semantic Repository
BPaaS Execution Environment

Figure 48 - The component diagram of the BPaaS Evaluation Environment

The respective component diagram of the envisioned BPaaS Evaluation Environment is shown in Figure 48. As it can be seen, there are actually three levels covering the visualisation, the analysis and the underlying storage and management of the (semantic) data. In the context of conducting a specific analysis, components in each level interact with each other in order to properly deliver and visualise the respective analysis functionality. More information about how these interactions are performed and what are the respective scenarios covered can be found at the following URL: https://www.cloudsocket.eu/uml/4-EvaluationEnvironment/remotedocu/modeldocu/27012016104756/modelContentHTML in which the corresponding UML diagrams [2] can be viewed. This information was also covered in the D4.1 deliverable [1].

In the following, we describe only those components that are part of this first release of the CloudSocket prototype. These components are the Hybrid Business Dashboard, the Meta Model Platform, the Conceptual Analytics Engine and the Semantic Repository. In this respect, it becomes apparent that the Process Mining Engine has not been implemented yet and it will be incorporated in the next and final release of the CloudSocket prototype. This means that the process mining functionality is not yet supported by the BPaaS Evaluation Environment.

4.3.1.1 Hybrid Business Dashboard

The implementation of the Hybrid Business Dashboard is available as the KPI Monitor described in Section 4.3.2.1. By visualising performance indicators, this component enables the integration between the requirements level in the BPaaS Design Environment and the analysis information/knowledge provided via the analytics engine. This component is composed by the blue rectangles in the component diagram of the BPaaS Evaluation Environment (See Figure 48).

4.3.1.2 Conceptual Analytics Engine

The Conceptual Analytics Engine, for the time being, supports only the evaluation of KPIs by exploiting the semantic information already provided by the Semantic Repository. The rest of the envisioned analysis functionality will be available in the next and final release of the CloudSocket prototype. This component mainly maps to the functionality of a REST service which provides a respective API providing the following three main methods:

(a) one enabling conducting KPI analysis;
(b) one enabling the retrieval of those tenants (BPaaS clients) for which we do have measurements for a respective BPaaS and KPI;
(c) one enabling the posing of arbitrary SPARQL queries which could be used by the broker in order.

The API also provides some additional utility methods that enable the broker to manage the underlying data stored in the Semantic Repository spanning functionality to upload/import, update/modify, export and query the data. Such methods could also be exploited by other components which need to interact with the Semantic Repository in a high rather than a low and detailed level (which would demand the knowledge of how to interact with the underlying Triple Store and would lead to increased implementation effort for realising such interaction).

The communication between the Conceptual Analytics Engine and the Semantic Repository mainly relies on the posing of semantic SPARQL queries over the respective semantic data stored which result in the direct retrieval of the KPI assessment value. In this respect, as the repository is realised via the Virtuoso Triple Store Server, the sesame interface was exploited in order to enable the posing of queries. This interface provides also capabilities mapping to the complete management of the semantic data stored in the Triple Store. These capabilities were adopted in order to realise the aforementioned utility functionality of the Conceptual Analytics Engine API.

Details of the component.

Type of ownership	Creation
Original tool	New component - developed in the context of this project
Planned OS License	Mozilla Public Licence (MPL) 2.0. [28]
Reference community	ADOxx community
Lead Partner	FORTH

Table 20 - Details of the Conceptual Analytics Engine

Depends

- Meta-Model platform (KPI, BPMN, DMN definitions)
- IdM Marketplace (User access control)
- Repository Manager (KPI/SLO metric definition)

Architecture design

This component maps to the yellow rectangle in the component diagram of the BPaaS Evaluation Environment (See Figure 48). It does not involve any sub-components so no internal architecture applies for it. In the next release, as this component maps to multiple functionality, it will be considered to split it into sub-components such that a detailed internal architecture will apply for it.

The component has been partially integrated in the CloudSocket prototype. This is due to the fact that some components are still missing required for the retrieval of KPI information. In this sense, this information is currently hardcoded in this component. However, the component is already well-integrated inside the BPaaS Evaluation Environment as it is exploited by the Hybrid Business Dashboard and can be used to perform KPI assessments by exploiting the content of the Semantic Repository.

As has already been stated, only the KPI evaluation functionality has been fully implemented. The rest of the functionalities intended to be supported by this component have not yet been realised. The following table highlights this while details about the workitems and the respective functionality realisation status can be found at shared excel file, which has been referenced in Section 2.1. Details about the interactions required to realise this functionality can be found in D4.1 [1] and especially the *EvE-UC-1-KPI Analysis and Visualisation* Use Case along with respective UML diagrams [2].

This component could be easily replaced by another developed by an organisation outside the consortium. Its integration will be easy as it is loosely coupled with the rest of the components it connects to. In particular, it retrieves standardised specifications from the Meta-Model platform and exploits them in order to produce respective analysis functionality. For instance, in the case of KPIs, it exploits the KPI specification information in order to produce semantic SPARQL queries to be posed over the Semantic Repository. Furthermore, the replacement of the Conceptual Analytics component would only have to be able to connect correspondingly to the Semantic Repository. As such, the Virtuoso Triple Store Server provides different interfaces to achieve this.

Functionalities

Functionality	Description	Completed for release	Integrated (which level)
KPI evaluation (current & historical values)	The component enables the evaluation of the KPI either with respect to the current moment or with respect to a particular time range (thus covering historical KPI assessment information)	Yes (1st Release)	Still missing integration with Meta-Model Platform.
KPI drill-down	The component enables a drill-down on KPI analysis in order to discover which low-level KPIs are to blame for the violation of high-level ones	No (2nd release)	To be integrated once developed
Best BPaaS deployment discovery	The component will enable the discovery of best deployments for a BPaaS, based on the execution history of the BPaaS and other similar BPaaSs, in order to optimise its allocation.	No (2nd release)	To be integrated once developed
Discovery of event patterns leading to SLO/KPI violations	The component will be able to discover those event patterns that lead to the violation of SLOs/KPIs. Such event patterns could be exploited to develop respective adaptation rules in an semi-automatic or manual manner.	No (2nd release)	To be integrated once developed

Table 21 - Functionalities of the Conceptual Analytics Engine

Manuals

The installation manual, API description, unit test and handbooks for this component are detailed in the alive wiki documentation in the project web site at the following URL: https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Conceptual+Analytics+Engine.

