EXPLANATORY NOTES FOR FINAL BPAAS PROTOTYPE

D4.6 - D4.7 - D4.8

Editor Name	Damiano Falcioni (BOC)
Submission Date	30 June 2017
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 note to introduce the second and final BPaaS prototype, which integrates the complete BPaaS lifecycle through the different BPaaS environments, while it unifies and summarises the prototype deliverables: (i) D4.6 Final BPaaS Design and Evaluation Environment, (ii) D4.7 Final BPaaS Execution Environment and (iii) D4.8 Final BPaaS Allocation Environment. The final prototypes, which are explained in this document and provided via corresponding download and demonstration links, not only improve the initial prototoypes, but also integrate research results where appropriate.

The BPaaS Environments that support the BPaaS lifecyle are: (a) the BPaaS Design Environment, (b) the BPaaS Allocation Environment, (c) the BPaaS Execution Environment, and (d) the BPaaS Evaluation Environment. Additionally, the BPaaS Marketplace is required to enable the BPaaS Customer to find and buy a 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 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 defined in the CloudSocket architecture (D4.5 [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 invoices by email automatically. It includes all the basic concepts allowing the integration of the BPaaS final prototype, fluidly.

As this documents 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], (ii) a summary documentation for the components drawn from their complete specification in the component wiki and (iii) details about component instances offered in the form of a SaaS (e.g. actual endpoint), deployed on the infrastructure from the University of Ulm and the romanien company YMENS Teamnet. Besides, this reporting is complemented by demonstration videos [29] and web presentations [30] involving a subset of all software components of the BPaaS prototype that aim at enhancing the understanding of this prototype.

The document introduces the final 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 involved inspection, such as the wiki component description [19], source repository [20] and component instances offered 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 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. In addition, a more detailed documentation is provided online following the various links in the document.

PROJECT CONTEXT

Workpackage	WP4: BPaaS Implementation
Task	T4.2 – T4.3 – T4.4
Dependencies	This explanatory note is an update of D4.2_4.3_4.4 the description of the initial prototypes and explains D4.6, D4.7 and D4.8. Research results are integrated from WP3, demonstration experience have been gained from WP5.

Contributors and Reviewers

Contributors	Reviewers
Damiano Falcioni, Wilfrid Utz, Robert Woitsch (BOC), Joaquin Iranzo, Roman Sosa (ATOS), Antonio Gallo, Simone Cacciatore (FHOSTER), Radu Davidescu, Alex Ganga, Constantin Valeriu Tuguran, Andreea Popovici (YMENS), Daniel Seybold, Frank Griesinger (UULM), Kyriakos Kritikos (FORTH), Knut Hinkelmann, Emanuele Laurenzi (FHNW)	Kyriakos Kritikos (FORTH) Antonio Gallo (FHOSTER) Andreea Popovici (YMENS) Sosa Gonzalez Roman (ATOS)

Approved by: Joaquin Iranzo (ATOS) as WP4 Leader

Version History

Version	Date	Authors	Sections Affected
0.1	April 20, 2017	Damiano Falcioni	Initial version, TOC
0.2	May 12, 2017	Damiano Falcioni and all the contributors	First integration of all the partner contribution
0.3	May 19, 2017	Damiano Falcioni and all the contributors	All the sections
0.4	May 29, 2017	Damiano Falcioni and all the contributors	All the sections
0.5	June 07, 2017	Damiano Falcioni and all the contributors	All the sections
0.6	June 14, 2017	Damiano Falcioni and all the contributors	Version ready for the internal review
0.7	June 29, 2017	Damiano Falcioni and all the contributors	All the sections

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

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

- D4.6 Final BPaaS Design and Evaluation Environment (M18) - Final prototype implementing the BPaaS design and evaluation/analysis functionality according to requirements raised in D4.5 and during demonstration.
- D4.7 Final BPaaS Execution Environment (M18) - Final prototype implementation of the BPaaS adaptive provisioning functionality according to requirements raised in D4.5 and during demonstration.
- D4.8 Final BPaaS Allocation Environment (M18) - Final prototype for the CloudSocket allocation modelling and BPaaS publishing functionality for the CloudSocket broker.

For the avoidance of doubt, the planned deliverables D4.6, D4.7 and D4.8 are provided separately as prototypes; however the accompanying explanatory notes for each of the prototypes, which is provided as an additional information, is consolidated in this document in order to provide a uniform and complete picture of the prototypes.

The separation of concerns can be identified in the following sections:

- The content of D4.6 can be found mainly inside the analysis of the Broker perspective in Chapter 5, which includes two main and related subsections: Section 5.3.1 for the BPaaS Evaluation Environment and Section 5.3.2 for the BPaaS Design Environment.
- The content of D4.7 is covered mainly by the analysis of the BPaaS Customer perspective in Chapter 4. The main related subsections are: Section **Fehler! Verweisquelle konnte nicht gefunden werden.** for the BPaaS Marketplace and Section 4.3.3 for BPaaS Execution Environment.
- The content of D4.8 is related mainly to the analysis of the Broker perspective in Chapter 5. The main related subsection is Section 5.3.3 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 defined in D4.5 [1], allowing to have a complete lifecycle including all the BPaaS (management) phases. In addition, research results have been integrated into the various BPaaS environments where appropriate.

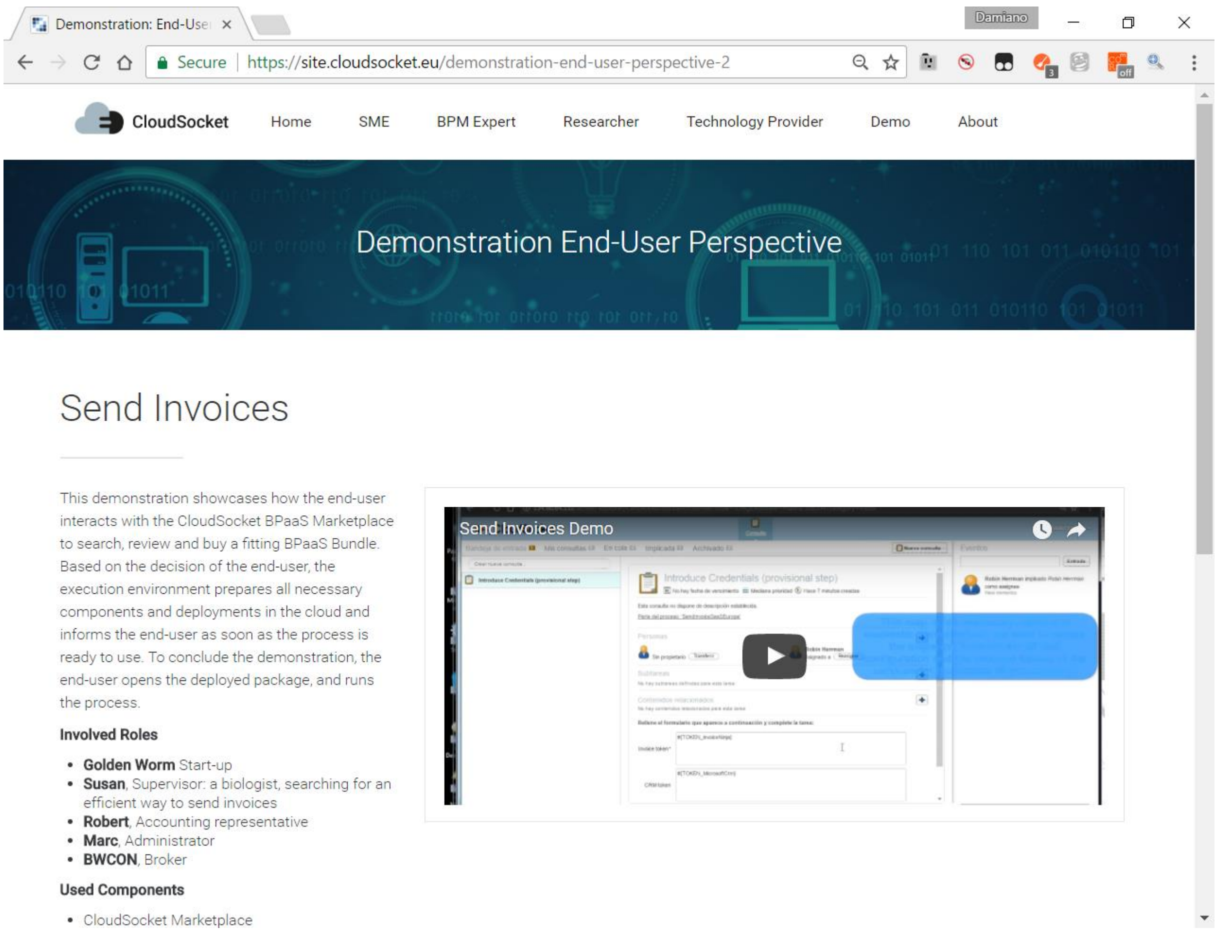
This harmonisation is enabled through the introductional sample of a business process that is used by all the phases in order to produce, publish, adaptively provision and evaluate the respective BPaaS bundle(s). As such, this process constitutes a common example utilised as the thread that runs through the complete lifecycle. This process concerns the automatic sending of Invoices via email and is shortly analysed in Section 2. This process, 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 discovery within the registries, KPI drilldown, the evaluation environment or common understanding to deploy software components and services in the cloud, and the harmonisation of a common understanding with respect to the process definition.

This deliverable is a software prototype, so it includes:

- Access information to the software tools and services developed.

- 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].
- Summary documentation for the components drawn from their complete specification in the component wiki [19].
- Details about component instances offered as SaaS (e.g. actual endpoint), deployed on the UULM and YMENS infrastructure.
- Besides, demonstration videos 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 Final 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 indicated in Figure 1 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 deeper inspection for the interested reader.



Demonstration End-User Perspective

Send Invoices

This demonstration showcases how the end-user interacts with the CloudSocket BPaaS Marketplace to search, review and buy a fitting BPaaS Bundle. Based on the decision of the end-user, the execution environment prepares all necessary components and deployments in the cloud and informs the end-user as soon as the process is ready to use. To conclude the demonstration, the end-user opens the deployed package, and runs the process.

Involved Roles

- **Golden Worm** Start-up
- **Susan**, Supervisor: a biologist, searching for an efficient way to send invoices
- **Robert**, Accounting representative
- **Marc**, Administrator
- **BWCON**, Broker

Used Components

- CloudSocket Marketplace

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:

- **Chapter 2 – CloudSocket Final Prototype**: this is a core section that summarizes all the components, allowing to quickly identify what the readers need in order to later go into deeper details. Hence, this section presents the Final BPaaS prototype in order to (a) understand better the overall integration and platform through the introduction of the running example process for sending Invoices 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 final architecture defined in D4.5 [1]. Finally, an overview of the integrated research results and lessons learned with respect to the different user perspectives are provided.
- **Chapter** Fehler! Verweisquelle konnte nicht gefunden werden. **– Integrated Research Items**: this section introduces for every environment, details on all the integrated research prototypes.
- **Chapter 4 – BPaaS Customer Perspective**: this section covers the different steps of a BPaaS Customer when interacting with the BPaaS prototype. It is structured into:
    - **Section 4.2 – Demonstration**: The involved BPaaS Customer 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 4.3 – 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 documentation through the portal, wiki and gitlab as well as for the provisioning of the environment instances.
- **Chapter 5 – BPaaS 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 structured according:
    - **Section 5.2 – Demonstration**: The same approach, as used in Section 4.2, is followed for this section, but now focusing on Broker-based respective interactions.
    - **Section 5.3 – Environments**. The same approach, as used in Section 4.3, is followed for this section.
- **Chapter 6 – Conclusion:** This section concludes this deliverable and outlines potential future work.

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 environments with their 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 on all these main components.

This same analysis structure of the environment components comprises the following 7 parts supplied in 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:

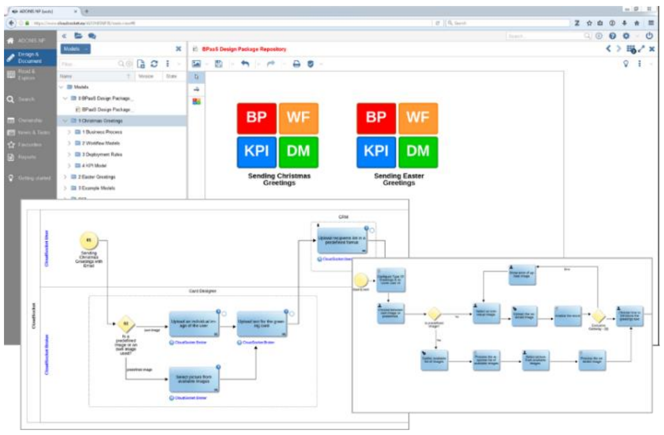
- **Summary:** introduces a general overview, the functionalities, the kind of interfaces as well as the actors and roles involved. It also includes a classification to quickly see the details of the component:

    - *Type of ownership*: Indicates the type of ownership of that 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 when an existing component is re-used, then either options (ii) Extension or (iii) Usage apply to it.
    - *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 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 for which release it has been completed (giving the possibility to distinguish between functionalities present from the first prototype and functionalities introduced in the final prototype) 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-source.
- **Instances:** includes the links of the instances to be used in the integration environment for the prototype.

2 FINAL PROTOTYPE

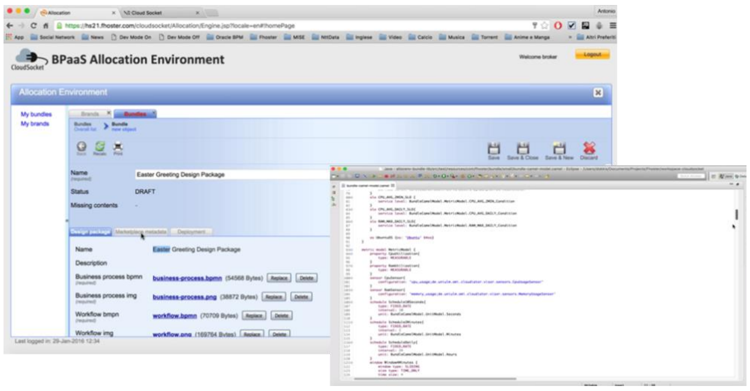
In the following, the software components contained in each environment are introduced using a common factsheet to ease navigation and accessibility. Detailed contextual information for each artefact is available in the documentation provided in Sections 4.3 and 5.3.

2.1 Fact Sheets on Final Prototype

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, a decision model and a set of KPIs/requirements mapping to these 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 a metamodel and provides the possibility to model domain-specific business processes, execution workflows, decision models and key performance indicators.  BP WF KPI DM Sending Christmas Greetings BP WF KPI DM Sending Easter Greetings
Access	SaaS Deployment: https://www.cloudsocket.eu/BPAASDesigner/ Credentials: available on demand. Experimentation Version Download: https://www.adoxx.org/live/web/cloudsocket-developer-space/downloads
License	Closed source
Further Details	Demonstration: Section 5.2.3 Technical Details: Section 5.3.2.1
Manual	https://www.cloudsocket.eu/common-understanding-wiki/-/wiki/Main/Design+Environment+Components

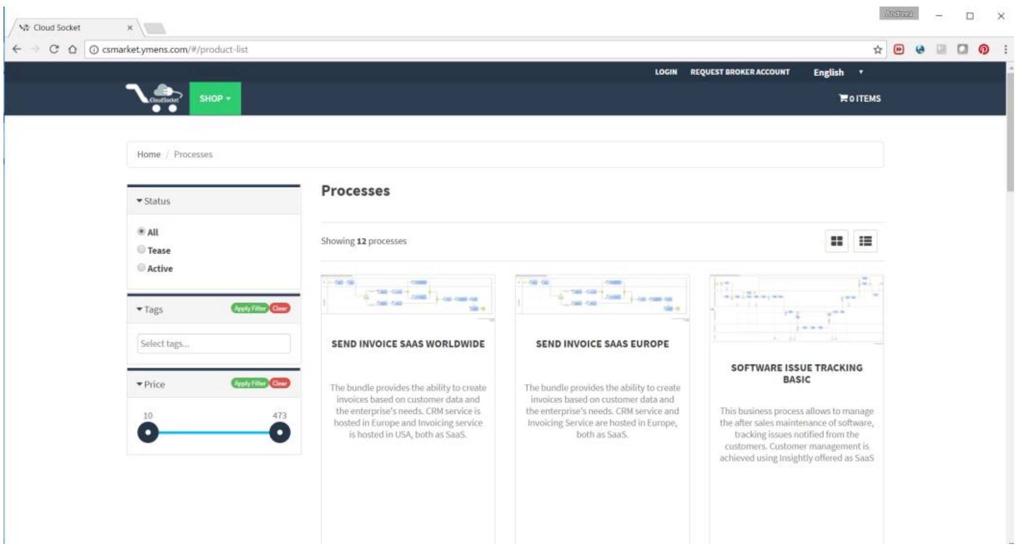
Lead Partner	BOC

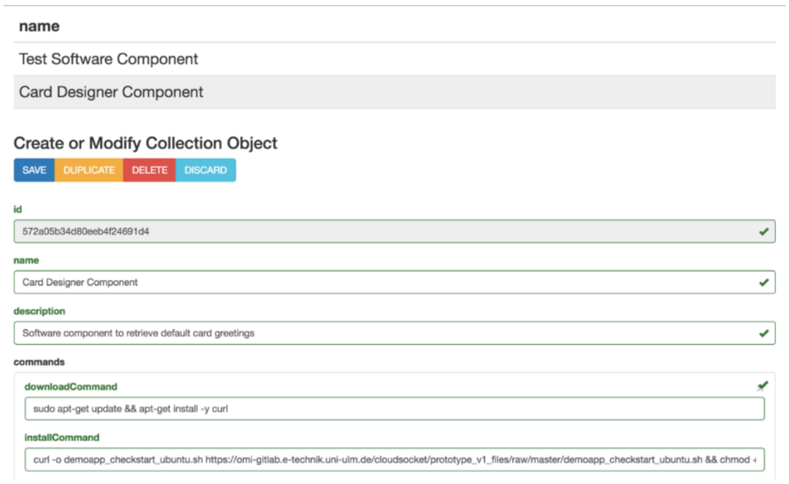
Table 1 – Prototype Components of Design Environment

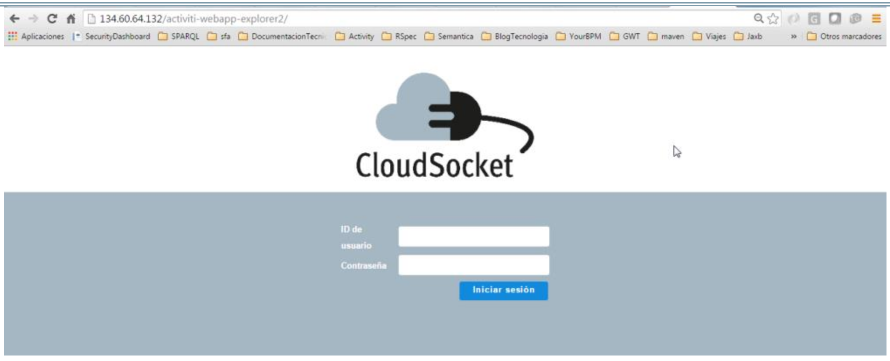
BPaaS Allocation Environment	
The goal of the BPaaS Allocation Environment is to configure allocation directives and rules for an executable workflow model to be deployed and executed in the cloud. An executable workflow model, as produced by the BPaaS Design Environment, does not contain information 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.  https://hs21.fhoster.com/cloudsocket/Allocation/Engine.jsp?locale=en#homePage BPaaS Allocation Environment Welcome broker Allocation Environment 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 Allocation Cloud Socket CloudSocket Logout My bundles My brands Brands Bundles Save Save & Close Save & New Discard Design package Marketplace metadata Deployment Replace Delete
Access	SaaS Deployment: https://hs21.fhoster.com/cloudsocket/Allocation_demo/Engine.jsp (user credentials on demand through YMENS IdM. Logging as a broker profile will enable the feature to publish the bundle in the Marketplace) https://hs21.fhoster.com/cloudsocket/Allocation_demo/ApplicationResource.jsp?application=Allocation (guest account, using the Allocation Tool as guest you can explore all the functionalities but you cannot publish in the Marketplace) Credentials: available on demand using the YMENS IdM
License	Proprietary
Further Details	Demonstration: Section 5.2.4 Technical Details: Section 5.3.3