Download

The official version can be downloaded from: https://www.cloudsocket.eu/download. Continuous updates on code are committed at the git repository of UULM: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/evaluation_skb/repository/archive.zip?ref=master.

Instance

The component has been currently deployed in a VM at the Omistack cloud of UULM and is available at the following URL: http://134.60.64.222:8080/rest-test-swagger-0.0.1-SNAPSHOT/. Internally, as being a REST service, a servlet container (tomcat) has also been deployed in the same VM in order to host this REST service.

4.3.1.3 Semantic Repository

The semantic repository is a host of semantic data which enables their proper management and retrieval. It has been realised as a Semantic Triple Store Server mapping to the Virtuoso[1] open-source component. The latter component provides different interfaces through which semantic data can be managed mapping to a Sesame, Jena and native SQL-based interface. Such interfaces constitute the interaction points with other components at the higher levels of the BPaaS CloudSocket Evaluation Environment. In any case, the Virtuoso Triple Store Server offers the following functionality, irrespectively of the interface to be exploited:

- semantic queries issuing via the SPARQL language
- update of semantic data via SPARUL statements
- various functions to import or export semantic data
- abilities to integrate with relational databases via R2RML mapping specifications

Details of the component:

Type of ownership	Usage
Original tool	Virtuoso Triple Store
Planned OS License	GPL v2 [27]
Reference community	Semantic web community
Lead Partner	FORTH

Table 22 - Details of the Semantic Repository

Depends

- IdM Marketplace (User access control)
- Monitoring Engine (execution history)
- Repository Manager (KPI/SLO metric definition)
- Meta-Model platform (KPI, BPMN definitions)

Architecture

This component maps to the red rectangle in Figure 48 (BPaaS Evaluation Environment component diagram). It does not involve any sub-components so no internal architecture applies for it. In the next release, a specific component will be realised on top of it responsible for the retrieval of execution history information from the Execution Environment / Monitoring Engine which will transform and semantic lift this information into the respective semantic content of the Triple Store.

[1] http://virtuoso.openlinksw.com/

The component has been partially integrated in the CloudSocket prototype due to the missing component responsible for the live population of its underlying semantic database/triple store from the monitoring history information. This information is currently manually inserted into the Triple Store. However, the component is already well-integrated inside the BPaaS Evaluation Environment as it is exploited by the Conceptual Analytics Engine in order to pose those SPARQL queries that directly lead to the assessment of KPIs.

This component could be easily replaced by another developed by an organisation outside the consortium. In fact, there are many semantic triple stores, either open-source or proprietary those are currently offered. The sole issue with its integration relies on the respective interface exploited. For instance, the Conceptual Analytics Engine exploits the sesame interface but another Triple Store might not offer this interface. This will require changing slightly the implementation of the Conceptual Analytics Engine in order to exploit the current interfaces offered by the replacement triple store.

Functionalities

Highlights the whole functionality to be offered by this component while details about the workitems and the respective functionality realisation status can be found at the follow-up file, which has been introduced at the Section 2.1. Details about the interactions required to exploit this functionality can be found at D4.1 and actually concern all the use cases specified. The interested reader can also inspect the respective UML diagrams [2] covering these use cases.

Functionality	**Description**	**Completed for release**	**Integrated (which level)**
Management of semantic data	The component enables the management of semantic data stored into its underlying Triple Store by offering different interfaces mapping to the realisation of this management functionality.	Yes (1st release)	
Semantic Repository Population	The component will be coupled with functionality enabling the population of its underlying Triple Store from information produced by the Monitoring Engine which is semantically lifted and stored in this Triple Store.	No (2nd release)	To be integrated once developed

Table 23 - Functionalities of the Semantic Repository

Manuals

Full documentation about this component can be currently found at the Virtuoso's official web page at the following URL: http://docs.openlinksw.com/virtuoso/.

Download

Proprietary versions of this component can be downloaded from the Virtuoso's official web page at the following URL: http://virtuoso.openlinksw.com/download/. The open-source version is available at github: https://github.com/openlink/virtuoso-opensource.

Instance

The component has been currently deployed in the same VM at the Omistack cloud of UULM where the Conceptual Analytics Engine has also been deployed.

4.3.2 BPaaS Design Environment

The Design Environment allows the CloudSocket Broker to construct the BPaaS Design Packages for the Cloud Service to provide. The BPaaS Design Environment provides appropriate conceptual modelling tools for (a) designing domain specific business processes, (b) abstract workflows, (c) additional description and rules for deployment as well as (d) Key Performance Indicators (KPIs).

In order to support aforementioned functionalities, the BPaaS Design Environment is composed by two components called BPaaS Design Tool and Executable Workflow Modeler.