Manual	https://www.cloudsocket.eu/common-understanding-wiki/-/wiki/Main/Allocation+Environment+Components
Lead Partner	FHOSTER

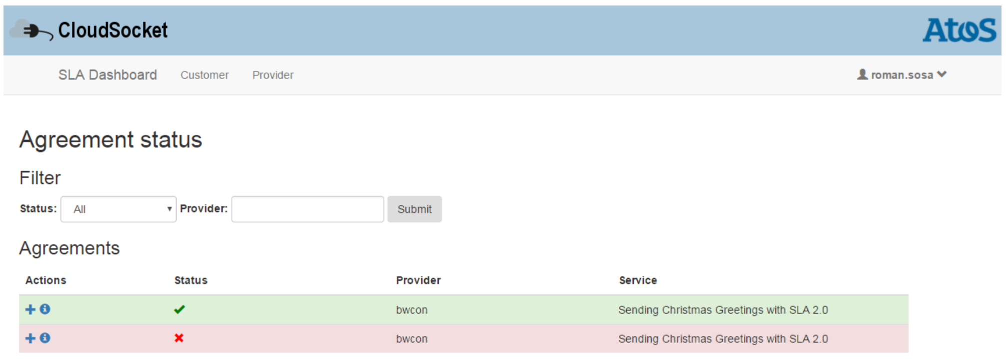
Table 2 - Prototype Components of Allocation Environment

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.  Cloud Socket csmarket.ymens.com/#/product-list LOGIN REQUEST BROKER ACCOUNT English SHOP 0 ITEMS Home / Processes Status All Tease Active Tags Apply Filter Clear Select tags... Price 10 473 Processes Showing **12** processes **SEND INVOICE SAAS WORLDWIDE** The bundle provides the ability to create invoices based on customer data and the enterprise's needs. CRM service is hosted in Europe and Invoicing service is hosted in USA, both as SaaS. **SEND INVOICE SAAS EUROPE** The bundle provides the ability to create invoices based on customer data and the enterprise's needs. CRM service and Invoicing Service are hosted in Europe, both as SaaS. **SOFTWARE ISSUE TRACKING BASIC** This business process allows to manage the after sales maintenance of software, tracking issues notified from the customers. Customer management is achieved using Insightly offered as SaaS
Access	SaaS Deployment: http://csmarket.ymens.com/ Credentials: available on demand.
License	Proprietary
Further Details	Demonstration: 4.2.2 and 0 Technical Details: Section 0
Manual	https://www.cloudsocket.eu/common-understanding-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	Technical Details: Section 0
Manual	https://www.cloudsocket.eu/common-understanding-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-explorer/ (deployed as part of a bundle) Credentials: available on demand using the YMENS IdM 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. Automatic deployment for the continuous integration and the automatic testing: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/workflow-meta/tree/master Credentials: A UULM GitLab account is required in order to access the resources.
License	Apache License Version 2.0 [7]
Further Details	Demonstration: Section 4.2.4 Technical Details: Section 4.3.3.1
Manual	https://www.cloudsocket.eu/common-understanding-wiki/-/wiki/Main/Workflow+Engine+Component
Lead Partner	ATOS
Cloud Provider Engine	This component is responsible for the provisioning, deployment and lifecycle management of all the required components of a BPaaS, including software components and VMs across multiple cloud providers (IaaS/PaaS)
Access	Cloudiator REST API: http://134.60.64.155:9000/ Execution Environment Entrypoint: http://134.60.64.155:9012/job Credentials: Available on demand Download Cloudiator: https://github.com/cloudiator/ Download Execution Environment Entrypoint: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/execution-environment-simple-entrypoint Credentials: A UULM GitLab account is required in order to access the resources.
License	Apache License Version 2.0 [7]

Further Details	Demonstration: Section 0 Technical Details: Section 4.3.3.3
Manual	Cloudiator: http://cloudiator.org/docs/introduction.html Execution Environment Entrypoint: https://www.cloudsocket.eu/common-understanding-wiki/-/wiki/Main/Execution+Environment+Entrypoint
Lead Partner	UULM
Monitoring Engine	This component is responsible to monitor deployed BPaaS bundles and correlate/aggregate monitoring data from different levels, from atomic services or cloud components up to the level of workflows.
Access	WEB UI: http://134.60.64.155:8080/ REST API: http://134.60.64.155:8080/api/v1/datapoints Credentials: Available on demand Download Visor: https://github.com/cloudiator/visor
License	Apache License Version 2.0 [7]
Further Details	https://github.com/cloudiator/visor
Manual	https://github.com/cloudiator/visor/blob/master/documentation/README.md
Lead Partner	FORTH, UULM
Adaptation Engine	This component is responsible for reconfiguring the BPaaS (via, e.g. service substitution, as well as horizontal and vertical scaling) to resolve the problematic situations identified by triggered adaptation rules.
Access	Composed Monitor: http://134.60.64.155:9000/api/composedMonitor Horizontal Out Scaling Action: http://134.60.64.155:9000/api/componentHorizontalOutScalingAction Hoirzontal In Scaling Action: http://134.60.64.155:9000/api/componentHorizontalInScalingAction Credentials: Available on demand Download: https://github.com/cloudiator/axe-aggregator
License	Apache License Version 2.0 [7]
Further Details	https://github.com/cloudiator/visor
Manual	https://github.com/cloudiator/axe-aggregator

Lead Partner	FORTH, UULM
SLA Engine	The SLA Engine represents the component responsible for generating, storing and observing the formal documents describing electronic SLAs between customers and service providers. 

Agreement status

Filter

Status: All Provider: Submit

Agreements

Actions	Status	Provider	Service
+ ℹ	✓	bwcon	Sending Christmas Greetings with SLA 2.0
+ ℹ	✗	bwcon	Sending Christmas Greetings with SLA 2.0

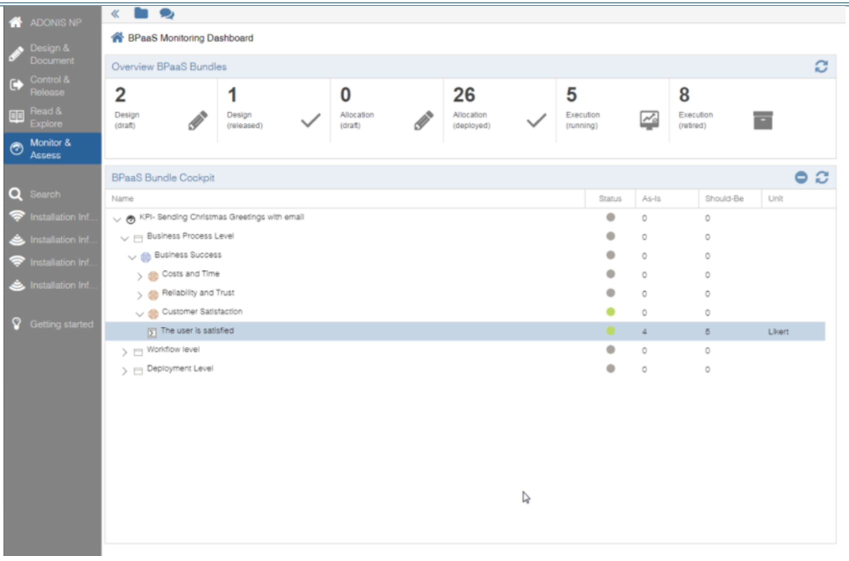
Access	SLA Core instance http://134.60.64.232:8080/api SLA Dashboard instance: http://134.60.64.232:8000 Download: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/sla-framework Credentials: A UULM GitLab account is required in order to access the resources.
License	Apache License Version 2.0 [7]
Further Details	Technical Details: Section 4.3.3.5
Manual	https://www.cloudsocket.eu/common-understanding-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 optimisation suggestions to the broker. This evaluation comes in various forms: (a) the assessment & drill-down of KPIs, (b) the derivation of best deployments for the BPaaS, and (c) the discovery of bottlenecks and problematic business model parts via process mining. Thus, the externally seen functionality of the BPaaS Evaluation Environment maps to initiate 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 KnowledgeBase	A semantic KnowledgeBase organised in the form of a Triple Store on top of which lies a REST API allowing the management of the stored semantic information. Such information enables performing the different types of analysis offered by the BPaaS Evaluation Environment.

Access	http://134.60.64.222:8080/evaluation/ Download: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/skb/repository/archive.zip?ref=master. Credentials: A UULM GitLab account is required in order to access the resources.
License	GPL v2 [27]
Further Details	Technical Details: Section 5.3.1.6
Manual	https://www.cloudsocket.eu/common-understanding-wiki/-/wiki/Main/Semantic+Knowledge+Base
Lead Partner	FORTH
Harvesting Engine	Collects information from the BPaaS Execution Environment and the registries of the Repository Manager, semantically uplifts and links this information according to two ontologies developed in T3.3 and stores this information in the Semantic KB in order to support all types of analysis functionality realised in the BPaaS Evaluation Environment.
Access	Download: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/harvester/repository/archive.zip?ref=master Credentials: A UULM GitLab account is required in order to access the resources.
License	Mozilla Public Licence (MPL) 2.0 [28]
Further Details	Technical Details: Section 5.3.1.2
Manual	https://www.cloudsocket.eu/common-understanding-wiki/-/wiki/Main/Harvesting+Engine
Lead Partner	FORTH
Conceptual Analytics Engine	Provides an API through which KPI assessment and drill-down can be performed on top of the Semantic KnowledgeBase (Semantic KB).
Access	http://134.60.64.222:8080/evaluation/ Credentials: available on demand. Download: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/evaluation/repository/archive.zip?ref=master. Credentials: A UULM GitLab account is required in order to access the resources.
License	Mozilla Public Licence (MPL) 2.0 [28]
Further Details	Demonstration: Section 5.2.2 Technical Details: Section 5.3.1.3

Manual	https://www.cloudsocket.eu/common-understanding-wiki/-/wiki/Main/Conceptual+Analytics+Engine
Lead Partner	FORTH
Deployment Discovery Engine	Provides an API through which a ranked list of relevant deployment(s) for a certain BPaaS can be derived out of the execution history, recorded in the Semantic KB, of this BPaaS as well as of other BPaaS that are similar to it.
Access	http://134.60.64.222:8080/evaluation-dd/ Credentials: available on demand. Download: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/evaluation-dd/repository/archive.zip?ref=master Credentials: A UULM GitLab account is required in order to access the resources.
License	Mozilla Public Licence (MPL) 2.0 [28]
Further Details	Technical Details: Section 5.3.1.4
Manual	https://www.cloudsocket.eu/common-understanding-wiki/-/wiki/Main/Deployment+Discovery+Engine
Lead Partner	FORTH
Process Mining Engine	Provides an API through which certain process mining algorithms can be executed over the workflow execution history recorded in the Semantic KnowledgeBase (Semantic KB).
Access	http://134.60.64.222:8080/evaluation-pm/ Credentials: available on demand. Download: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/evaluation-pm/repository/archive.zip?ref=master Credentials: A UULM GitLab account is required in order to access the resources.
License	Mozilla Public Licence (MPL) 2.0 [28]
Further Details	Technical Details: Section 5.3.1.5
Manual	https://www.cloudsocket.eu/common-understanding-wiki/-/wiki/Main/Process+Mining+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

ADONIS NP
Design & Document
Control & Release
Read & Explore
Monitor & Assess
Search
Installation Inf...
Installation Inf...
Installation Inf...
Installation Inf...
Getting started

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	

Access	SaaS Deployment: https://www.cloudsocket.eu/BPAASDesigner/ Credentials: available on demand.
License	Closed source
Further Details	Demonstration: Section 5.2.1 Technical Details: Section 5.3.1.1
Manual	https://www.cloudsocket.eu/common-understanding-wiki/-/wiki/Main/Design+Environment+Components
Lead Partner	BOC

2.2 Implementation Approach

2.2.1 Project context of Implementation

The different environments are aligned with the corresponding 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 [37] and consolidated in D4.5 [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) integration between environment pairs; (iv) final integration of all components and coverage of 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. 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 CAMEL extension, and the authentication integration.

This document summarizes the final CloudSocket prototype that covers all the phases of a BPaaS defined in D4.5, through respective environments and components. So, the final 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 access details of the component as a SaaS are provided.
- 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 [19].

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 handling of invoice sending. This process is shortly analysed in the following subsection.

2.2.2 BPaaS Sample - Invoice Sending

The project has used the business process for sending Invoices to illustrate the idea of a BPaaS. We use this well-known and simple example to introduce the vision of BPaaS and to further proceed with the integration of the different BPaaS Environments.

First, the business process flow describes the necessary activities of Invoices distribution, such as entering the acquired items, the information of the client or sending the email. Those actions, when executed in the cloud, may cause particular issues, which have been identified by respective business demands.

Such issues or requirements can drive the application of Cloud-specific extensions on the business process that are necessary to configure the technical behaviour of: (a) the workflow – e.g. by introducing service tasks that check the sum of all the invoice items - and (b) the IT-infrastructure – e.g. by the data processing of private data within Europe.

A domain specific business process is transferred into an executable workflow by considering the current IT-cloud offerings and enabling either the automatic alignment between the business and IT level or a manual alignment via the interaction of the responsible roles involved (i.e., business process and workflow designers). For example, in a manual alignment, the broker 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 (interaction) loops, or worse, having misunderstandings that might lead to an incorrect definition of the business process.

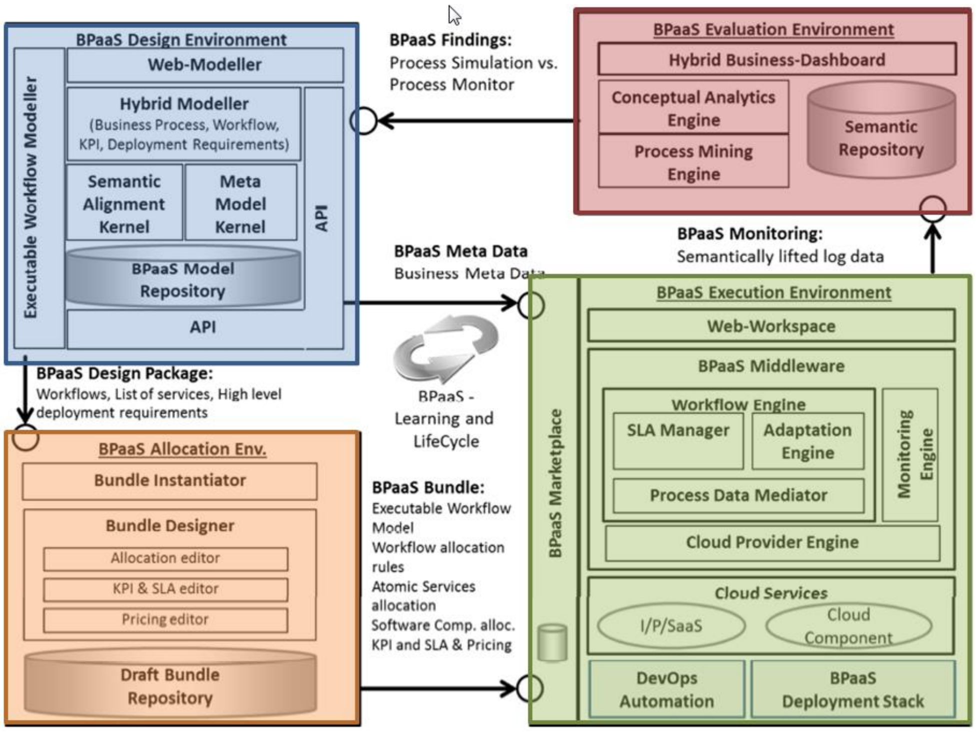
Once an executable workflow is generated, it needs to be allocated accordingly. Two different types of allocations actually take place: (i) each service task in the workflow is mapped to a respective atomic service able to realise its functionality; (ii) for each internal service mapping to a workflow 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) a Service Level Agreement (SLA) template that specifies 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.

2.3 Architecture Overview

The final CloudSocket prototype architecture was defined in D4.5 [1], which led to a technical and detailed common understanding, hence enabling a coordinated 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 in order to enable the different combination possibilities of such environments (e.g. for the creation of a broker's CloudSocket platform) as well as the customisation and enhancement of such environments with alternative implementations.

Figure 2 introduces the four major building blocks in the CloudSocket architecture as well as their relationships and 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 deployment and operation in the cloud. 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.



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 Meta Data
Business Meta Data
BPaaS Monitoring: Semantically lifted log data
BPaaS Design Package: Workflows, List of services, High level deployment requirements
BPaaS - Learning and LifeCycle
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
BPaaS Execution Environment
BPaaS Marketplace
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

Figure 2 - 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 (DMN), workflows and semantic annotations, (ii) the BPaaS Allocation Environment able to link deployable workflows to concrete services and prepare the BPaaS offering, the so called BPaaS bundle, (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 entry point for BPaaS customers that 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 D4.5 [1].

2.4 Prototype Innovation through Research Items

In this section is provided an overview of all the integrated research items for every Environment.

2.4.1 BPaaS Design Environment Prototypes

Semantic lifting: The implemented prototypes address the challenge of Business-IT alignment in the Cloud by identifying cloud services and/or workflows according to the specified user/broker requirements. In this context the integrated prototypes includes a Semantic Lifting extension in the Design Environment and a Questionnaire facility to support the Semantic Lifting. The prototypes were built within the context of the human and machine interpretable paradigm, which (a) allows specifying requirements from a business perspective, and (b) translates and subsequently compare them with cloud service specifications. While the human interpretability is enabled by a modelling environment, the machine interpretability relies on semantic technologies.

2.4.2 BPaaS Allocation Environment Prototypes

CAMEL PaaS/SaaS support: The integration of the PaaS/SaaS CAMEL prototype into the final prototype of CloudSocket enhances the flexibility in offering BPaaS Bundles. By supporting not only IaaS but also PaaS and SaaS services in the CAMEL model, the Allocation Environment can refer to a significantly increased number of services to specify the abstract workflow specifications with concrete cloud services.

2.4.3 BPaaS Execution Environment Prototypes

Engine PaaS Orchestration: Based on the PaaS/SaaS extension of CAMEL, an extension to the Cloud Provider Engine is required, to support the orchestration of PaaS in a similar way to the already supported IaaS, i.e. by abstracting the technical details of different PaaS providers and providing a unified interface for common PaaS providers. Hence, with PaaS orchestration support, the Broker is able offer a greater variety of BPaaS Bundles with respect to available services, SLAs and costs.

2.4.4 BPaaS Evaluation Environment Prototypes

Evaluation Ontology: A novel ontology that enables the capturing of whole BPaaS hierarchies across all abstraction levels and that can assist in conducting various types of BPaaS analysis.

OWL-Q KPI Extension: An extension to the OWL-Q ontology for the non-functional description of services focusing on covering the modelling of KPIs in a very rich and extensible way. This ontology is also exploited for BPaaS analysis purposes while is appropriately linked with the Evaluation Ontology.

Harvester: A thread-based component executed in period time intervals in order to harvest, semantically lift and store in broker-specific partitions in the Semantic Knowledge Base deployment, measurement and registry information collected from the BPaaS Execution Environment and Repository Manager. Such information really supports the various analysis functionalities offered by the components mentioned below.