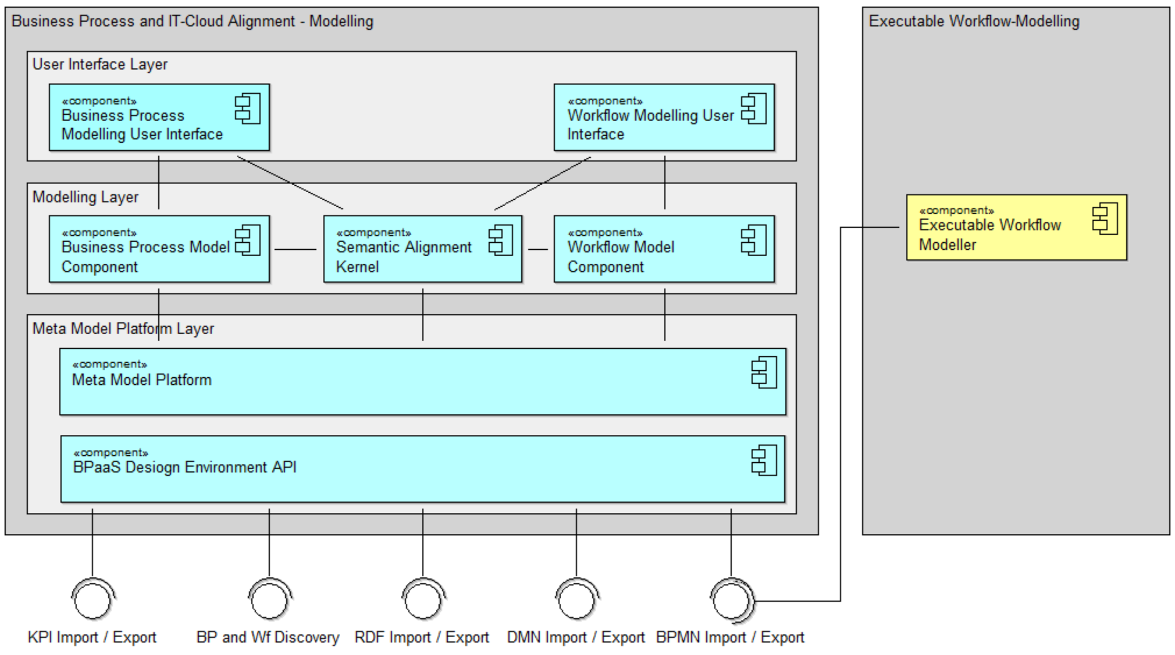

Business Process and IT-Cloud Alignment - Modelling
User Interface Layer
«component» Business Process Modelling User Interface
«component» Workflow Modelling User Interface
Modelling Layer
«component» Business Process Model Component
«component» Semantic Alignment Kernel
«component» Workflow Model Component
Meta Model Platform Layer
«component» Meta Model Platform
«component» BPaaS Desiogn Environment API
Executable Workflow-Modelling
«component» Executable Workflow Modeller
KPI Import / Export
BP and Wf Discovery
RDF Import / Export
DMN Import / Export
BPMN Import / Export

Figure 49 - The component diagram of the BPaaS Design Environment

4.3.2.1 BPaaS Design Tool

The BPaaS design Tool has been created on the base of the CloudSocket meta-model specified in D3.1 [24]. The meta-model define all the aspects required in order to provide the service, that in this specific context are:

- Domain-specific Business Process applying the BPMN 2.0 standard
- Execution Workflow applying the BPMN 2.0 standard but allowing also custom extensions
- Decision Models as DMN 1.1 standard
- The company structure as defined in D3.1 (no standards exist for that)
- Documents involved in the Business Process or Workflow as defined in D3.1 [24] (no standards exist for that)
- Key Performance Indicators KPIs as defined in D3.1 [24] (no standards exist for that)

- Semantic annotations on the domain-specific business processes (and executable workflows) via the RDF standard.

In order to provide those different modelling tools within one environment, a meta modelling platform is used that enables the plug-in of different modelling aspects. For BPaaS, the aforementioned different modelling approaches & tools correspond to the first two BPaaS layers – the i) domain specific business process as well as the ii) executable workflows – as well as their alignment from the business to the technical layer.

Hence the meta modelling platform enables to manage all models in one repository and the interaction between the different layers via so-called model weaving and semantic lifting techniques. Please see more details in Deliverable D3.1 [24].

Details of the component:

Type of ownership	Extension
Original tool	ADOxx
Planned OS License	Closed source. Component available in standalone manner and as service
Reference community	ADOxx
Lead Partner	BOC

Table 24 - Details of the Design Environment

Comprises

- User inteface for the modeling of the bundle: the tool is provided in an experimental space for open development and research using a rich-client UI as well as a productive setting allowing web-based interaction
- Hybrid Business Dashboard: as a result of the CloudSocket Evaluation Environment
- REST API in oder to allow other environments to interact with the BPaaS Design Environment
- Simulation engine to evaluate and analyze the modelled BPaaS services

Architecture design

The general architecture of this component can be viewed by cheking the components coloured in blue, in the Figure 49 above as the Design Environment is composed of the BPaaS Design Tool.

The BPaaS Design Tool is built using the following components:

- A meta modelling platform that provides a BPaaS model repository for all its models, the corresponding management and security infrastructure and a development environment that enables the implementation of modelling components.
- BPaaS modelling components are distinguished by their modelling languages – which are implementations of standards like BPMN, DMN, or RDF – as well as modelling features such as user interaction and model processing. Hence the domain specific business process modeller and the executable workflow modeller are both such a meta modelling component.
- The corresponding (Web-)GUI realizes the user interaction features, to manipulate a model.
- API Interfaces enable the access to the BPaaS Design Environment in particular to the BPaaS model repository, which consists of domain specific business process models, executable workflow models,

additional business requirement models and KPI models. This interaction relies on standard features such as BPMN export / import or on implemented proprietary exchange formats.

- The BPaaS Design Tool API is the component that provides the possibility to programmatically interact with the tool in order to import and export models. Model can be listed and exported in different formats in relation to the specific model type. In particular the API exposes the following features:
  - Get models: Obtain the list of all models in JSON format. This list can be filtered by model type in order to obtain only BPMN or DMN models or other kind of.
  - Image export: give the possibility to obtain an image in JPEG format for the model. The quality and the scale of the image can be specified by parameters
  - BPMN export: give the possibility to export a Business process model or a workflow in standard BPMN 2.0 format
  - BPMN import: give the possibility to import a BPMN 2.0 standard model into the Design Environment
  - XML export: give the possibility to export any kind of model in a generic xml format specifically relevant for the BPaaS Design Environment
  - XML import: give the possibility to import into the Design Environment a model described in its specific XML format
  - DMN export: give the possibility to export a decision tables model in standard DMN format
  - RDF import: give the possibility to import a model defined in standard RFD format into the Design Environment
  - BPaaS Design Package export: enables access to retrieve BPaaS design packages individually.

  These REST API are available at the following URL:
  https://www.cloudsocket.eu/ADONISNP36/rest/cloudsocket/API/

  The following WADL file describe the structure of this API:
  https://www.cloudsocket.eu/ADONISNP36/rest/application.wadl

- The Meta Model Platform ADOxx is used to create the tool. As a result of using this platform it has been possible to create the CloudSocket meta-model in all its aspects and automatically generate a specific modelling environment based on them. Once a meta-model is defined in the platform, it becomes automatically available in the BPaaS Design Environment that providing the possibility to create models for the defined meta-models. This component is also responsible to efficiently store and retrieve of models (of a BPaaS Design Package) defined in the BPaaS Design Environment.

- The Simulation Service enables the statistical evaluation of Business Process or a Workflow in terms of costs, time and unexpected behaviours. Currently the simulation is available only for such type of models but its internal structure give the possibility to easily extend it in order to support other kind of models. The simulation service is integrated into the tool but can be executed also as a separate service through the REST API available at: https://www.adoxx.org/simulation/rest/application.wadl or through the User Interface at: https://www.adoxx.org/simulation/

- The Business Process Modelling User Interface is the entry point for the Business Process Designer in order to model a BPaaS Package. Two versions of this component are provided: an experimentation version and a production version:
  - The experimentation version is a standalone modeller based on the ADOxx toolkit that need to be installed on the designer's pc. This version supports all the meta-models and features defined in the project and it is the only one available for the whole community.
  - The product version is a web app that can be accessed with any browser. In this version all the features of the experimentation version has been replied and some new ones (like the new simulation engine) are introduced.
- The Workflow Modelling User Interface provides similar features with respect to those offered by the business process modelling component (Business Process Modelling User Interface). The actual design and configuration of a workflow is performed in a separate so-called "Executable Workflow Designer" described in Section 3.3.2.1. This enables a higher flexibility and reduces the vendor dependencies, as any workflow designer that is compatible with the Workflow Engine operating in the cloud, can be used. The alignment and CloudSocket relevant parameters can be modelled in the Workflow Modelling Component, and the workflow engine can be designed in the separate tool. The executable workflow is then imported back to enable semantic annotation and discovery but not for the sake of modifying the executable workflow. Hence this component provides user management, model management and model design features for workflows.
- The Semantic Alignment Kernel is a light-weight model editor without own user interface that enables the semantic lifting of business processes and workflows (templates). Hence modelling features for annotations are provided. In addition, full fletched discovery and analysis capabilities are established to enable the discovery of workflow (templates) for business processes and from respective business process requirements.
- The KPI Monitor is a web portal that give the possibility to monitor the current status of all the defined KPIs defined for a BPaaS Design Package. A user friendly interface has been provided in order to explore the KPIs and check if there are inline with the allowed value ranges or not. In the background, the Conceptual Analytics Engine in the BPaaS Evaluation Environment (see Section 4.3.1.2) is exploited in order to derive the actual KPI metric values to be examined.

Functionalities

The Table 25 indicates the covered functionalities and their status:

Create/Edit BpaaS package	This functionality allows a broker to create new BPaaS packages or edit one of his own packages creating/changes models for – Business Processes – Workflows – Decisions – Key Performance Indicators and cause/effect relations – Bundle/Design package overviews	Yes (1st release)	Complete lifecycle
Monitor KPI	This functionality allows a designer to monitor the status of the KPIs defined on every BPaaS Package.	Yes (1st release)	Complete lifecycle

REST API	This functionality gives the possibility to interact with the BPaaS Design Environment programmatically. It is possible to import/export the models in different formats like BPMN, DMN or RDF (according to their respective formats) and to generate images.	Yes (1st release)	Complete lifecycle.
Simulation	This functionality give the possibility to statistically evaluate a Business Process or a Executable Workflow in terms of costs, time and unexpected behaviours	Yes (1st release)	Complete lifecycle.

Table 25 - Details Functionalities of the Design Environment

Manuals

The installation manual, the API description, unit test and handbooks are detailed in the alive wiki documentation at https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Design+Environment+Components
More information can also be found in the ADOxx.org portal: https://www.adoxx.org/live/adoxx-documentation

Instances
The product version of the BPaaS Design Environment is a cloud application that can be accessed using the following credentials:

URL: https://www.cloudsocket.eu/ADONISNP36/
User: ON DEMAND
Password: ON DEMAND

The experimentation version of the BPaaS Design Environment is instead a standalone app based on the ADOxx Toolkit and can be downloaded from here:

URL: https://www.adoxx.org/live/web/cloudsocket-developer-space/downloads

The source code of the BPaaS Design Environment is not available.

4.3.2.2 Executable Workflow Modeler

This component is responsible for the modeling of the executable workflow. Please have a look at the Section 3.3.2.1 for a detailed description. This component maps to the subcomponent of the Workflow Engine called Editor Workflow, which is responsible for the editing of the executable workflows at the design phase, and it is identified in the BPaaS Design Environment at the Figure 49, checking the component coloured in yellow.