Conceptual Analytics Engine: A component that enables the evaluation of KPIs for BPaaSes as well as their drill-down for root-cause analysis purposes.

Deployment Discovery Engine: A component that supports the production of a ranked list of deployments for a certain BPaaS which have been derived from the execution history of this BPaaS and of those BPaaSes that are similar to it. The top entries from this list represent best BPaaS deployments which can be considered as suggestions for improving the allocation of a certain BPaaS.

Process Mining Engine: A component that supports the execution of state-of-the-art process mining algorithms over the execution history of a certain BPaaS for BPaaS process model re-creation purposes by also enabling a specific form of semantic process mining. The re-created process models can be compared to the originally design ones to find discrepancies as well as points in the optimisation of the BPaaS design.

2.5 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 are directly involved in CloudSocket and hence 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 final prototype is aligned with this level, where the following stakeholders are considered (see Figure 3):

- *CloudSocket Customer can be seen as the End-End Users* like a 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.
- *CloudSocket Broker can be seen as the End-Users* of the platform. 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 for cloud migration. Moreover, such brokers can realize the whole lifecycle of cloud-based business processes (i.e., BPaaSes) 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 implementations 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 the relevant set of 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, elaborating their teaching courses, building network and research communities, consolidating and conceptualising 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.

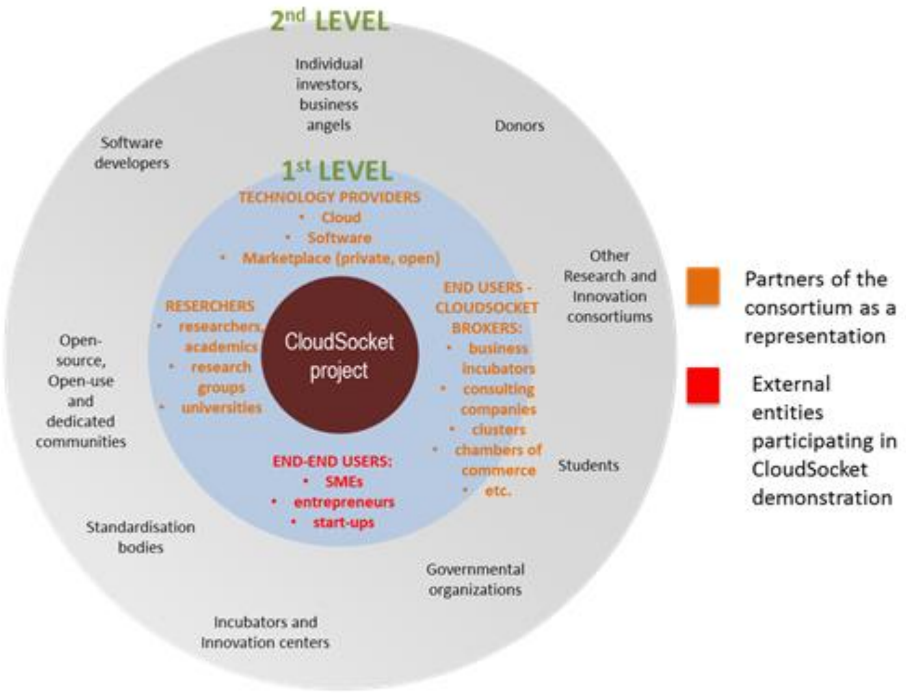

2nd LEVEL
Individual investors, business angels
Donors
Software developers
1st LEVEL
TECHNOLOGY PROVIDERS
• Cloud
• Software
• Marketplace (private, open)
Other Research and Innovation consortiums
END USERS - CLOUDSOCKET BROKERS:
• business incubators
• consulting companies
• clusters
• chambers of commerce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

Figure 3 - CloudSocket ecosystem

The final prototype focuses at this set 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. As the BPaaS Customer has to analyse, purchase and use the different available bundles, there are different kinds of roles associated with it which should exhibit different skills based on the respective responsibilities assigned to them (with respect to the interaction of the customer with the CloudSocket platform):
    - *Business Engineer* with business skills is responsible for finding and purchasing 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 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. 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 i.e., between the BPaaS customer and the providers of cloud services. The CloudSocket Broker is a physical person or an organization whose business is to create BPaaS Bundles and sell 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 aas well as for defining 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.

Therefore, the two main actors and corresponding perspectives pertaining to them 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 chapter 4 and 5 by relying on respective demonstrations of the CloudSocket prototype.

3 SELECTED RESEARCH ITEMS FOR PROTOTYPE

In this chapter is provided a more detailed overview on the research items released from the WP3 and integrated in the final CloudSocket Prototype. Furthermore, references to corresponding WP3 deliverables for additional reading and inspection are also supplied.

3.1 BPaaS Design Environment Prototypes

Identifying appropriate cloud services as well as their orchestrations (i.e., workflows) is an arduous task. This is mainly true in enterprises where IT expertise is scarce or the Business side is poorly aligned with the IT side. We support this task by proposing an environment that enables both the human and machine interpretation. The former allows specifying requirements from a business perspective. The machine interpretability, on the other side, is made possible through semantic technologies that support the matching between the specified requirements and the cloud services / workflows descriptions.

Within this context, we developed two research items – the Semantic Lifting and the Context-Adaptive Questionnaire. These were described in D3.2 and address the following research questions, respectively:

1) How can a modeling environment support consistency between human and machine interpretation of BPaaS models for the Business-to-IT alignment?
2) How can semantic lifting and service discovery be performed more intuitively and smartly?

The following subsections will describe the two prototypes, respectively.

3.1.1 Semantic Lifting

Leading partners: FHNW

Integrated into: Design Environment

Concept and technical details: D3.1 [24]

The Semantic Lifting Prototype comprises the following:

- A modelling environment, where the user / cloud broker can design BPaaS models. Namely, he / she can specify both business process requirements through business language constructs and workflows through technical language constructs (see Figure 4).
- A web service to ensure consistency between the human and machine interpretation of BPaaS models. This paradigm is known as semantic lifting and the BPaaS ontology is the formal representation of the BPaaS models.
- A matching functionality, which takes into account the BPaaS ontology and semantic rules to perform (a) the translation between Business language constructs and IT language constructs (see blue line that connects downtime in minute and availability in Figure 5), and (b) the matching between business process requirements and cloud service specifications / workflow descriptions. As depicted in Figure 5, the business process requirements that match with the related cloud services / workflows are shown with the green color, while the rest in red. This functionality has not been integrated in the Design Environment but is a standalone application.

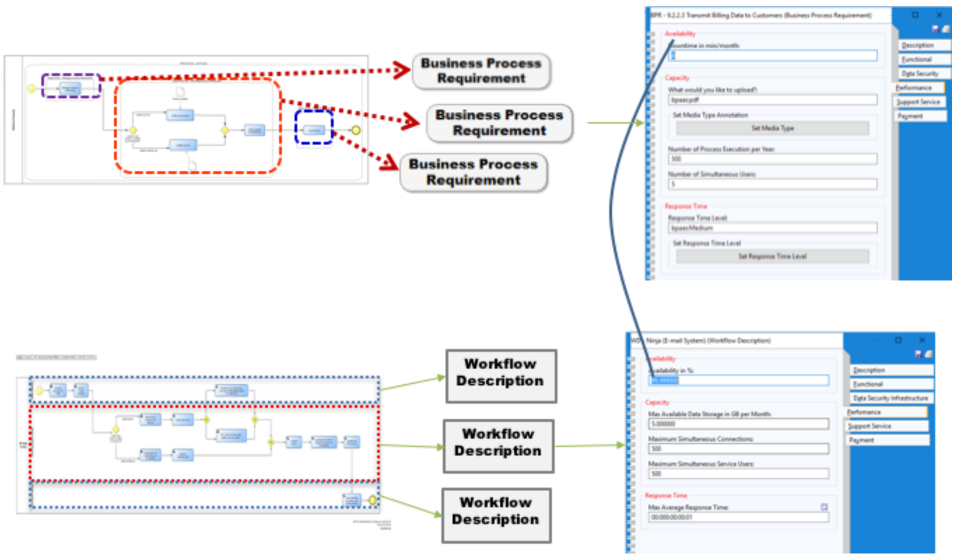
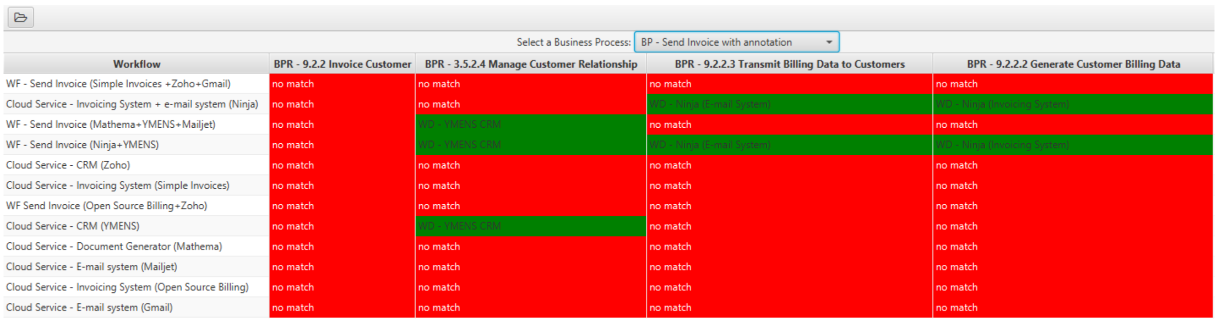

Business Process Requirement
Business Process Requirement
Business Process Requirement
Workflow Description
Workflow Description
Workflow Description

Figure 4: Semantic Annotation of Business Process and Workflows



Select a Business Process: BP - Send Invoice with annotation

Workflow	BPR - 9.2.2 Invoice Customer	BPR - 3.5.2.4 Manage Customer Relationship	BPR - 9.2.2.3 Transmit Billing Data to Customers	BPR - 9.2.2.2 Generate Customer Billing Data
WF - Send Invoice (Simple Invoices +Zoho+Gmail)	no match	no match	no match	no match
Cloud Service - Invoicing System + e-mail system (Ninja)	no match	no match	WD - Ninja (E-mail System)	WD - Ninja (Invoicing System)
WF - Send Invoice (Mathema+YMENS+Mailjet)	no match	WD - YMENS CRM	no match	no match
WF - Send Invoice (Ninja+YMENS)	no match	WD - YMENS CRM	WD - Ninja (E-mail System)	WD - Ninja (Invoicing System)
Cloud Service - CRM (Zoho)	no match	no match	no match	no match
Cloud Service - Invoicing System (Simple Invoices)	no match	no match	no match	no match
WF Send Invoice (Open Source Billing+Zoho)	no match	no match	no match	no match
Cloud Service - CRM (YMENS)	no match	WD - YMENS CRM	no match	no match
Cloud Service - Document Generator (Mathema)	no match	no match	no match	no match
Cloud Service - E-mail system (Mailjet)	no match	no match	no match	no match
Cloud Service - Invoicing System (Open Source Billing)	no match	no match	no match	no match
Cloud Service - E-mail system (Gmail)	no match	no match	no match	no match

Figure 5: Matching Business Process Requirements and Workflow Descriptions

The Semantic Alignment Kernel is a component for annotating modelling elements with the BPaaS Ontology. This was implemented in the ADOxx Modelling Toolkit and ensures consistency between the human and machine interpretation paradigms. In addition, a matching user interface is available, that shows in a matrix which business process requirements match with which cloud service specifications (Figure 6).

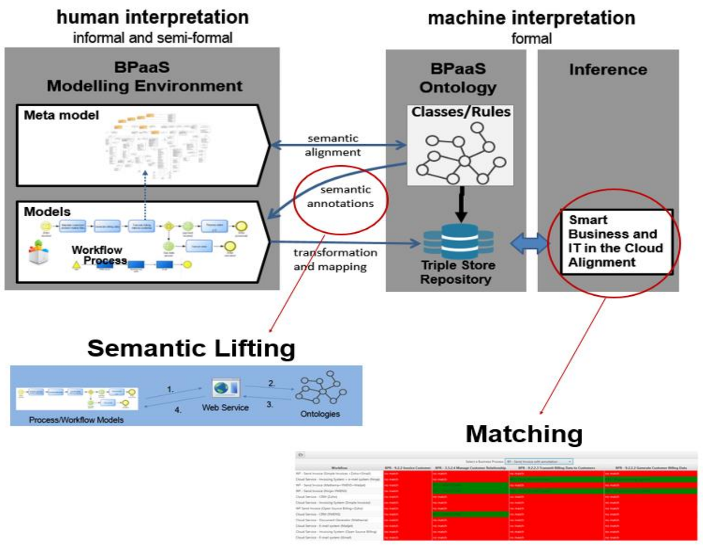


Figure 6 – Semantic Lifting Architecture

3.1.2 Context-Adaptive Questionnaire

Leading partners: FHNW

Integrated into: Design Environment

Concept and technical details: D3.1 [24]

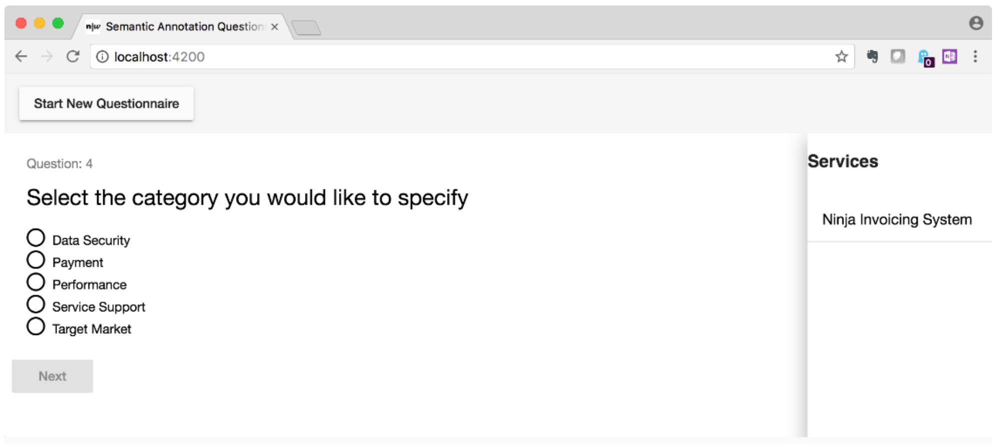
The context-adaptive questionnaire prototype provides a more user centric approach than the Semantic Lifting prototype. It guides the user through a set of questions that reflect the business process functional and non-fucntional requirements. Follow-up questions are calculated via a smart algorithm that dynamically prioritises the questions to display. Questions are prioritised such that services are discovered as quickly as possible.

The questionnaire can be applied on the whole process, group of activities and / or single activities. If no services can be found for the process, we can then move down to groups of activities, until the level of single activities.

The context-adaptive questionnaire prototype comes with a user interface that contains:

- a question stepper,
- a result service viewer and
- a search enpoint for drill-down and selection questions.

Questions focus first on functional requirements by first asking to introduce an Action and an Object. The combination of these two provides "what-is-about" knowledge, which can apply on the whole process, group of activities or single activities. For instance, if the process is about sending invoice, the user will select *send* as the action while *invoice* as the object. Next, the user/broker is asked to select the most appropriate APQC category.For this, a search function was implemented to enable the identification of the category number and/or the category name. Next, the user / broker is asked to start with the preferred dimension for the non-functional requirements (e.g. performance) and respective questions will be displayed according to an intelligent question prioritisation algorithm.



Question: 4

Select the category you would like to specify

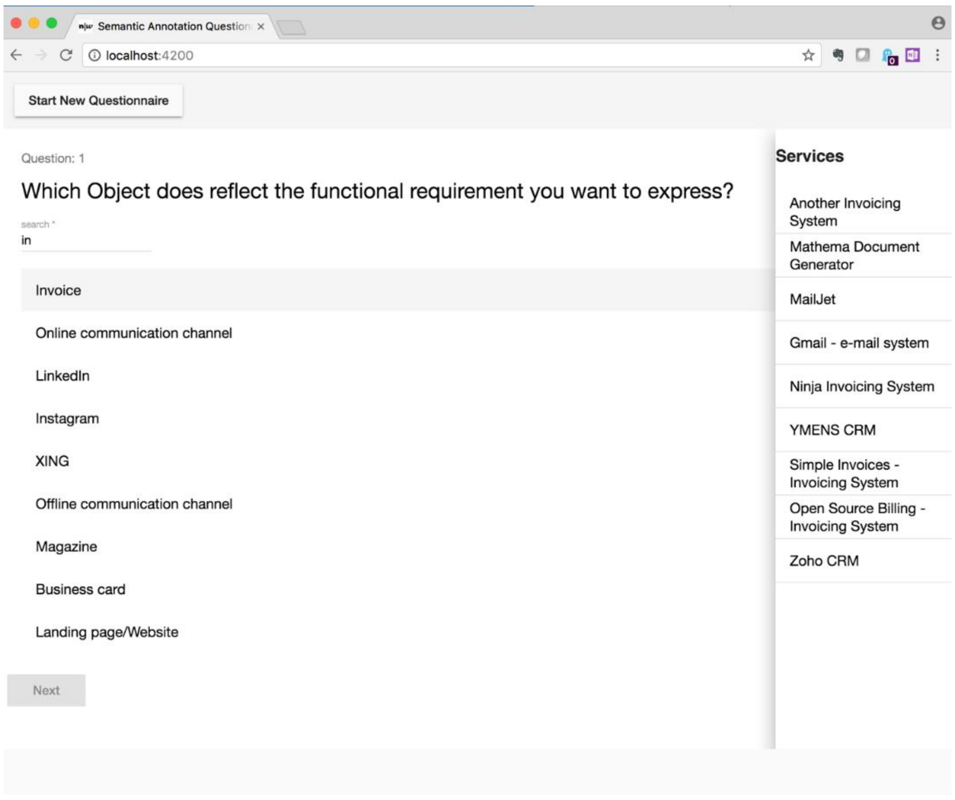
- Data Security
- Payment
- Performance
- Service Support
- Target Market

Ninja Invoicing System

Figure 7: Questionnaire showing a single select question

Single select questions provide a set of answers from which only one can be chosen. Multiselect questions provide a set of possible answers from which more than one can be selected. However, value insert and search-insert questions require input from the user. While value insert questions provide the possibility to input values, such as integer and double values (e.g. downtime in minutes, number of people working in the process, etc.), search-insert questions provide the possibility to crawl predefined values from the ontology and select the appropriate value.

Considering the list of APQC processes, the user can insert keywords for the process he/she is looking for. The ontology returns the concepts that are matching the keywords. An example is given in the following picture.



Question: 1

Which Object does reflect the functional requirement you want to express?

search *
in

Invoice
Online communication channel
LinkedIn
Instagram
XING
Offline communication channel
Magazine
Business card
Landing page/Website

Services

Another Invoicing System
Mathema Document Generator
MailJet
Gmail - e-mail system
Ninja Invoicing System
YMENS CRM
Simple Invoices - Invoicing System
Open Source Billing - Invoicing System
Zoho CRM

Figure 8: Questionnaire showing a search-insert question

The questionnaire is integrated in the BPaaS Design Environment and comprises the following:

- A web service enabling to match the answered questions into a set of service specifications. It contains the intelligent question prioritizer (see right-hand side of Figure 9, labelled as /questionnaire), the discovery component (see right-hand side of Figure 9, labelled as /discover), and a search function (see right-hand side of Figure 9, labelled as /search).
- A user Interface (see left-hand side of Figure 9), containing the Question viewer, the search box, and the service viewer. These are making use of the components implemented in the SMART web service (see arrows in Figure 9).