4.3.3 BPaaS Allocation Environment

The BPaaS Allocation Environment allows a CloudSocket Broker to select a BPaaS Design Package (previously created via the Design Environment) and create a BPaaS Bundle ready to be published in the Marketplace and deployed in the Execution Environment.

The BPaaS Allocation Environment provides the following high-level features: i) creation of new BPaaS Bundle; ii) search and editing of existing BPaaS bundles; iii) selection of BPaaS Design Package; iv) configuration of marketing metadata; v) workflow services allocation (both atomic services and software components); vi) publication into the marketplace.

In order to support aforementioned functionalities, the BPaaS Allocation Environment is composed by one main component called Allocation Tool.

The following Figure 50 shows the detailed internal architecture of the Allocation Environment architecture.

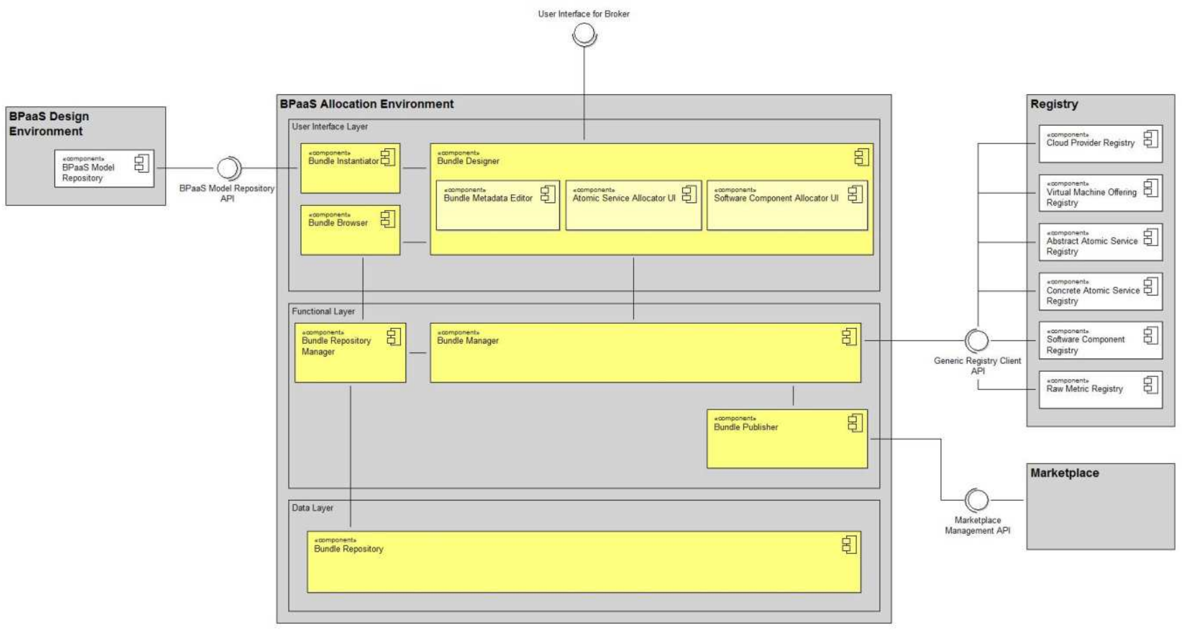

User Interface for Broker
BPaaS Design Environment
«component» BPaaS Model Repository
BPaaS Model Repository API
BPaaS Allocation Environment
User Interface Layer
«component» Bundle Instantiator
«component» Bundle Browser
«component» Bundle Designer
«component» Bundle Metadata Editor
«component» Atomic Service Allocator UI
«component» Software Component Allocator UI
Functional Layer
«component» Bundle Repository Manager
«component» Bundle Manager
«component» Bundle Publisher
Data Layer
«component» Bundle Repository
Registry
«component» Cloud Provider Registry
«component» Virtual Machine Offering Registry
«component» Abstract Atomic Service Registry
«component» Concrete Atomic Service Registry
«component» Software Component Registry
«component» Raw Metric Registry
Generic Registry Client API
Marketplace
Marketplace Management API

Figure 50 - Internal architecture of Allocation Environment

More information about this component and the covered scenarios can be found at the following URL: https://www.cloudsocket.eu/uml/2-AllocationEnvironment/remotedocu/modeldocu/27012016103400/modelContentHTML/ in which corresponding UML diagrams [2] can be viewed. This information was also covered in the D4.1 deliverable [1].

4.3.3.1 Allocation Tool

The Allocation Tool uses the selected BPaaS Design Package as the basic key part of the BPaaS Bundle to be created containing the Business Process Model, the Executable Workflow Model and additional meta-data (such as domain-specific (business-level) KPIs, ontological mappings, etc.). In fact, the Allocation Tool provides a user interface to complete the definition of the BPaaS Bundle by incorporating all additional information required to deploy and execute it.

A BPaaS Bundle binds an Executable Workflow Model with the concrete Atomic Services and Cloud Infrastructures that will be invoked by the workflow, respectively, when executed.

When creating a BPaaS Bundle, the CloudSocket Broker is responsible for defining the overall pricing and SLA for the whole bundle, taking into account the pricing and the SLA of all the resources – e.g. Workflow engine, atomic services and cloud infrastructures - configured in the bundle.

The BPaaS Bundle is a data structure containing all the information required by the Execution Environment (to deploy, execute, monitor and assess a BPaaS) and by the Marketplace (to allow the BPaaS Customer to effectively search the bundles of interest). Such information includes:

- Business process Model
- Workflow Model
- Atomic Service allocation
- Software Component allocation
- KPI model
- Service Level Agreement (SLA)
- Marketing Metadata
- Adaptation Rules

Type of ownership	Creation
Original tool	New component - developed in the context of this project
Planned OS License	No source released. Component available only as service
Reference community	FHOSTER R&D staff
Lead Partner	FHOSTER

Table 26 - Details of the Allocation Tool

Comprises

- Web UI for the allocation of the workflow (Graphical User interface)
- *Bundle Manager to manage the creation of the bundles*
- *Bundle Repository to store the bundles*
- *Bundle Publisher to publish the bundles into the Marketplace*

Depends on

- *BPaaS Design Environment*
- *Marketplace*
- Repository Manager

The general architecture of this component can be viewed in the Figure 50 above as the Allocation Environment is composed only by the Allocation Tool.

The Allocation Tool provides a web application follows a multi-tier architecture and the respective components are distributed across three main layers:

- A user-interface layer, responsible for the interaction with the users;
- A functional layer, implementing the logic of the services invoked;
- A data layer, responsible for saving, loading and managing data in the involved databases.

- The Bundle Instantiator is the UI component allowing a CloudSocket Broker to create a new BPaaS Bundle. Since a BPaaS Bundle is conceptually a commercial and technical “packaging” of an Executable Workflow Model, the creation of a draft BPaaS Bundle necessarily starts with the selection of a workflow model from a Design Package of the BPaaS Design Environment. The tasks performed by the Bundle Instantiator are:

    - Browse the workflow models mapping to Design packages already created in the Design Environment;
    - Instantiate a new BPaaS Bundle selecting one workflow model.

- The Bundle Designer is the main UI component of the Allocation Tool. It provides all the functionality needed by a CloudSocket Broker in order to configure (fill or edit) the sections of a BPaaS Bundle and move it from the initial Draft (incomplete) state to the Consistent (ready to be published) state. The Bundle Designer includes three main sub-components that allow to define allocation for Atomic Services and Software components and to define the bundle metadata

- The Bundle Browser is the UI component allowing a Broker to explore and manage the Bundle Repository. The component shows to the Broker a list of all the Bundles he/she has created along with their status (Draft, Consistent or Published). The tasks performed by the Bundle Browser are the following:
  - Browse all the BPaaS Bundles developed by the authenticated Broker that have been stored into the Bundle Repository
  - Enable the editing of a BPaaS Bundle not already marked as published
  - Duplicate the content of an existing into a new BPaaS bundle

- The Bundle Repository Manager acts as a management interface to the Bundle Repository for all the other components of the Allocation Tool. It provides all the functionality to list and search the content of the Bundle Repository, to load a bundle in memory, to save a bundle in the Repository and to delete a bundle from the Repository.

  The Bundle Repository Manager supports multi-tenancy functionality, where the tenant is the CloudSocket Broker. Indeed, it is his responsibility to let each Broker access only its own bundles as well as to allow only legitimate state transitions and actions to be executed over the bundles of a certain Broker. Tasks performed by the Bundle Repository Manager:

    - Provide a list of descriptors of the BPaaS Bundles in the Bundle Repository;
    - Search for a BPaaS Bundle in the Bundle Repository;
    - Load a BPaaS Bundle from the Bundle Repository into the memory;
    - Save a BPaaS Bundle from the memory into the Bundle Repository;
    - Delete a BPaaS Bundle from the Bundle Repository;
    - Change the state of a bundle.

- The Bundle Manager implements all the business logic required by the Bundle Designer UI and its sub-components. It manages the BPaaS Bundle in memory, making sure that all the sections of the bundle's data structure are properly aligned at any time (i.e., no invalid cross-references and overall data structure consistency) and contain only allowed values.

  The Bundle Manager also detects when a Bundle has been sufficiently configured (i.e., configured with at least the minimal information required to be published) and performs the transition from the Draft to the Consistent state or vice-versa, in case some required information has been removed.

  Tasks performed by the Bundle Manager: this server-side component just manages in memory the BPaaS Bundle currently edited by the CloudSocket Broker. The Bundle Manager interacts with the following components:

    - The registries (Software Component Registry, Cloud Provider Registry). Note: the registries are not accessed directly by the Bundle Designer because the raw lists provided by each registry must be pre-filtered in order to isolate only the registry items which are compatible with the bundle in memory;
    - The Bundle Repository Manager as it is responsible for loading/saving the bundles from/to the Bundle Repository;
    - The Bundle Publisher as it is responsible to interact with the Marketplace management API in order to publish a bundle.

- The Bundle Publisher is responsible for publishing the BPaaS Bundle in the Marketplace. The Bundle Manager forwards to the Bundle Publisher the Bundle currently in memory and delegates it to perform all the interactions with the Marketplace management API required to get the Bundle published. If the publishing operation succeeds, the Bundle Manager changes the state of the Bundle from Consistent to Published and sends it to the Bundle Repository Manager in order to make the state transition persistent. The Bundle Publisher allows also the update of a Bundle already published in the Marketplace with the same flow followed in case of the first publication.
  Tasks performed by the Bundle Publisher: this server-side component is responsible to publish and update a Bundle into the Marketplace.

Functionalities

The Table 27 indicates the covered functionalities and their status

Functionality	Description	Completed for release	Integrated (which level)
Explore existing BpaaS Bundles	This functionality allows a broker to explore and open details of his/her own bundles	Yes (1st release)	Complete lifecycle.
Create/Edit BpaaS Bundle	This functionality allows a broker to create new bundles or edit existing own bundles	Yes (1st release)	Complete lifecycle
Multi-Tenant	The same instance of the allocation environment manages the bundles of various brokers. Each broker can manage only his own bundles	Yes (1st release)	Complete lifecycle
Browse BPaaS Design Packages	This functionality allows the browsing of the BPaaS Design packages offered by the Design environment	Yes (1st release)	Complete lifecycle.
Edit Marketplace metadata	This functionality allows the editing of all the marketplace relevant information for a BPaaS bundle, like title, description, price, categories and tags	Yes (1st release)	Complete lifecycle.
Allocate Software components	This functionality allows the browsing and selection of virtual machines and assign it to each software component (internal service) mapping to a service task in the BPaaS	Yes (1st release)	Complete lifecycle