As already mentioned above, the questionnaire starts by displaying questions related first to functional requirements and then to non-functional requirements. The latter are displayed according to the intelligent question prioritization algorithm. The follow-up question is displayed according to:

- Domain-specific dependencies among non-functional attributes. For instance, assuming that a cloud service has availability and response time as attributes of the domain «performance». If the user is asked to answer one of the two, the follow-up question will be the remaining one in the same domain.
- Entropy (i.e., expected values of the information contained in each message) of each non-funcitonal attribute (e.g. availability). Entropy of an attribute is «0» when every service that is stored in a repository contains the same annotated attribute, while "1" in the opposite case. For example, if the repository has in total seven cloud services with data location in Switzerland (i.e., entropy = 0), the question relating to the preferred data location will not be asked as it will not filter out any services from the Service Viewer.

Figure 10, Figure 11 and Figure 12 show the idea of the discovery component. Namely, a temporary questionnaire ontology model instance is created out of the questions and related answers. Next, we first apply sematic rules to bound this model instance with the BPaaS Ontology (Figure 10– D3.1 introduces the BPaaS Ontology). Next, semantic rules are applied on the BPaaS Ontology to translate business prosess requirements (coming from answers) into cloud service specifications (Figure 11). The result is then compared through a SPARQL query with the specifications of all cloud services in the repository, and matching cloud services are displayed in the UI (Figure 12).

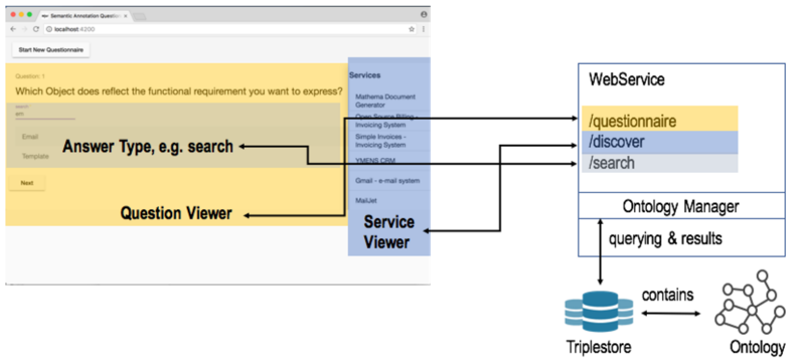
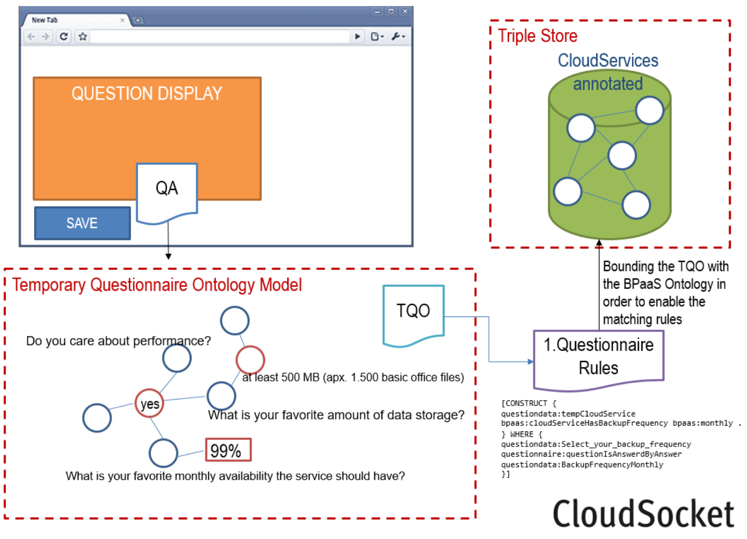
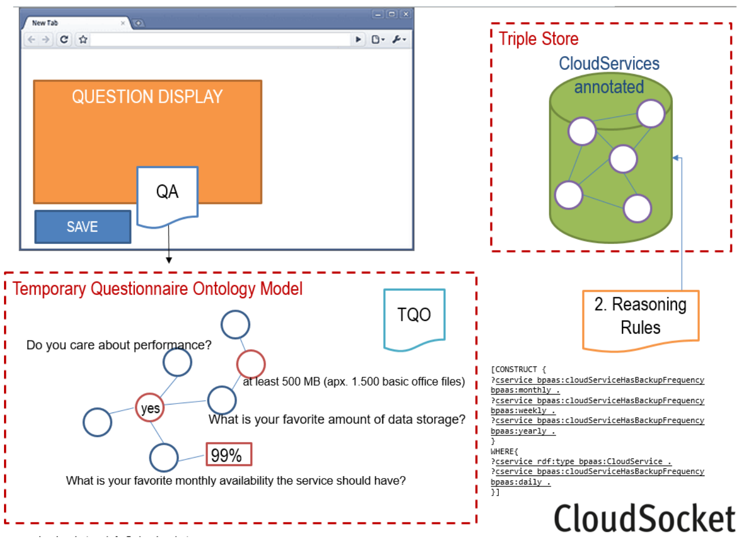


Figure 9: Context Adaptive Questionnaire


New Tab
QUESTION DISPLAY
QA
SAVE
Temporary Questionnaire Ontology Model
TQO
Do you care about performance?
at least 500 MB (apx. 1.500 basic office files)
yes
What is your favorite amount of data storage?
99%
What is your favorite monthly availability the service should have?
Triple Store
CloudServices
annotated
Bounding the TQO with the BPaaS Ontology in order to enable the matching rules
1.Questionnaire Rules
CloudSocket

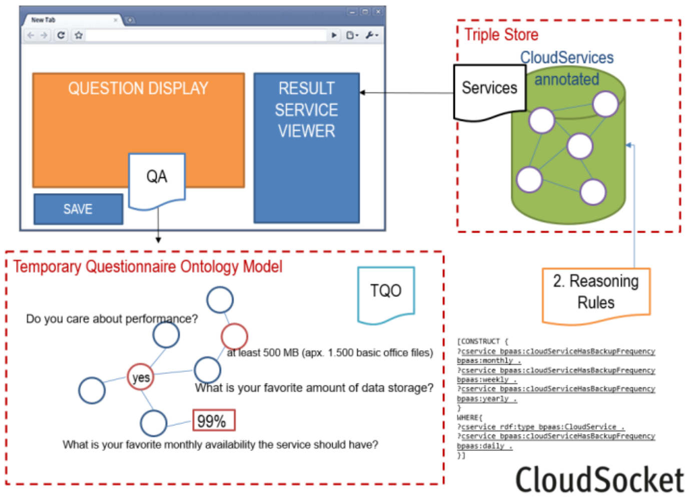
```
[CONSTRUCT {
questiondata:tempCloudService
bpaas:cloudServiceHasBackupFrequency bpaas:monthly .
} WHERE {
questiondata:Select_your_backup_frequency
questionnaire:questionIsAnswerdByAnswer
questiondata:BackupFrequencyMonthly
}]
```

Figure 10: Discovery component (1/3)


QUESTION DISPLAY
QA
SAVE
Triple Store
CloudServices annotated
2. Reasoning Rules
Temporary Questionnaire Ontology Model
TQO
Do you care about performance?
at least 500 MB (apx. 1.500 basic office files)
yes
What is your favorite amount of data storage?
99%
What is your favorite monthly availability the service should have?
CloudSocket

```
[CONSTRUCT {
?cservice bpaas:cloudServiceHasBackupFrequency
bpaas:monthly .
?cservice bpaas:cloudServiceHasBackupFrequency
bpaas:weekly .
?cservice bpaas:cloudServiceHasBackupFrequency
bpaas:yearly .
}
WHERE{
?cservice rdf:type bpaas:CloudService .
?cservice bpaas:cloudServiceHasBackupFrequency
bpaas:daily .
}]
```

Figure 11: Discovery component (2/3)


QUESTION DISPLAY
RESULT SERVICE VIEWER
QA
SAVE
Triple Store
CloudServices annotated
Services
2. Reasoning Rules
Temporary Questionnaire Ontology Model
TQO
Do you care about performance?
at least 500 MB (apx. 1.500 basic office files)
yes
What is your favorite amount of data storage?
99%
What is your favorite monthly availability the service should have?
CloudSocket

```
[CONSTRUCT {
?cservice bpaas:cloudServiceHasBackupFrequency
bpaas:monthly .
?cservice bpaas:cloudServiceHasBackupFrequency
bpaas:weekly .
?cservice bpaas:cloudServiceHasBackupFrequency
bpaas:yearly .
}
WHERE{
?cservice rdf:type bpaas:CloudService .
?cservice bpaas:cloudServiceHasBackupFrequency
bpaas:daily .
}]
```

Figure 12: Discovery component (3/3)

3.2 BPaaS Allocation Environment Prototypes

3.2.1 CAMEL PaaS/SaaS support

Leading partners: FORTH, UULM

Integrated into: CAMEL meta-model / DSL

Concept and technical details: D3.3 [32], D3.4 [33]

The cloud computing stack comprises currently three major cloud service models, namely IaaS, PaaS and SaaS. While the original version of CAMEL in the PaaSage project focused only on the IaaS level, CloudSocket extended CAMEL in order to provide also support for the coverage of the PaaS and SaaS levels for the following reasons: (a) PaaS gains a significant momentum lately as it enables users to focus on the development and provisioning of the core application functionality without requiring to deal with any information regarding the underlying infrastructure; (b) the deployment of BPaaS service components can be accelerated via the use of a PaaS service and this can be invaluable both for the initial BPaaS deployment as well as its reconfiguration; (b) the coverage of the SaaS level enables to represent service task allocations as well as to measure both BPaaS workflow tasks and services.

3.3 BPaaS Execution Environment Prototypes

3.3.1 PaaS Orchestration

Leading partners: UULM, ATOS

Integrated into: Cloud Provider Engine

Concept and technical details: D3.3 [32], D3.4 [33]

As the Cloud Provider Engine of CloudSocket, namely the Cloudiator Framework, focuses in its first version on the orchestration of multi-cloud deployments on the IaaS level, the orchestration of additional cloud service levels was required to enable a holistic BPaaS Bundle deployment. In order to enable the usage of additional cloud service levels, the Cloud Provider Engine is extended by the PaaS Orchestration Prototype to support the orchestration on the PaaS level. The enhanced Cloud Provider Engine enables the deployment of services on IaaS and PaaS resources in a transparent manner, reducing the technical complexity and providing the Broker a significantly extended set of cloud services that can be exploited. With the integrated PaaS orchestration, the Cloud Provider Engine is able to support the deployment of software components on OpenShift, Heroku and CloudFoundry. An example is provided in section 0.

3.4 BPaaS Evaluation Environment Prototypes

3.4.1 Evaluation Ontology

Leading partners: FORTH

Integrated into: Many components of the BPaaS Evaluation Environment (see details below)

Concept and technical details: D3.5 [36], D3.6 [38]

In order to support the different kinds of analysis that can be performed over a BPaaS, we need to be able to appropriately capture the whole BPaaS hierarchy. Such a capturing needs also to be performed via an ontology in order to enable some semantic inferencing that could take the form of matching different elements that can be involved in such a hierarchy. By also considering the state-of-the-art, current propositions are either narrow-scoped, non-semantic or focus on at most two abstraction levels. To this end, to solve the above problem as well as advance the state-of-the-art, a novel ontology has been designed and implemented called Evaluation Ontology. This ontology does cover all abstraction levels as well as appropriately links all elements in each level to support the complete coverage of a BPaaS hierarchy. It also provides all necessary details required for the description of each element. This ontology has also been linked to the OWL-Q KPI extension ontology (see next sub-section) in order to enable to correlate a BPaaS deployment with all the measurements over all KPI metrics that have been produced for it.

Many components of the BPaaS Evaluation environment currently exploit this ontology which also witnesses its degree of integration:

- The Harvester in order to semantically uplift deployment information.

- The Conceptual Analytics Engine in order to formulate the appropriate KPI evaluation queries over the Semantic Knowledge Base.
- The Deployment Discovery Engine in order to record the deployments of a BPaaS workflow and rank them based on their execution history.

3.4.2 OWL-Q KPI Extension

Leading partners: FORTH

Integrated into: Many components of the BPaaS Evaluation Environment (see details below)

Concept and technical details: D3.5 [36], D3.6 [38]

One of the most critical BPaaS analysis functionality relates to the evaluation of the BPaaS performance according to a set of KPIs. However, by considering the state-of-the-art, we can see that most KPI meta-models are either not rich or extensive enough, with clear lack on some measurement details and aspects, leading to the implementation of fixed and inflexible KPI measurement & analysis systems. Moreover, such KPI meta-models are in many cases non-semantic, thus prohibiting any kind of semantic analysis to be performed over them. To this end, to cover this gap but also not to design a new ontology from scratch, we have relied on the OWL-Q[1] Ontology and especially its specification facet, in order to extend it accordingly so as to cover the KPI domain. By relying on OWL-Q, all measurement details are provided which enables the implementation of more flexible KPI measurement systems that can measure any kind of KPI. The OWL-Q KPI extension also exhibits some nice features, such as the linkage of KPIs to goals in order to enable the assessment of the satisfaction degree of an organisations strategic, tactical and operational goals, the capability to call external information sources for generating a required input parameter of a KPI metric formula, and the grouping of KPIs to form KPI hierarchies in order to support root-cause analysis in the form of KPI drill-down.

Many components of the BPaaS Evaluation environment currently exploit this ontology which also witnesses its degree of integration:

- The Harvester in order to semantically uplift monitoring information.
- The Conceptual Analytics Engine in order to formulate the appropriate KPI evaluation queries over the Semantic Knowledge Base.
- The Deployment Discovery Engine in order to rank the deployments of a BPaaS workflow according to their execution history and especially their measurements over respective KPI metrics.

3.4.3 Harvester

Leading partners: FORTH

Integrated into: BPaaS Evaluation Environment

Concept and technical details: D3.5 [36], D3.6 [38]

[1] Kritikos, Kyriakos, and Dimitris Plexousakis. "OWL-Q for semantic QoS-based web service description and discovery." Proceedings of the 2007 International Conference on Service Matchmaking and Resource Retrieval in the Semantic Web-Volume 243. CEUR-WS. org, 2007.

By relying on the aforementioned ontologies, relevant information can be gathered, semantically uplifted and then stored in the Semantic Knowledge Base in order to support different kinds of analysis over it. This is the task of the Harvester component, a more or less independent component from the rest of the BPaaS Evaluation Environment, which gathers measurement, deployment and registry information from various components of the CloudSocket platform. This component collects this information in regular time intervals by invoking the respective REST APIs offered by the latter components. A nice feature of this component is that it enables multi-tenancy in the BPaaS Evaluation Environment. This is achieved via the storage of the collected and semantically uplifted information into broker-specific partitions of the Semantic Knowledge Base. As such, all kinds of analysis offered by the BPaaS Evaluation Environment can be performed in a dedicated manner for each CloudSocket broker. Another important feature lies on its capability to mark down the previous collection time point such that it can then collect only new information from the current information sources exploited.

3.4.4 Conceptual Analytics Engine

Leading partners: FORTH

Integrated into: BPaaS Evaluation Environment

Concept and technical details: D3.5 [36], D3.6 [38]

A critical analysis functionality over a BPaaS relates to the evaluation of its performance via a set of KPIs as well as the ability to find root-causes when one or more of the KPIs in this set are violated. Such analysis functionality is already offered either partially (only KPI evaluation) or in full by various prototype systems. However, such systems suffer from the following disadvantages: (a) there is inflexibility in the evaluation of KPIs as only a small, fixed set of KPIs is supported; (b) in case that there is some flexibility in KPI evaluation via the introduction of new KPIs that can be evaluated, this flexibility comes with the drawback that the definition of such KPIs is performed in a low level which leads to the requirement of possessing the appropriately knowledge and expertise at that level; (c) there is a lack of semantics which can prohibit reaching a high-accuracy level in the evaluation; (d) at most two levels in the BPaaS hierarchy are covered; (e) root-cause analysis relies on machine learning techniques which pre-suppose that a rich execution history is already in place for a certain BPaaS. To resolve these drawbacks, a new component has been designed and developed by building on the content structured by the Harvester in the Semantic Knowledge Base. This component called the Conceptual Analytics Engine supports the semantic evaluation and drill-down of KPIs. More importantly, this component allows the BPaaS broker to explore the possible KPI metric space via the use of a high-level language which is not tied to low-level peculiarities, something invaluable in the usually creative process of determining and defining the right KPIs for a certain BPaaS. This also brings the appropriate flexibility which can enable a KPI measurement system to measure any KPI and not only those belonging to a certain fixed set. In addition, it enables the definition and evaluation of KPIs at all levels of the BPaaS hierarchy. In contrast to the state-of-the-art, this component finally enables root cause analysis via KPI drill-down that can take two forms: (a) drill-down via exploiting KPI hierarchies: this means that the Conceptual Analytics Engine can evaluate a top-level KPI as well as its descendants; (b) drill-down via exploiting KPI metric hierarchies: here the engine can exploit the hierarchy involved in the aggregation of a KPI metric in order to enable to evaluate all the metrics involved in this hierarchy. As such, these two KPI drill-down forms rely on already established relationships between KPIs and their metrics rather than relying on machine learning to also attempt to discover such relationships, in case a rich execution history exists for a certain BPaaS.

3.4.5 Deployment Discovery Engine

Leading partners: FORTH

Integrated into: BPaaS Evaluation Environment

Concept and technical details: D3.5 [36], D3.6 [38]

While attempting to allocate a BPaaS, even with a sophisticated mechanism as the one reported in D3.4 [33], mapping to the Smart Service Discovery and Selection Tool, such an allocation relies on the advertisement of the service non-functional capabilities by the service provider. However, such capabilities may be formed in an imprecise manner. Moreover, for large BPaaSes, involving many BPaaS (service) components, the solution space can be quite big, thus leading to increased service selection times which hamper the user experience. In this respect, there is a need for a new mechanism which can address the inaccurate description of services by also providing suitable knowledge in order to accelerate the service selection process. Such a mechanism comes via the capability to discover the best deployments for a certain BPaaS or its parts by exploring the execution history of this BPaaS. Such a knowledge can then be exploited during service selection to fix parts of the optimisation problem to be solved or rely the selection only over a small set of best deployments. This capability is offered by the Deployment Discovery Engine which is currently able to discover the best deployments for a BPaaS only at the workflow level. This new component also exhibits the important feature of being capable of deriving the similarity of BPaaSes based on the similarity of their workflows. As such, this enables to consider a broader execution history for a certain BPaaS that includes the one recorded for the BPaaSes that resemble it. In result, by broadening the execution history, more optimised allocation solutions can be derived.

3.4.6 Process Mining Engine

Leading partners: FORTH

Integrated into: BPaaS Evaluation Environment

Concept and technical details: D3.5 [36], D3.6 [38]

The recording of the execution history of a business process (BP) (or a BPaaS) enables to perform various kinds of analysis over it. One analysis kind comes with the employment of process mining techniques. Such techniques have been used in the past to: (a) recreate the model of a BP to enable to compare it with the one originally designed in order to possibly detect some discrepancies or points for more optimisation; (b) to check the conformance of a BP against a set of requirement; (c) to discover decision points for user/manual tasks and thus provide a higher automation degree in BP execution; (d) to perform organisational mining to discover interesting organisational patterns that can also assist in the improvement of the currently deployed organisational structures. Moreover, by also exploiting semantics, coming from the annotation of the BPs with ontology concepts, such techniques can rich a higher-level of accuracy. In this respect and by relying on the rich execution history recorded for BPaaSes, a new component has been designed and implemented, called Process Mining Engine, which is able to currently integrate state-of-the-art process mining algorithms focusing on BP model re-creation. A nice feature of this component is that it does support semantic process mining. This is achieved by exploiting the semantic annotations that are available for a BPaaS workflow tasks and extracting a process log from the BPaaS execution history which includes them and not the actual names of these tasks. As such, any process mining algorithm / technique exploited is transformed into a semantic one. The outcome of the execution of this component is a BPMN model which can be immediately compared with the original one designed for the BPaaS at hand by using the facilities of the Hybrid Business Dashboard of the BPaaS Evaluation Environment.