	workflow		
Allocate Atomic services	This functionality allows the browsing and selection of concrete service endpoints and assign it to each atomic service mapping to the BPaaS workflow service tasks	Yes (1st release)	Complete lifecycle
Automatic creation of deployment plan	Based on allocation configuration, this functionality creates automatically the camel file which specifies, along with other information, the technical deployment plan of the BPaaS bundle	Partially (1st release)	Complete lifecycle. To be integrated with the other features not yet developed, i.e., adaptation rules, service level objectives (SLOs)
Integration with Repository Manager	The environment is currently integrated with software component, cloud provider and atomic service registries of the Repository Manager	Partially (1st release)	Complete lifecycle. To be integrated with raw metrics registry
Adaptation rules for Software components	This functionality will allow the definition of adaptation rules for a software component, in terms of horizontal scaling and/or migration of virtual machine	No (2nd release)	
Adaptation rules for Atomic services	This functionality will allow the definition of adaptation rules for an atomic service in terms of switching the services	No (2nd release)	
Creation of Metric Conditions	This functionality will allow defining KPIs based on raw metrics and/or other KPIs. The KPIs will be used to define SLO conditions and adaptation rules	No (2nd release)	
Definition of SLA	This functionality will allow defining SLOs based on metric conditions (see previous row), which will compose the overall SLA. It would also enable specifying additional information about the SLA, such as the penalties involved in the violation of a specific SLO.	No (2nd release)	
Publish/update BpaaS bundle in Marketplace	This functionality allows the publication or updating of bundles on the Marketplace	Yes (1st release)	Complete lifecycle.

Table 27 - Functionalities of the Allocation Tool

Table 27 highlights these functionalities while details about the workitems and the respective functionality realisation status can be found at the follow-up excel file, which has been referenced in Section 2.1. Details about the interactions required to realise these functionalities can be found at D4.1 [1] and especially the AE-UC-1 Creation of BPaaS Bundle, AE-UC-2 Workflow Allocation, AE-UC-4 Software Component Allocation, AE-UC-6 SLA Model Editing, AE-UC-8 Business Process Metadata Editing and AE-UC-9-BPaaS Bundle Publishing in the Marketplace along with the respective UML diagrams [2].

Manuals

The Allocation Tool is created with Livebase [17], a platform as a service developed by Fhoster. Livebase allows to create web applications starting from a conceptual model, defined with a proprietary model language that extends the UML class diagram. Livebase generates web applications composed by a GWT client,= and a Java business layer on top of a SQL database hosted on Maria DB DBMS [18].
In the context of this project, the Livebase platform has been integrated with an OSGI framework in order to create pluggable web applications. In this way, the applications generated by the platform can be extended with custom logic developed in Java. For instance, the integration of the Allocation Tool with other environments has been achieved by implementing specific plugins.

The installation manual and handbooks are detailed in the alive wiki documentation:
https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Allocation+Environment+Components - User_Manual

Instances

There is available an instance of the Allocation Environment deployed on the Fhoster infrastructure to integrate the first prototype https://hs21.fhoster.com/cloudsocket/Allocation_prototype/Engine.jsp?locale=en#!homePage with other dependent environments: BPaaS Design Environment and Marketplace.

4.3.4 BPaaS Marketplace

For CloudSocket Brokers the Marketplace's role is the main place where BPaaS bundles are offered to the BPaaS Customers. Through the marketplace BPaaS bundles can be bought and automatic provision process is started. The general description of the BPaaS Marketplace is detailed in the Section 3.3.1, and only the Marketplace component will be describe from the point of view of the Broker functionalities.

4.3.4.1 Marketplace

Through Product service API, brokers can manage and publish their bundles. By using Broker dashboard component brokers can monitor the purchases of their bundles by the BPaaS Customers.

For the brokers, The Marketplace will expose the following interfaces:

- A REST API for product management.
- A graphical user interface for registered brokers which provides Dashboard information through the Portal

The main functionalities are:

- Manage bundles through REST API. Check manual for full details https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Marketplace+Environment+Components .
- Manage various broker related information through User portal component.

Comprise

- Shop – main module that manage shopping experience
- Portal – module that manage non-shop related interactions of the users (purchase history, account details, access provisioned items)

- Service API – provides data for Web UI and core workflow functionality (Executions Environment – provision endpoint)
- Identity provider – ensures M2M and U2M authentication and authorization flows

Depends on

- Allocation Environment – to provide BPaaSs to be listed in marketplace
- Execution Environment – to provision in cloud BPaaSs purchased by the user

Check Section 3.3.1.1-Marketplace for Architectural information and component responsible.

Functionalities

The Table 28 indicates the covered functionalities and their status:

Functionality	**Description**	**Completed for release**	**Integrated (which level)**
Shop	The module ensure BPaaS listings into shop environment and allows registered users to checkout items	Yes (1st release)	Complete lifecycle.
Service APIs	The module ensure API connectivity of the Marketplace in the CloudSocket value stream (Product API – used by Allocation Environment; Provisioning API and Order Management – used by Execution Environment)	Yes (1st release)	Complete lifecycle
Portal	The module provides broker dashboard	No (2nd release)	
Identity provider	This module provides authentication and authorization services for the Marketplace as well as for entire CloudSocket value chain	Yes (1st release)	Complete M2M with Allocation Environment Complete M2M with Execution Environment In progress U2M with Execution Environment

Table 28 - Functionalities of the Marketplace

The manuals and instances are described in the Section 3.3.1.1.

5 CONCLUSIONS

The first CloudSocket prototype has been developed based on the first architecture D4.1 [1] as well as on the necessities of the WP5 CloudSocket Demonstration Brokers while it is aligned with WP8 Exploitation.