4 PROTOTYPE FROM CUSTOMER PERSPECTIVE

4.1 Introduction of Customer perspective

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 information is provided 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.

4.2 Demonstration from BPaaS Customer Viewpoint

4.2.1 Involved Roles

In the demonstration, we assume a scenario in which Susan, a fictive biologist, funds a Start-up to sell special worms named Golden Worm Company, and needs a cloud solution to send invoices to her clients.

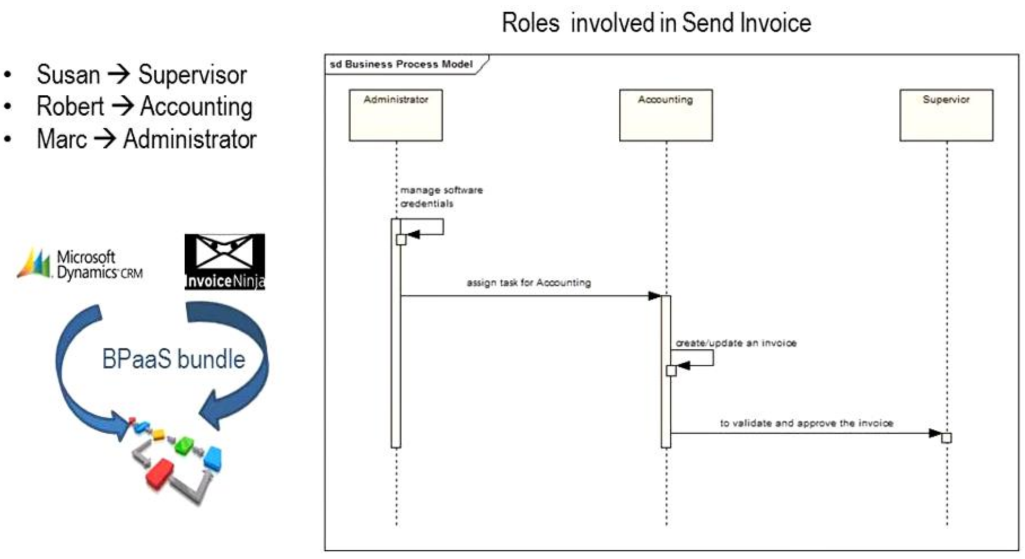


Figure 13 - BPaaS Customer perspective roles

The actors from the CloudSocket-Customer – the Golden Worm Company - present in this scenario are:

- Susan Smidt as the supervisor of the Start-up.
- Robert Herrman as the accounting representative of the Start-up.
- Marc Miller as the system administrator of the Start-up.

The actors from the CloudSocket Broker presented in this scenario are:

- BWCON as the broker, contacted by Susan in order to identify the best solution that fits her needs.

In this demonstration, Susan will access the marketplace and perform the purchase of the Send Invoice bundle. After Susan completes the checkout, the Send Invoice bundle deployment is triggered and Daniel, an Execution Environment Engineer Operator, checks the new deployments on the Cloud Provider Engine UI. After the deployment, Marc, who belongs to the Golden Worm Company and has the admin role, configures the bundle guided by a provided handbook. Once the configuration is completed and the Send Invoice Workflow is ready, Robert, who belongs to the accounting representative group in the Golden Worm Company, start creating an invoice. At the end, Susan, who belongs to the supervisor group, has to accept the Invoice, and the mail with the invoice will be sent to the client.

4.2.2 Purchase the BPaaS "Send Invoice"

Susan Smidt logs into CloudSocket Marketplace (Figure 14), whose URL was given by BWCON, the broker (Figure 15).

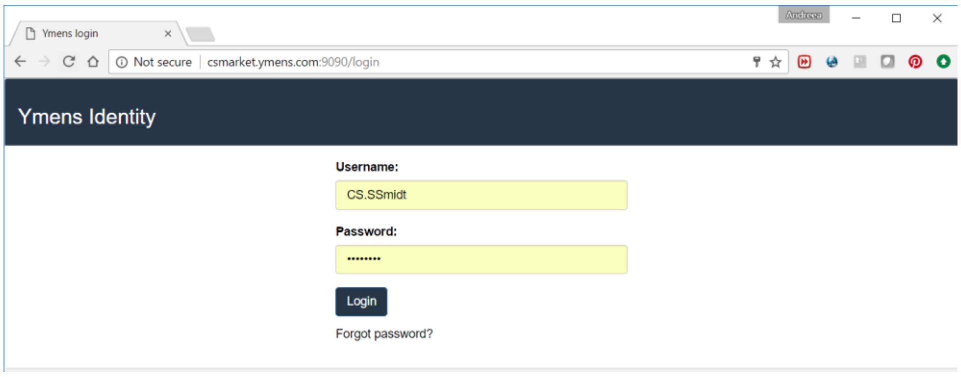
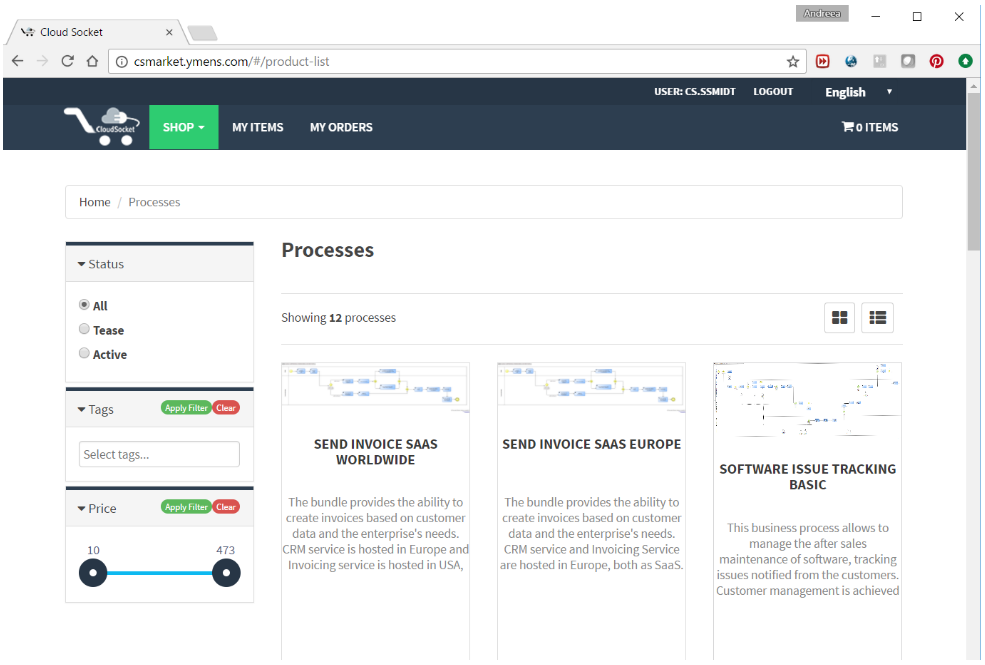


Figure 14 – Login Marketplace



SEND INVOICE SAAS WORLDWIDE

The bundle provides the ability to create invoices based on customer data and the enterprise's needs. CRM service is hosted in Europe and Invoicing service is hosted in USA,

SEND INVOICE SAAS EUROPE

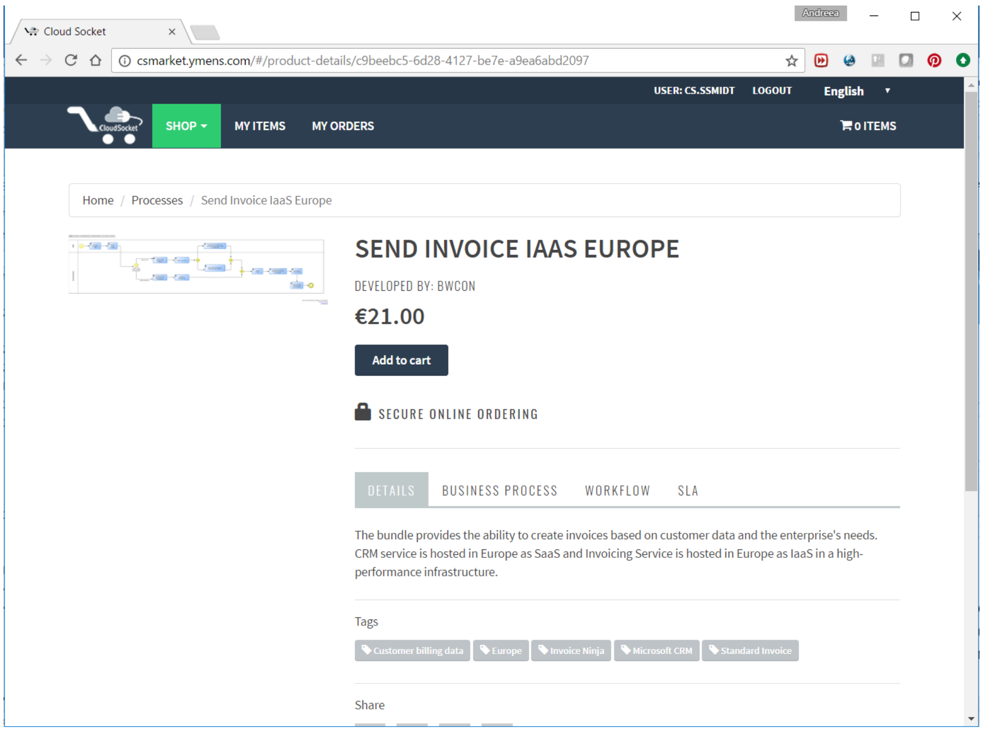
The bundle provides the ability to create invoices based on customer data and the enterprise's needs. CRM service and Invoicing Service are hosted in Europe, both as SaaS.

SOFTWARE ISSUE TRACKING BASIC

This business process allows to manage the after sales maintenance of software, tracking issues notified from the customers. Customer management is achieved

Figure 15 – Browsing the offer of the broker.

After being successfully logged in (Figure 14), Susan searches for appropriate BPaaS bundles that are being offered by BWCON and identifies a fitting one. Susan has the chance to view related details of the selected BPaaS bundle in both a textual and graphical manner (Figure 16). The respective marketplace information is provided as descriptions, tags, bundle prices and SLAs.


Cloud Socket

csmarket.ymens.com/#/product-details/c9beebc5-6d28-4127-be7e-a9ea6abd2097

USER: CS.SSMIDT LOGOUT English

SHOP MY ITEMS MY ORDERS 0 ITEMS

Home / Processes / Send Invoice IaaS Europe

SEND INVOICE IAAS EUROPE

DEVELOPED BY: BWCON

€21.00

Add to cart

SECURE ONLINE ORDERING

DETAILS BUSINESS PROCESS WORKFLOW SLA

The bundle provides the ability to create invoices based on customer data and the enterprise's needs. CRM service is hosted in Europe as SaaS and Invoicing Service is hosted in Europe as IaaS in a high-performance infrastructure.

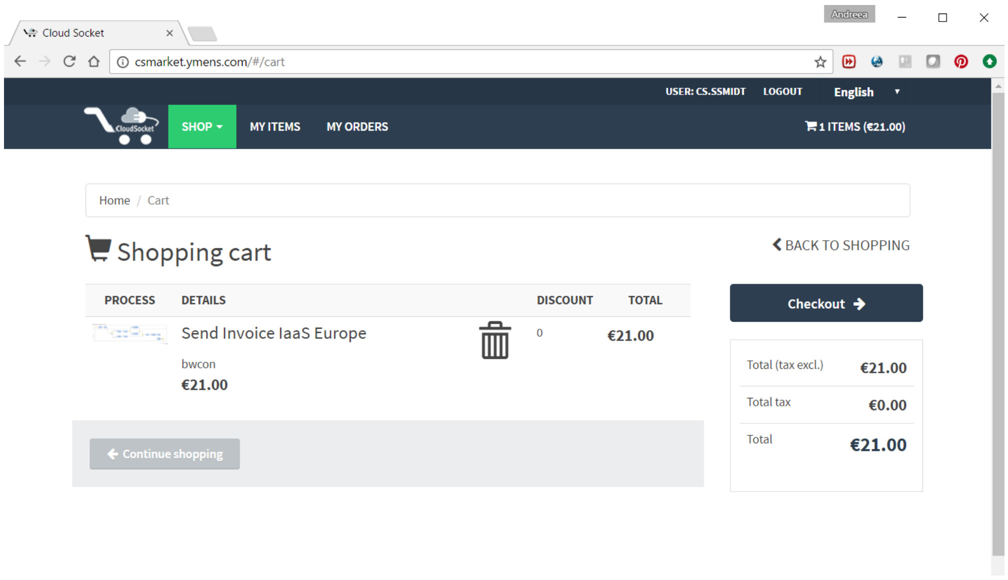
Tags

Customer billing data | Europe | Invoice Ninja | Microsoft CRM | Standard Invoice

Share

Figure 16 – View BPaaS bundle details.

Susan performs a checkout to her cart (Figure 17). She has two options for payment: PayPal or Payment Order (Figure 18). She chooses to use her PayPal account. She must agree to terms and conditions to make the payment and to finalize the order. A notification window provides information that the order was successfully registered and that the deployment has been triggered, attempting to deploy the BPaaS bundle (along with all related assets) in the cloud (Figure 19). Susan will be notified by email as soon as the deployment has been completed.


Cloud Socket

csmarket.ymens.com/#/cart

USER: CS.SSMIDT LOGOUT English

SHOP MY ITEMS MY ORDERS 1 ITEMS (€21.00)

Home / Cart

Shopping cart

BACK TO SHOPPING

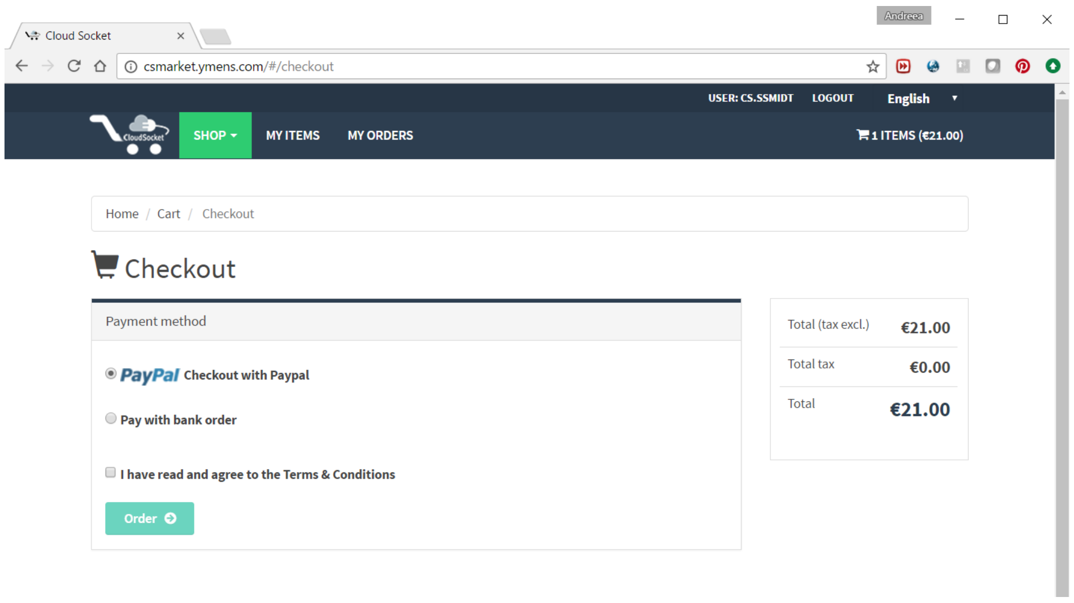
PROCESS	DETAILS	DISCOUNT	TOTAL
	Send Invoice IaaS Europe bwcon €21.00	0	€21.00

Continue shopping

Checkout

Total (tax excl.)	€21.00
Total tax	€0.00
Total	€21.00

Figure 17 – Marketplace cart



Checkout

Payment method

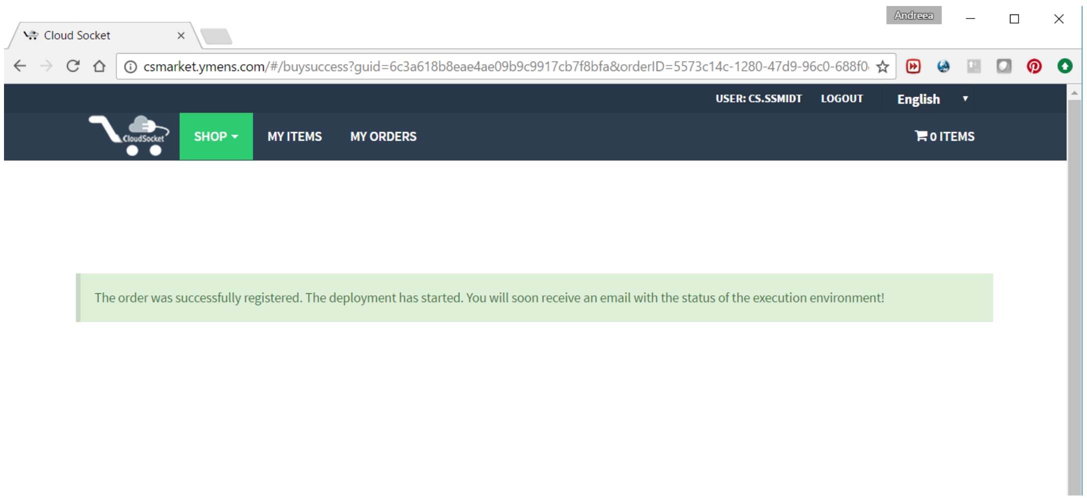
[✓ PayPal Checkout with Paypal]

Pay with bank order

I have read and agree to the Terms & Conditions

Total (tax excl.)	€21.00
Total tax	€0.00
Total	€21.00

Figure 18 – Payment options


Cloud Socket
csmarket.ymens.com/#/buysuccess?guid=6c3a618b8eae4ae09b9c9917cb7f8bfa&orderID=5573c14c-1280-47d9-96c0-688f0
USER: CS.SSMIDT LOGOUT English
SHOP MY ITEMS MY ORDERS
0 ITEMS

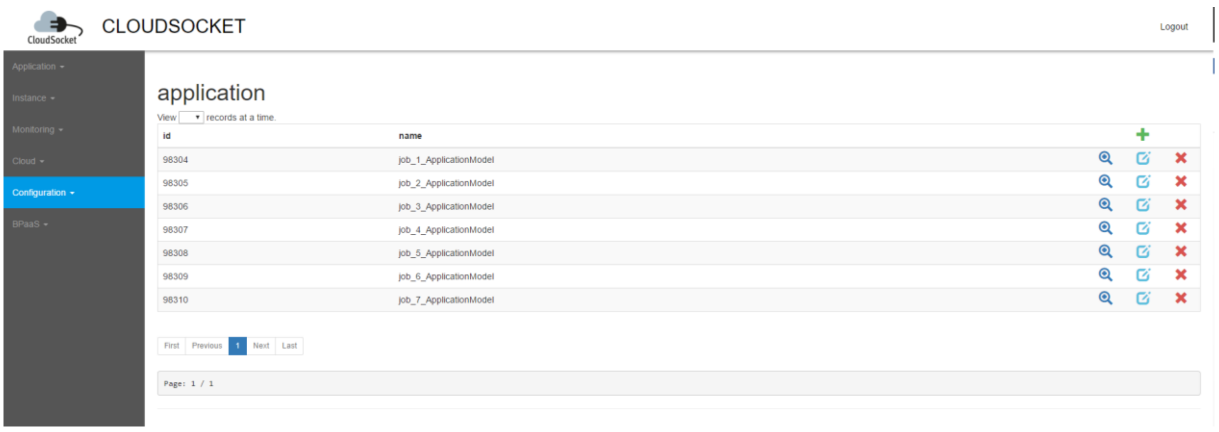
The order was successfully registered. The deployment has started. You will soon receive an email with the status of the execution environment!

Figure 19 – Notification message for the purchased BPaaS bundle

The following video shows how Susan buys a Send Invoice IaaS Europe bundle to satisfy her requirements: https://youtu.be/GeDKVHqXpbY

4.2.3 Excursus in the Infrastructure

The Send Invoice bundle deployment has been triggered by the interaction of Susan with the marketplace. In parallel, Daniel, acting as an Execution Environment Engineer Operator, checks the new deployments from the UI (Figure 20) of the Cloud Provider Engine and intervenes if something wrong takes place.



id	name
98304	job_1_ApplicationModel
98305	job_2_ApplicationModel
98306	job_3_ApplicationModel
98307	job_4_ApplicationModel
98308	job_5_ApplicationModel
98309	job_6_ApplicationModel
98310	job_7_ApplicationModel

Page: 1 / 1

Figure 20 - Cloud Provider Engine Dashboard

The deployment of the Send Invocie 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 (based on the extended Cloud Provider Engine and the integrated research prototypes; FiWare and PaaS offerings are now supported [32]) and also setup related services (e.g. monitoring) in the infrastructure. Figure 21 provides 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.

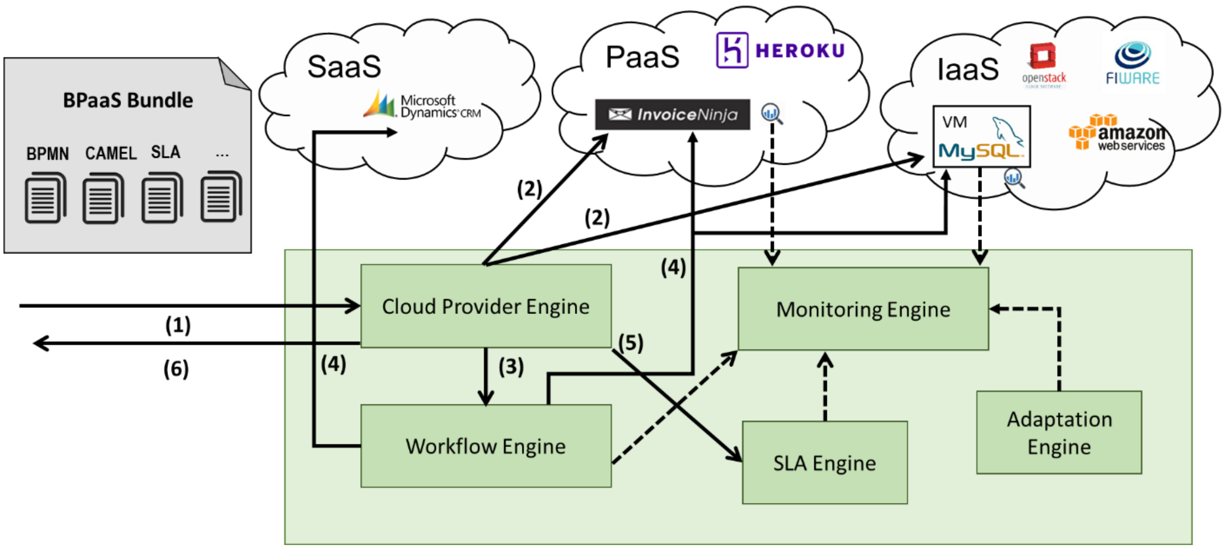
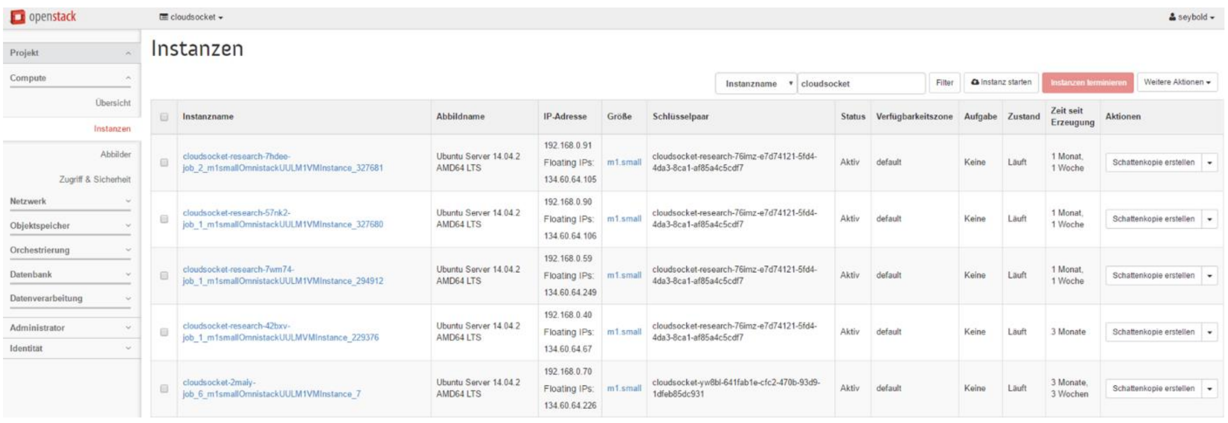


Figure 21 - BPaaS Bundle Deployment

(1) The cloud provider engine receives the BPaaS bundle from the Marketplace, when Susan has finished the order process. The bundle mainly contains the BPMN file, the CAMEL file and the SLA template along with additional meta 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 deploy the software components, which means (i) for IaaS: *provision the required virtual machines, deploy the software components, configure the monitoring and receive the actual endpoints for the (internal) service instances;* (ii) for PaaS: *provision the required runtime environment, deploy the software components, configure the monitoring and receive the actual endpoints for the (internal) service instances*.

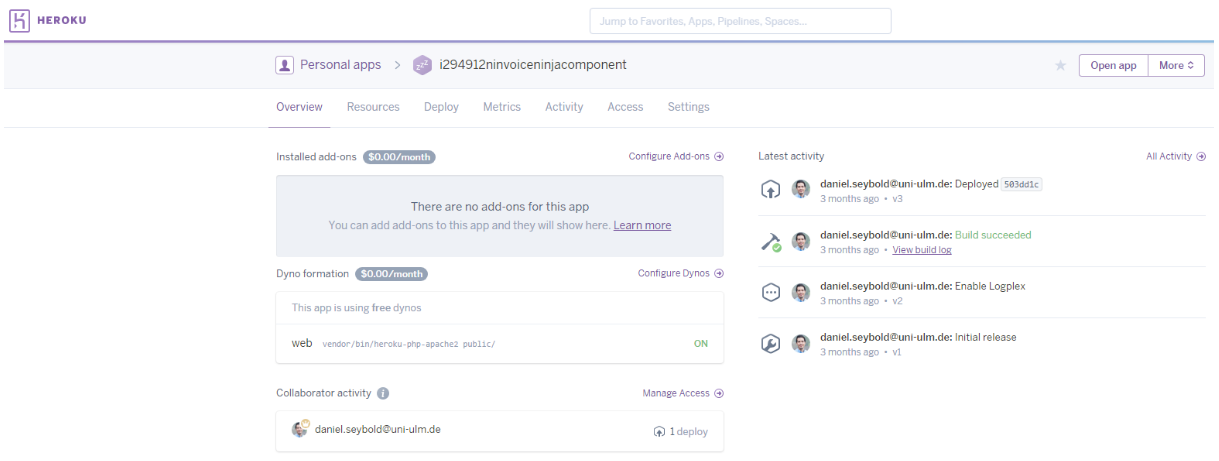
Figure 22 - Figure 24 show sample screenshot mapping the deployment of the Invoice Ninja software component to possible cloud infrastructures, e.g. OpenStack at Ulm University, the Heroku PaaS service and the FIWARE IaaS service, where Heroku and FIWARE represent the enhanced deployment capabilities of the Cloud Provider Engine by the research prototypes [32]. A detailed demonstration of the added-values of the research prototypes can be found at [31].



Instanzen

Instanzname	Abbildname	IP-Adresse	Größe	Schlüsselpaar	Status	Verfügbarkeitszone	Aufgabe	Zustand	Zeit seit Erzeugung	Aktionen
cloudsocket-research-7hdee-job_2_m1smallOmnistackUULM1VMInstance_327681	Ubuntu Server 14.04.2 AMD64 LTS	192.168.0.91 Floating IPs: 134.60.64.105	m1.small	cloudsocket-research-76imz-e7d74121-5fd4-4da3-8ca1-af85a4c5cdf7	Aktiv	default	Keine	Läuft	1 Monat, 1 Woche	Schattenkopie erstellen
cloudsocket-research-57nk2-job_1_m1smallOmnistackUULM1VMInstance_327680	Ubuntu Server 14.04.2 AMD64 LTS	192.168.0.90 Floating IPs: 134.60.64.106	m1.small	cloudsocket-research-76imz-e7d74121-5fd4-4da3-8ca1-af85a4c5cdf7	Aktiv	default	Keine	Läuft	1 Monat, 1 Woche	Schattenkopie erstellen
cloudsocket-research-7wm74-job_1_m1smallOmnistackUULM1VMInstance_294912	Ubuntu Server 14.04.2 AMD64 LTS	192.168.0.59 Floating IPs: 134.60.64.249	m1.small	cloudsocket-research-76imz-e7d74121-5fd4-4da3-8ca1-af85a4c5cdf7	Aktiv	default	Keine	Läuft	1 Monat, 1 Woche	Schattenkopie erstellen
cloudsocket-research-42bxv-job_1_m1smallOmnistackUULMVMInstance_229376	Ubuntu Server 14.04.2 AMD64 LTS	192.168.0.40 Floating IPs: 134.60.64.67	m1.small	cloudsocket-research-76imz-e7d74121-5fd4-4da3-8ca1-af85a4c5cdf7	Aktiv	default	Keine	Läuft	3 Monate	Schattenkopie erstellen
cloudsocket-2maly-job_6_m1smallOmnistackUULM1VMInstance_7	Ubuntu Server 14.04.2 AMD64 LTS	192.168.0.70 Floating IPs: 134.60.64.226	m1.small	cloudsocket-yw8bl-641fab1e-cfc2-470b-93d9-1dfeb85dc931	Aktiv	default	Keine	Läuft	3 Monate, 3 Wochen	Schattenkopie erstellen

Figure 22 - BPaaS Bundle on Openstack



Installed add-ons $0.00/month

There are no add-ons for this app
You can add add-ons to this app and they will show here. Learn more

Dyno formation $0.00/month

This app is using free dynos

web vendor/bin/heroku-php-apache2 public/ ON

Collaborator activity

daniel.seybold@uni-ulm.de 1 deploy

Latest activity

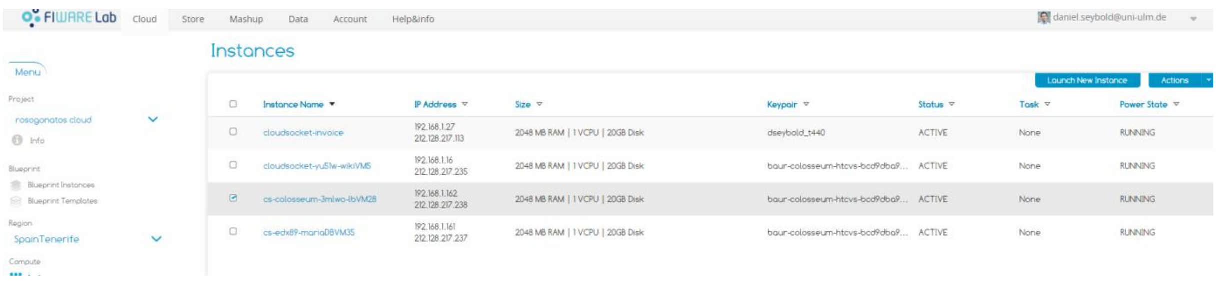
daniel.seybold@uni-ulm.de: Deployed 503dd1c
3 months ago · v3

daniel.seybold@uni-ulm.de: Build succeeded
3 months ago · View build log

daniel.seybold@uni-ulm.de: Enable Logplex
3 months ago · v2

daniel.seybold@uni-ulm.de: Initial release
3 months ago · v1

Figure 23 - BPaaS Bundle on Heroku



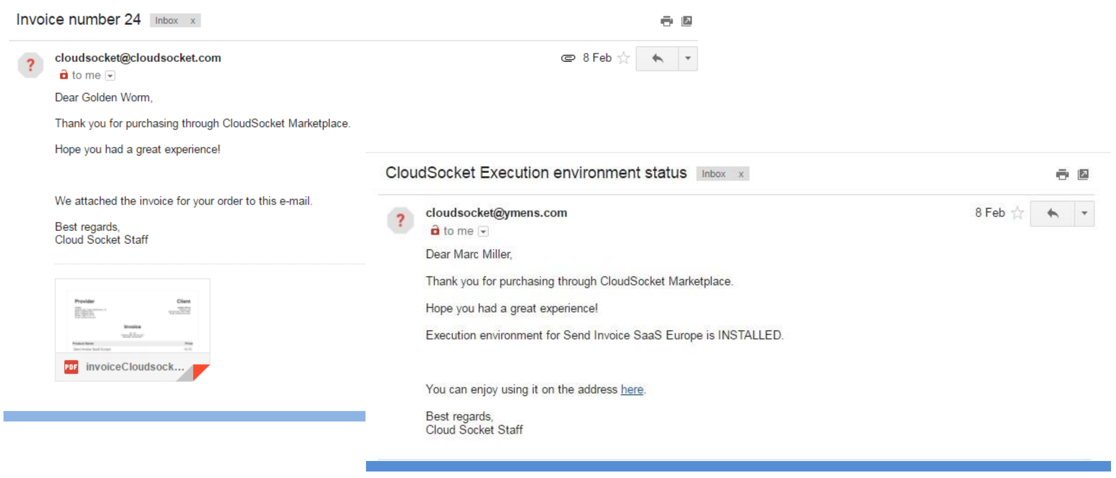
Instances

Instance Name	IP Address	Size	Keypair	Status	Task	Power State
cloudsocket-invoice	192.168.1.27 212.128.217.113	2048 MB RAM | 1 VCPU | 20GB Disk	dseybold_t440	ACTIVE	None	RUNNING
cloudsocket-yu5lw-wikiVM5	192.168.1.16 212.128.217.235	2048 MB RAM | 1 VCPU | 20GB Disk	baur-colosseum-htcvs-bcd9dba9...	ACTIVE	None	RUNNING
cs-colosseum-3mlwo-lbVM28	192.168.1.162 212.128.217.238	2048 MB RAM | 1 VCPU | 20GB Disk	baur-colosseum-htcvs-bcd9dba9...	ACTIVE	None	RUNNING
cs-edx89-mariaDBVM35	192.168.1.161 212.128.217.237	2048 MB RAM | 1 VCPU | 20GB Disk	baur-colosseum-htcvs-bcd9dba9...	ACTIVE	None	RUNNING

Figure 24 - BPaaS Bundle on FiWare

4.2.4 Customer Specific Configuration of the BPaaS “Send Invoice”

The Marketplace of Bwcon (the broker) forwards the (successful deployment) confirmation email to Marc, who belongs to the Golden Worm Company and has the admin role, in order to take care of the respective workflow execution. Then, he receives two emails: on one side, the purchased bundle invoice; and on the other side, the confirmation that the bundle has been deployed, including the link enabling the user to manage the BPaaS bundle purchased (e.g. create workflow instances, execute them or handle other lifecycle activities on them), (see Figure 25).


Invoice number 24 Inbox x

cloudsocket@cloudsocket.com — 8 Feb
to me

Dear Golden Worm,

Thank you for purchasing through CloudSocket Marketplace.

Hope you had a great experience!

We attached the invoice for your order to this e-mail.

Best regards,
Cloud Socket Staff

invoiceCloudsock...

CloudSocket Execution environment status Inbox x

cloudsocket@ymens.com — 8 Feb
to me

Dear Marc Miller,

Thank you for purchasing through CloudSocket Marketplace.

Hope you had a great experience!

Execution environment for Send Invoice SaaS Europe is INSTALLED.

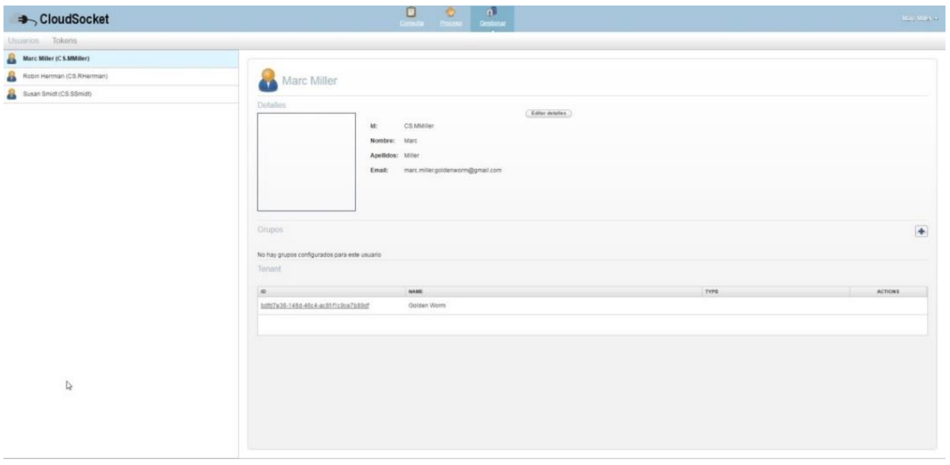
You can enjoy using it on the address here.

Best regards,
Cloud Socket Staff

Figure 25 - Email for the notification and the invoice of purchased BPaaS bundle

The process responsible of the customer is guided by the handbook (offered with the BPaaS) in order to perform the configuration of the bundle. It is not the intention to deep into details of the configuration at this document, since it is well described in the Invoice Sending handbook [41], as the Figure 28 shows. Deliverable D5.4 [39] specifies more deeply how to configure and how to use the different BPaaS bundles.

Marc, acting as a process responsible, logs in the Workflow Engine and manages the users (who have Knowledge Worker role) and their internal roles/groups (Figure 26); for these bundles there are two groups to be assigned: supervisor (responsible for validating the invoices) and accounting representative (responsible for creating the invoices).



Marc Miller (C S.MMiller)
Robin Herman (CS.RHerman)
Susan Smidt (CS.SSmidt)

Marc Miller

Detalles

Id: CS.MMiller
Nombre: Marc
Apellidos: Miller
Email: marc.miller.goldenworm@gmail.com

Grupos

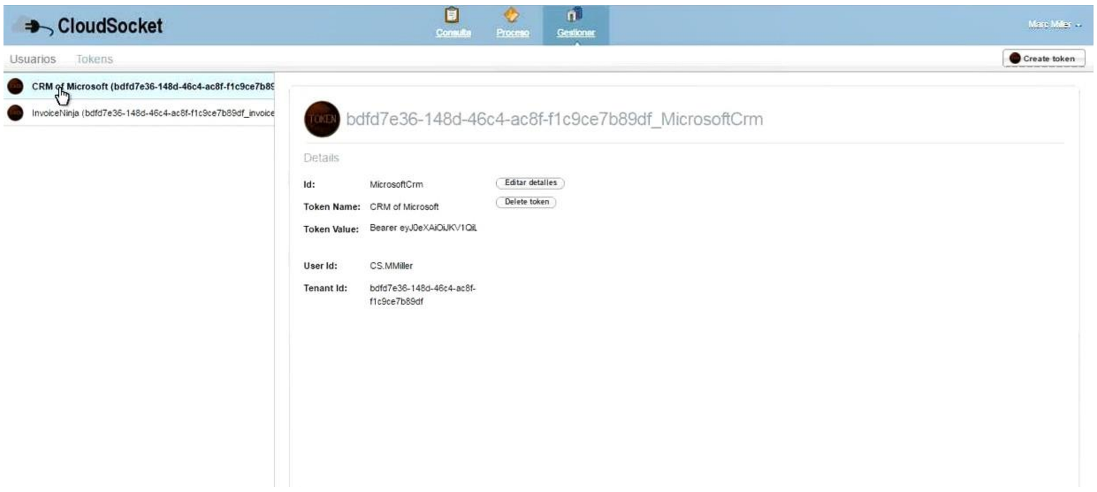
No hay grupos configurados para este usuario

Tenant

ID	NAME	TYPE	ACTIONS
bdfd7a36-148d-46c4-ac81f1c0ce7b89df	Golden Worm		

Figure 26 - Management of users and roles

He is also responsible to configure the necessary authentication tokens included in the BPaaS bundle in order to interact with the different services. In the case of the Send Invoice bundle, he needs to configure the tokens for the Microsoft CRM and the Invoice Ninja (Figure 27), which are described in detail in the mentioned handbooks (Figure 28).