The delivery integrates the complete lifecycle of the BPaaS bundle through a simple business process. This has allowed understanding, analysing, developing and integrating the different environments and their components in a collaborative and fluid work environment. Additionally, based on this knowledge and approach, new business processes have also been defined in the scope of WP5 using part of this first prototype, such as the BPaaS Design and Allocation Environments.

This deliverable has covered the analysis of the whole environments and their components by also highlighting updates and going deeper onto the architectural level. The analysis has been structured around the BPaaS demonstration according to the two main perspectives (BPaaS Customer and CloudSocket Broker) through the use of the simple business process; so, first the perspective-based demonstration is explained and then the respective relevant environments are detailed. Besides, the documentation has been produced for each environment in the wiki [19] that is aligned with the content of this deliverable, allowing to the readers to decide the level of detail (small - current deliverable, high - wiki) that they want to obtain depending on their needs.

Moreover, the prototype includes some mature results from the research results in WP3 such as the introduction of cloud orchestration based on the CAMEL standard and the BPaaS monitoring & adaptation approaches. Therefore, the next release will continue to work on this direction, analysing and evaluating the most suitable results of the research prototype(s) to be included at the final release in the form of, e.g., add-ons to existing components. D3.3 explicates the respective research results adoption process and assesses the current maturity of the research prototypes/assets.

This prototype is alive, hence, this first prototype is like a snapshot of the integrated platform, which covers the basic functionalities spanning the complete lifecycle: i) the design of the BPaaS Design Package, ii) the allocation of the resources to create the BPaaS bundle, iii) the publication of the BPaaS bundle into the Marketplace, (iv) the purchase of the bundle and its deployment into the cloud, v) the use, execution, monitoring and adaptation of the BPaaS bundle, and vi) the evaluation of the BPaaS bundle to provide feedback to the broker. The next deliverable will include the current integration activities (such as the transversal authentication and service level agreement support), the new functionalities realised (e.g., thedefinition of DMN decision models and their mapping to CAMEL), the improvement over existing functionalities (e.g., moving current single-layer (i.e., IaaS-based) BPaaS adaptation to cross-layer adaptation) and the incorporation of the research results, following and conforming to the final architecture version D4.5 Final CloudSocket Architecture (M21) of the CloudSocket prototype.

Finally, following this first release, respective dissemination actions will be performed to promote it as indicated in the dissemination roadmap in D7.3.2 [25].

6 REFERENCES

[1] D4.1 - CloudSocket_D4.1_First-CloudSocket-Architecture: https://www.cloudsocket.eu/deliverables
[2] UML diagrams: https://www.cloudsocket.eu/uml/
[3] GNU AGPL v3.0 - http://www.gnu.org/licenses/agpl-3.0.html
[4] MongoDB : https://www.mongodb.com/
[5] REstheart : http://restheart.org/
[6] Docker: https://www.docker.com/
[7] Apache License Version 2.0: http://www.apache.org/licenses/LICENSE-2.0
[8] Vaadin: https://vaadin.com/home
[9] Spring: https://spring.io
[10] SQLAlchemy: http://www.sqlalchemy.org/
[11] Python: https://www.python.org/
[12] Camunda: https://camunda.org/
[13] BonitaSoft: http://www.bonitasoft.com/
[14] Activiti : http://www.activiti.org/
[15] Mybatis : [http://www.mybatis.org/mybatis-3/]
[16] mySql: [https://www.mysql.com/].
[17] Livebase: https://www.fhoster.com/static/index.jsp
[18] MariaDB: https://mariadb.org/
[19] Wiki components: https://www.cloudsocket.eu/group/guest/wiki/-/wiki/Main/Components
[20] GitLab: https://omi-gitlab.e-technik.uni-ulm.de/]
[21] D5.1 Initial CloudSocket Setup Report
[22] CloudSocket roles in the common understanding wiki: https://www.cloudsocket.eu/common-understanding-wiki/-/wiki/Main/CloudSocket+Roles
[23] CloudSocket portal: [https://www.cloudsocket.eu/download]
[24] D3.1 - Modelling Framework for BPaaS: https://www.cloudsocket.eu/deliverables
[25] D7.3.2 - First Year Dissemination Collection: https://www.cloudsocket.eu/deliverables
[26] Maven - https://maven.apache.org/
[27] GPL v2 : https://www.gnu.org/licenses/old-licenses/gpl-2.0.html
[28] Mozilla Public License (MPL) : https://www.mozilla.org/en-US/MPL/2.0/
[29] CloudSocket Channel : https://www.youtube.com/channel/UC8qbCYS49FfG7S8j5AABpaA
[30] BPaaS prototype web presentation : https://www.cloudsocket.eu/web/guest/demonstration-end-user-perspective

ANNEX A: LIST OF ABBREVIATIONS

List of abbreviation used into the document.

- API: Application Programming Interface
- ARI: Atos Research & Innovation
- CRUD: Create, Read, Update and Delete
- BPaaS: Business Process as a Service
- BPMN: Business Process Model and Notation
- GUI: Graphical User Interface
- GPL: General Public License
- GRADY : Grails Rapid Application Development for Ymens
- HTTP: Hypertext Transfer Protocol
- HAL: Hypertext Application Language (HAL)
- IaaS: Infrastructure as a Service
- IT: Information Technology
- JPA: Java Persistence API
- JSON: JavaScript Object Notation
- KPI: Key Performance Indicators
- OSGI: Open Service Gateway Initiative
- QoE: Quality of Experience
- PaaS: Platform as a Service.
- R2RML:
- REST: Representational State Transfer
- SaaS: Software as a Service
- SLA: Service Level Agreement
- SLO: Service Level Objective
- SPARUL
- SPARQL
- TSDB : Time-Series Database (TSDB).
- UI: User interface
- VM: Virtual Machine
- WSAG: WS-Agreement