CRM of Microsoft (bdfd7e36-148d-46c4-ac8f-f1c9ce7b8

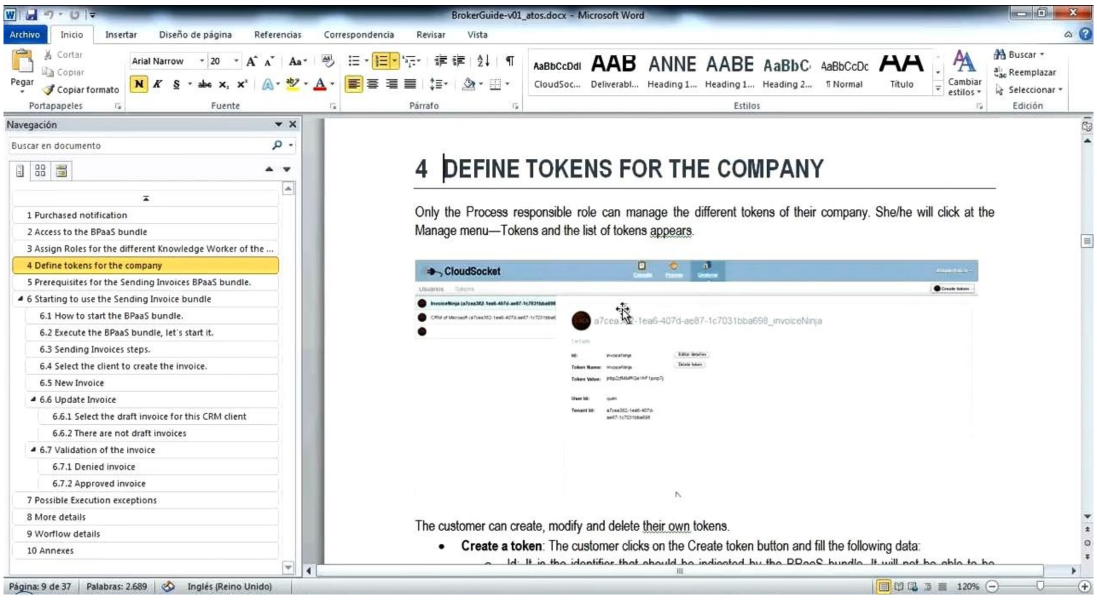
InvoiceNinja (bdfd7e36-148d-46c4-ac8f-f1c9ce7b89df_invoice

bdfd7e36-148d-46c4-ac8f-f1c9ce7b89df_MicrosoftCrm

Details

Id:	MicrosoftCrm
Token Name:	CRM of Microsoft
Token Value:	Bearer eyJ0eXAiOiJKV1QiL
User Id:	CS.MMiller
Tenant Id:	bdfd7e36-148d-46c4-ac8f-f1c9ce7b89df

Figure 27 - Management of tokens



1 Purchased notification
2 Access to the BPaaS bundle
3 Assign Roles for the different Knowledge Worker of the ...
4 Define tokens for the company
5 Prerequisites for the Sending Invoices BPaaS bundle.
6 Starting to use the Sending Invoice bundle
6.1 How to start the BPaaS bundle.
6.2 Execute the BPaaS bundle, let´s start it.
6.3 Sending Invoices steps.
6.4 Select the client to create the invoice.
6.5 New Invoice
6.6 Update Invoice
6.6.1 Select the draft invoice for this CRM client
6.6.2 There are not draft invoices
6.7 Validation of the invoice
6.7.1 Denied invoice
6.7.2 Approved invoice
7 Possible Execution exceptions
8 More details
9 Worflow details
10 Annexes

4 DEFINE TOKENS FOR THE COMPANY

Only the Process responsible role can manage the different tokens of their company. She/he will click at the Manage menu—Tokens and the list of tokens appears.

CloudSocket

a7cea382-1ea6-407d-ae87-1c7031bba698_invoiceNinja

The customer can create, modify and delete their own tokens.

- **Create a token**: The customer clicks on the Create token button and fill the following data:

Figure 28 - Broker and Customer Handbook

After configuring the users, roles and the authentication of the different atomic services, the deployed workflow is ready to be used. Marc creates a new instance of the deployed workflow for the BPaaS bundle, as shown in Figure 29.

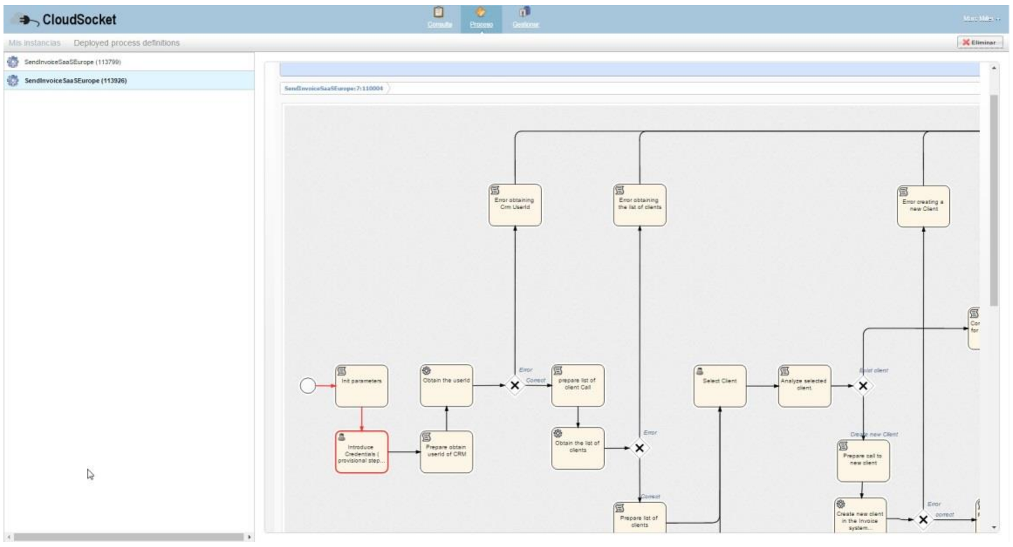
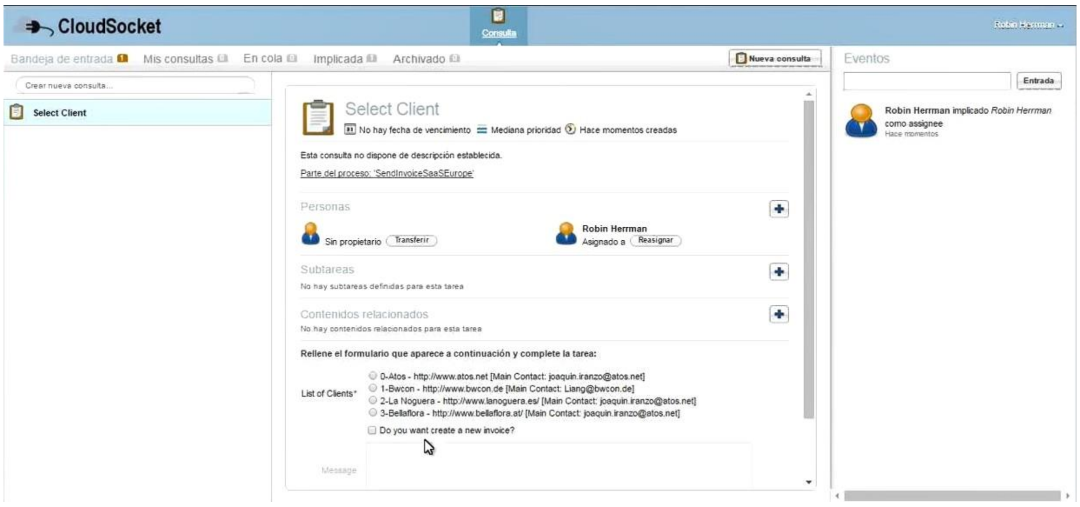

CloudSocket
Consulta
Proceso
Gestionar
Mis instancias
Deployed process definitions
Eliminar
SendInvoiceSaaSEurope (113799)
SendInvoiceSaaSEurope (113926)
SendInvoiceSaaSEurope:7:110004
Error obtaining Crm UserId
Error obtaining the list of clients
Error creating a new Client
Init parameters
Obtain the userId
Error
Correct
prepare list of client Call
Select Client
Analyze selected client.
Exist client
Introduce Credentials (provisional step...
Prepare obtain userId of CRM
Obtain the list of clients
Error
Create new Client
Prepare call to new client
Correct
Prepare list of clients
Create new client in the Invoice system...
Error
correct

Figure 29 - Creation of instance for the BPaaS bundle.

The workflow instance is started, and Robert, who belongs to the accounting representative group, receivers these associated user tasks, which are fulfilled by specifying the necessary input, such as the selection of the clients (collected from the CRM, Figure 30), the introduction of the invoice items (which are created automatically in the invoice system, Figure 31) and the confirmation of Susan who belongs to the supervisor group.



Select Client

No hay fecha de vencimiento · Mediana prioridad · Hace momentos creadas

Esta consulta no dispone de descripción establecida.

Parte del proceso: 'SendInvoiceSaaSEurope'

Personas

Sin propietario (Transferir)

Robin Herrman
Asignado a (Reasignar)

Subtareas

No hay subtareas definidas para esta tarea

Contenidos relacionados

No hay contenidos relacionados para esta tarea

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

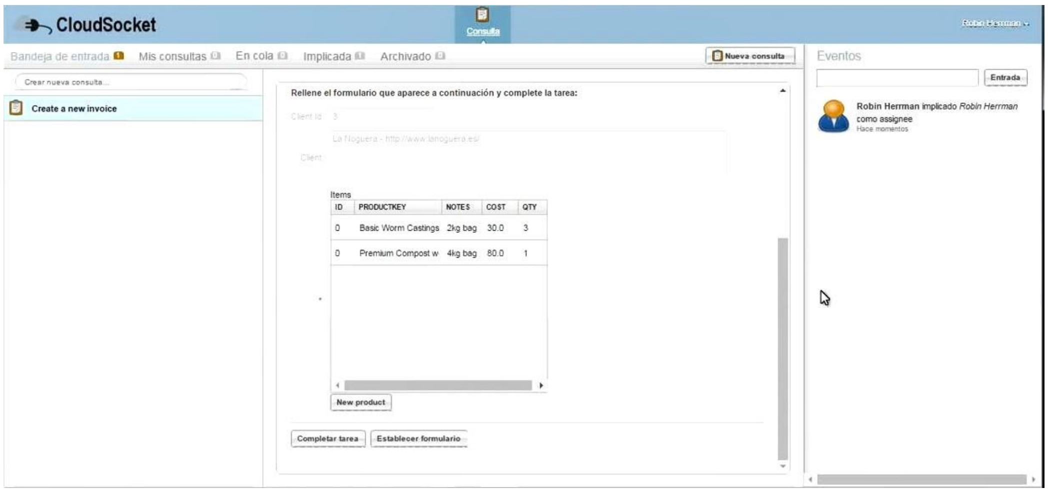
List of Clients*
- 0-Atos - http://www.atos.net [Main Contact: joaquin.iranzo@atos.net]
- 1-Bwcon - http://www.bwcon.de [Main Contact: Liang@bwcon.de]
- 2-La Noguera - http://www.lanoguera.es/ [Main Contact: joaquin.iranzo@atos.net]
- 3-Bellaflora - http://www.bellaflora.at/ [Main Contact: joaquin.iranzo@atos.net]

☐ Do you want create a new invoice?

Message

Robin Herrman implicado *Robin Herrman* como assignee
Hace momentos

Figure 30 - selection of the clients collected automatically from the CRM



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

Client Id 3

La Noguera - http://www.lanoguera.es/

Client

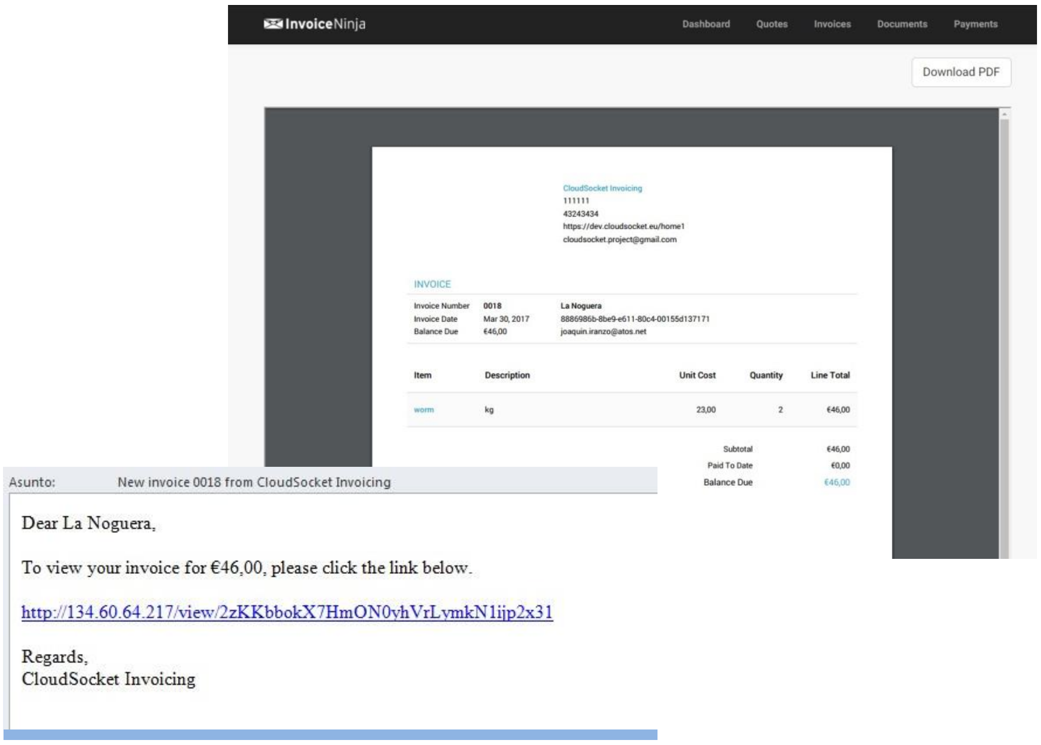
Items

ID	PRODUCTKEY	NOTES	COST	QTY
0	Basic Worm Castings	2kg bag	30.0	3
0	Premium Compost w	4kg bag	80.0	1

Robin Herrman implicado *Robin Herrman* como assignee
Hace momentos

Figure 31 - Introduction of the invoice, which is automatically synchronized with the invoice system.

Finally, the client of the Golden Worm Company receives an automatic email with the created invoice while the invoice system is automatically synchronised with the new invoices and their status (e.g. draft, sent) (Figure 32).



CloudSocket Invoicing
111111
43243434
https://dev.cloudsocket.eu/home1
cloudsocket.project@gmail.com

INVOICE

Invoice Number	0018	La Noguera
Invoice Date	Mar 30, 2017	8886986b-8be9-e611-80c4-00155d137171
Balance Due	€46,00	joaquin.iranzo@atos.net

Item	Description	Unit Cost	Quantity	Line Total
worm	kg	23,00	2	€46,00

Subtotal	€46,00
Paid To Date	€0,00
Balance Due	€46,00

Asunto: New invoice 0018 from CloudSocket Invoicing

Dear La Noguera,

To view your invoice for €46,00, please click the link below.

http://134.60.64.217/view/2zKKbbokX7HmON0yhVrLymkN1ijp2x31

Regards,
CloudSocket Invoicing

Figure 32 - Notification of the client and automatic invoice creation.

How the Golden Worm Company interacts with the Workflow Engine to manage the invoices is shown in the following video: https://youtu.be/-aZ4adXU9uA.

4.2.5 Service Level Agreement between Customer and Broker

When a bundle is bought through the Marketplace, the broker agrees to comply with a service level to the customer (in this case, BWCON and Golden Worm Company, respectively), in terms of execution of the bundle and execution of the workflow. This service level is expressed in the SLA tab of the bundle description on the Marketplace.

The users of the Golden Worm Company can check the status of the SLAs at the SLA Dashboard. This interaction is shown at the following video:

https://www.youtube.com/playlist?list=PLZVFNQ-78g4UUJM-_6qNRz5Dc1QIQK2uv

4.3 Required Environments

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 BPaaS 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) Authorisation (at service level) & Authentication (at user level).

To support the aforementioned functionalities, the BPaaS Marketplace comprises two main components, which are the following:

- Marketplace *(*yCONNECT*)* is the main entrypoint of the BPaaS customer to the CloudSocket platform. It allows the customers to discover, analyse and purchase a BPaaS bundle in the cloud environment.
- 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 by these environments and especially their components using standard web technologies.

Figure 33 shows the detailed scheme of the CloudSocket Reference Architecture and the location of the different components.

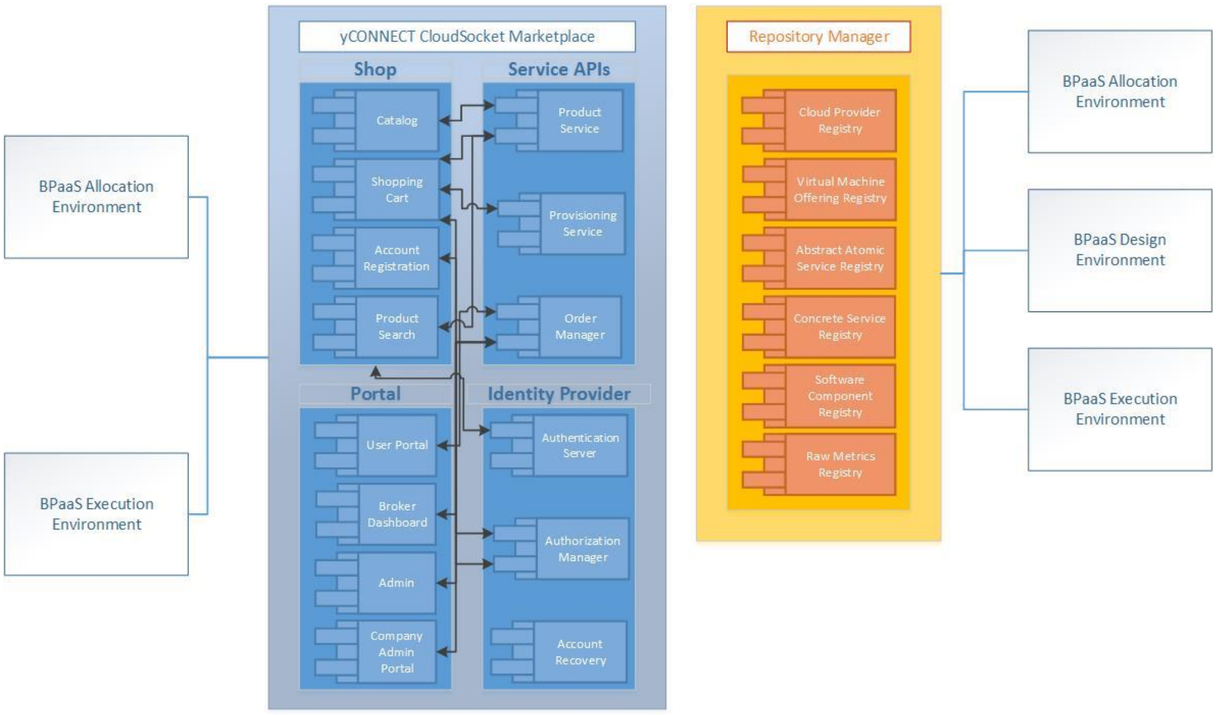


Figure 33 - Internal architecture for the BPaaS Marketplace

The BPaaS Marketplace components in each level interact with each other 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 [37].

4.3.1 BPaaS Marketplace

For BPaaS Customers, either SMEs or other organizations, the CloudSocket Marketplace is the main entrypoint to the CloudSocket platform and the offerings of a certain CloudSocket Broker. First of all, this Marketplace offers a place for registration. The registration process is ensured by the Account Registration component. Only registered users can buy BPaaS bundles.

Through the Catalog component, BPaaS Customers browse and view the details of published bundles. By using the Product Search component, BPaaS Customers can search through published bundles by tags, status of the bundle and price. By using the Shopping Cart component, BPaaS Customers can purchase the bundles. Moreover, by using the Payment Component, BPaaS Customers can pay for the selected component(s) in the shopping cart.

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 (Identity & Access Management)
- A graphical user interface for registered users that provides checkout capabilities for BPaaS as well as the non-shop related capabilities mapping to the User Portal component

The main functionalities offered by the Marketplace include:

- User and company registration through the Registration component. Brokers can register themselves through the portal, while companies and their users are registered by the broker.
- Available BPaaS bundle browsing through the Catalog component.
- Order management (list orders made by each BPaaS Customer, view corresponding invoices, etc.)
- BPaaS checkout through Shopping cart, Order Manager and Provisioning Service components
- Checked out bundles payment using PayPal or Bank Order
- User-related information management through the User Portal component
- Launching of deployment process from User Portal component (in the BPaaS Execution Environment) for those items that have been purchased by the BPaaS Customer
- Searching through published bundles by tags, status of the bundle and price

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 4 - Details of the Marketplace component

Comprises:

- Shop – main module that manages shopping experience, including checkout and payment
- Portal – module that manages non-shop related interactions of users (purchase history, account details, access provisioned items, reports and dashboards)
- Service API – module that provides data for the Web UI as well as the core BPaaS deployment triggering functionality (Execution Environment – provision endpoint)
- Identity Provider – module that ensures M2M and U2M authentication and authorization flows

Depends on:

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

Architecture design

The general architecture of this component can be viewed in grey, in Figure 33. Figure 34 introduces the deployment SOA model of the Marketplace.

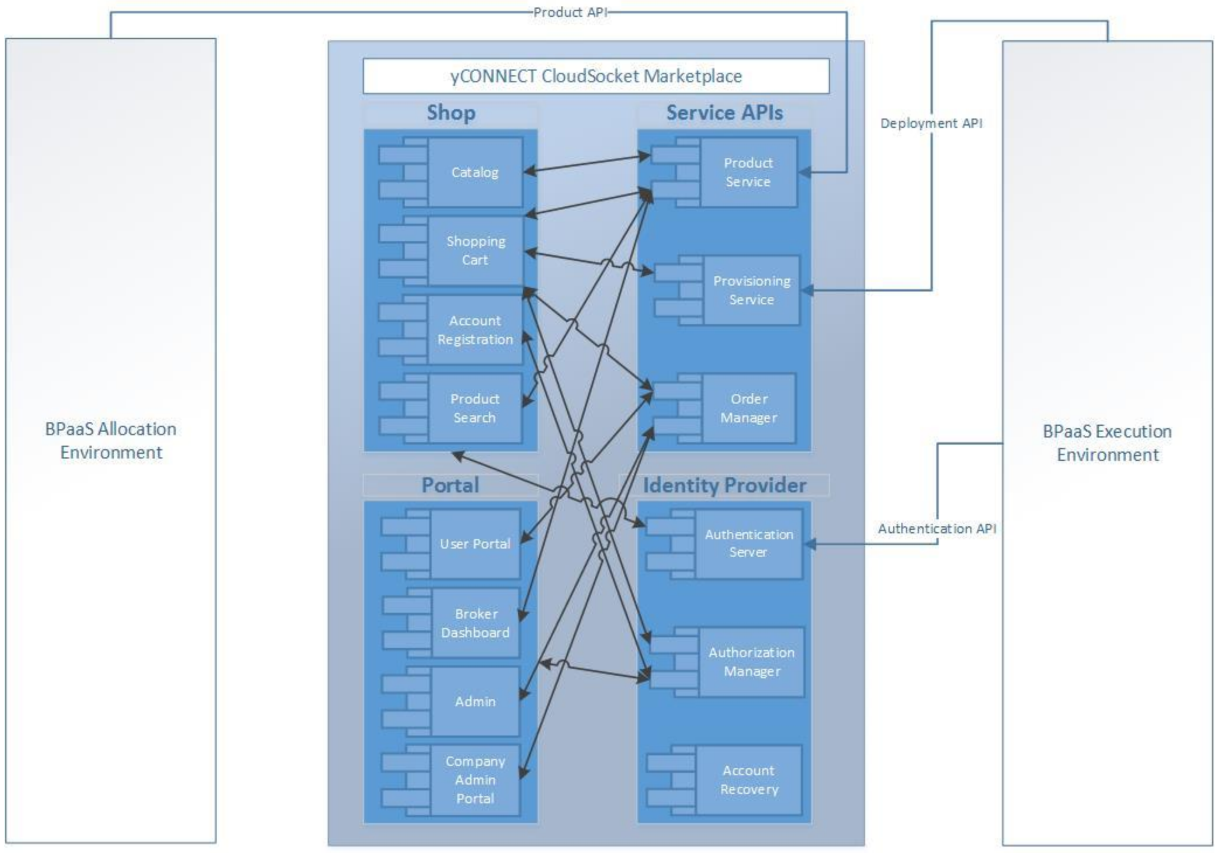


Figure 34 - Component diagram of the Marketplace

Deploy Diagram

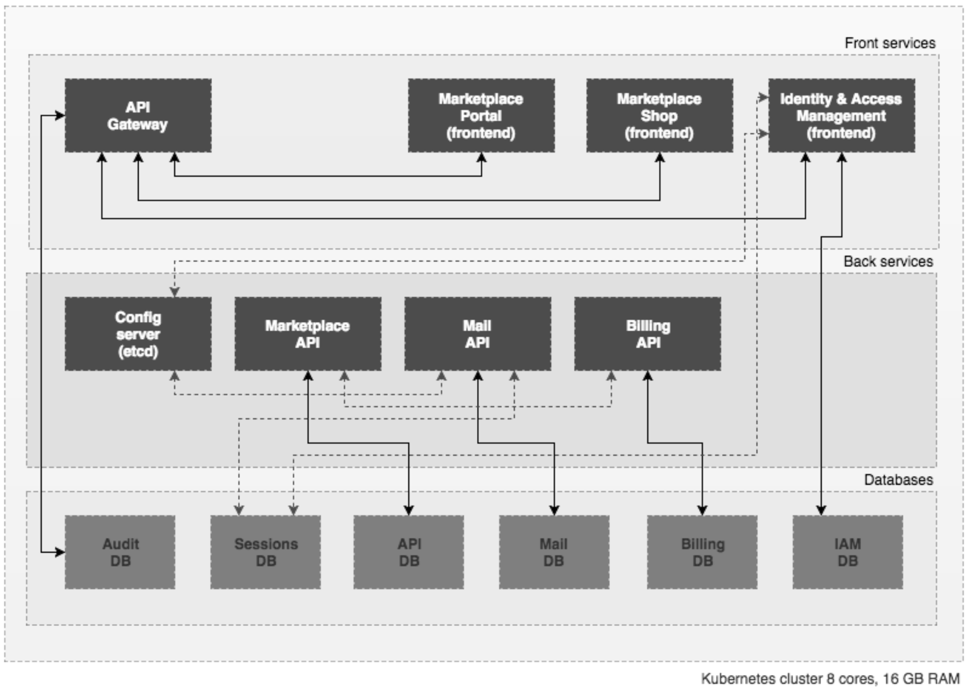


Figure 35 - Deploy diagram of the Marketplace

Shop and *Portal* components from the Figure 34 corresponds to *Marketplace Portal and Shop* component of the Front Services lane from the Figure 35. *Service APIs* component from the Figure 34 corresponds to *API Gateway* of the Front Services lane from the Figure 35. *Identity Provider* component from the Figure 34 corresponds to *Identity Provider Management* of the Front Services lane from the Figure 35.

Functionalities

Table 5 indicates the covered functionalities and their status:

Functionality	Description	Completed for release	Integrated (which level)
Shop	The module supplies the list of BPaaS published into the shop environment and allows registered users to checkout items (BPaaS offerings), to pay the checkout items using PayPal and Bank Order, as well as to search through such items by tags, status and price.	Yes (2nd release)	Complete lifecycle.
Service APIs	The module ensures API connectivity of the Marketplace in the CloudSocket value stream (Product API – used by Allocation Environment; Order Management API– used by Execution Environment for collection status for an order)	Yes (1st release)	Complete lifecycle
Portal	The module enables user to access to purchased BPaaS information as well as to shopping history visualisation and account information management. It also allows brokers to access a list of external tools, create customer companies and their users as well as give users access to different reports and statistics related to orders. Tools that can be accessed by brokers: Workflow Modeling, Allocation Tool, Raw Monitoring Data, Cloud Provider Engine, SLA Dashboard, Conceptual Analytics Engine.	Yes (2nd release)	Complete lifecycle
Identity provider	This module provides authentication and authorisation services for the Marketplace as well as for the entire CloudSocket value chain	Yes (1st release)	Complete M2M with Design Environment Complete M2M with Allocation Environment Complete M2M with Execution Environment Complete U2M with Execution Environment

Table 5 - Functionalities of the Marketplace

Manuals

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

Download

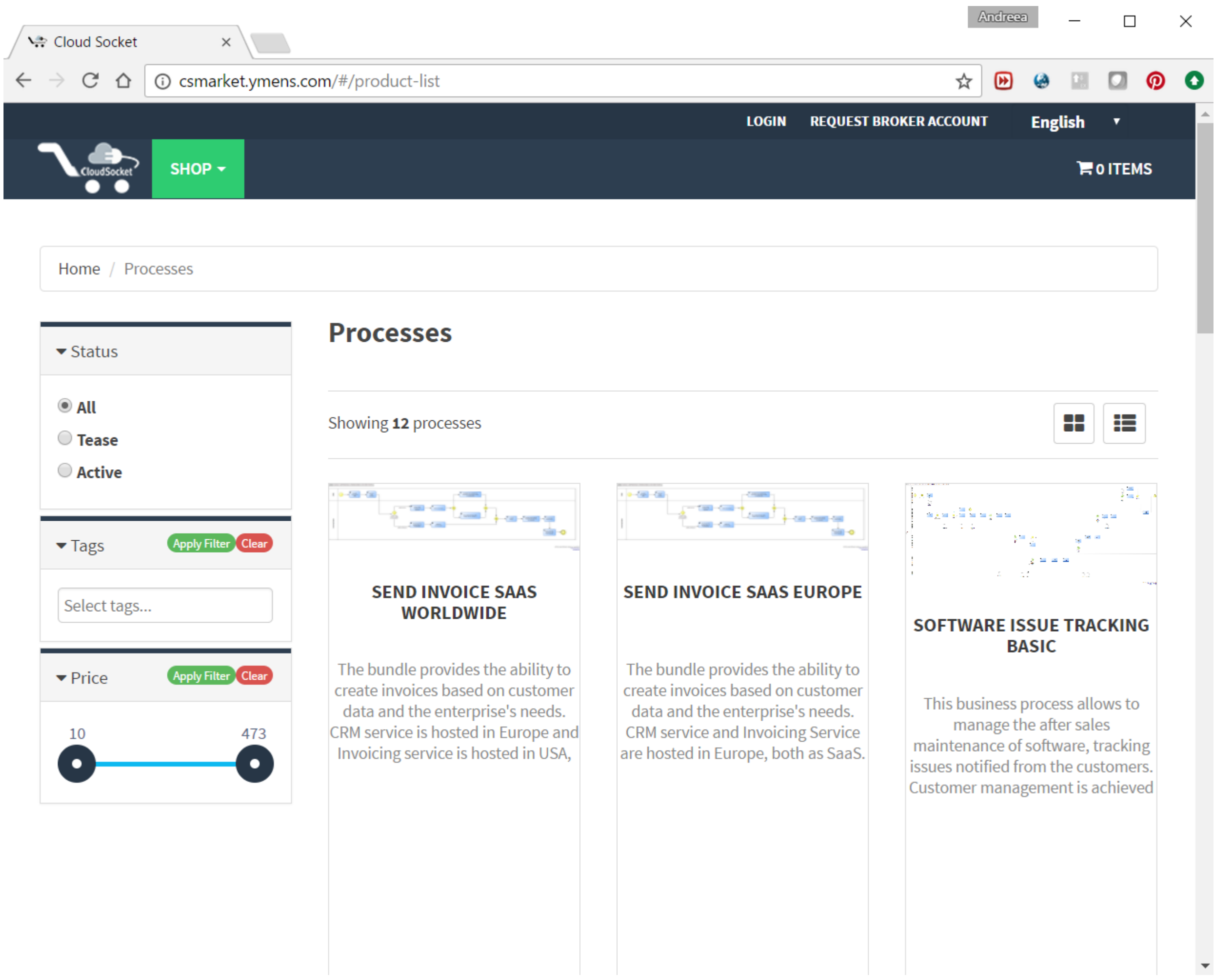
yCONNECT CloudSocket Marketplace is built on top of GRADY (Grails Rapid Application Development for Ymens) platform that is a collection of open tools for proprietary development.

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

Instances

There are two yCONNECT CloudSocket Marketplace instances:

- Demo, which is available at: http://csmarket.ymens.com
- Integration, which is available at: http://devmarket.ymens.com



Processes

Showing 12 processes

SEND INVOICE SAAS WORLDWIDE

The bundle provides the ability to create invoices based on customer data and the enterprise's needs. CRM service is hosted in Europe and Invoicing service is hosted in USA,

SEND INVOICE SAAS EUROPE

The bundle provides the ability to create invoices based on customer data and the enterprise's needs. CRM service and Invoicing Service are hosted in Europe, both as SaaS.

SOFTWARE ISSUE TRACKING BASIC

This business process allows to manage the after sales maintenance of software, tracking issues notified from the customers. Customer management is achieved

Figure 36 - Marketplace web page.

4.3.2 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. As such, it has a key role in the CloudSocket architecture, essentially as it is also exploited by all the CloudSocket Environments for accessing this information, for which now, there is only one structural level of registries offered.

The repository manager offers the information needed by the CloudSocket Environments by the means of a number of registries, which include the following:

- *Cloud Provider Registry*: it contains the information about a cloud provider like the name, the type of offerings that it provides (i.e., Platform as a Service or Infrastructure as a Service) and various 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 the Virtual Machine offerings provided by a specific Cloud Provider. The information recorded per Virtual Machine offering includes the number of cores, the size of the main memory and disk storage.
- *Platform as a Service Offering*: it contains the information about the Platform as a Service offerings provided by a specific Cloud Provider. The description of a Platform as a Service offering includes the number of cores, the size of the main memory, disk storage, middleware and services supported.
- *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 via 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), a list of instances for this service and an associated Abstract Service. Each service instance is 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 versions, 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 raw 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.

Table 6 shows which registries are accessed by which CloudSocket environment:

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

Table 6 - 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 7 - Details of the Repository Manager

Architecture design

The general architecture of this component can be viewed in yellow in Figure 33, while Figure 37 shows the Repository Manager's internal architecture.

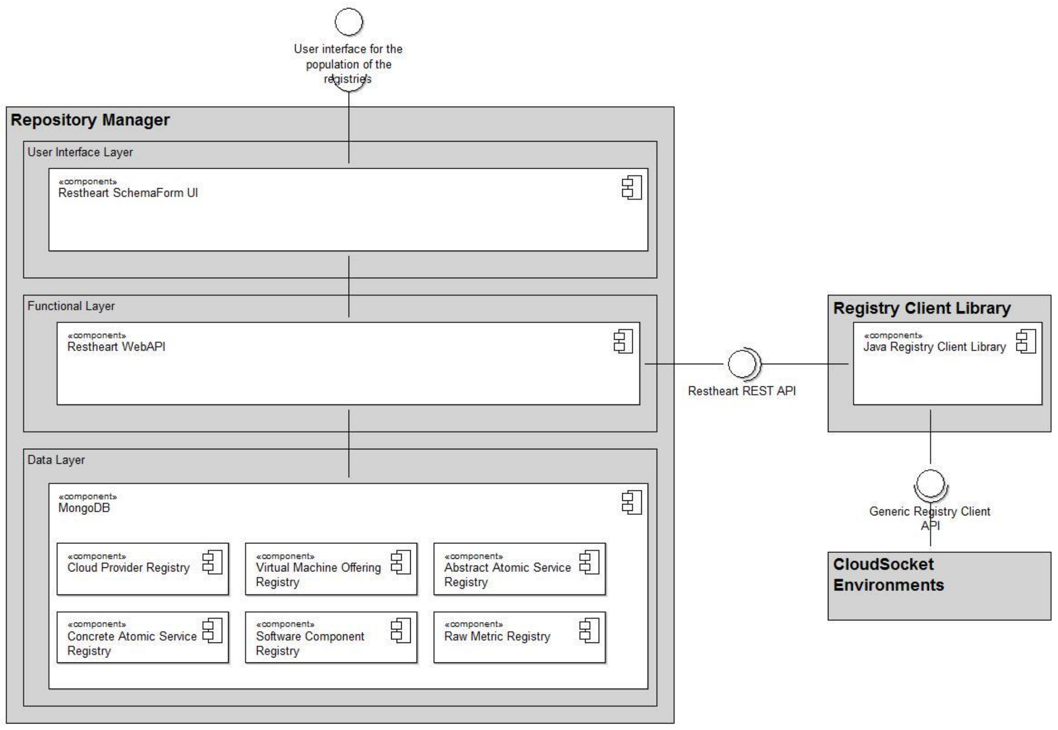

User interface for the population of the registries
Repository Manager
User Interface Layer
«component» Restheart SchemaForm UI
Functional Layer
«component» Restheart WebAPI
Data Layer
«component» MongoDB
«component» Cloud Provider Registry
«component» Virtual Machine Offering Registry
«component» Abstract 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 37 – Internal architecture of the Repository Manager

The Repository Manager follows a classic three-tier architecture pattern in which the user interface (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:

- User Interface Layer: There lies the Restheart SchemaForm UI, 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 arrays 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 also allows to create new kind of registries, both UI and data structure, 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.

- 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 following:

  - 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.

- Data Layer: In this 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. MongoDB [4] avoids the traditional table-based relational database structure in favour of JSON-like documents with dynamic schemas (MongoDB calls this 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's JSON documents. Each JSON document describes a different resource, while 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.

Functionalities

Table 8 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 objects into a NoSQL database.	Yes (1st release)	Complete lifecycle.
Create\Edit\Delete\Update registry objects	Restheart SchemaForm UI allows CRUD operations on each registry.	Yes (1st release)	Complete lifecycle.
Browsing registry objects	Restheart SchemaForm UI allows browsing of the registry objects. Registry Client Library is a Java client which allows the programmatically 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)	We didn't have the time and resources for this feature but it would be a good functionality to introduce in the future.

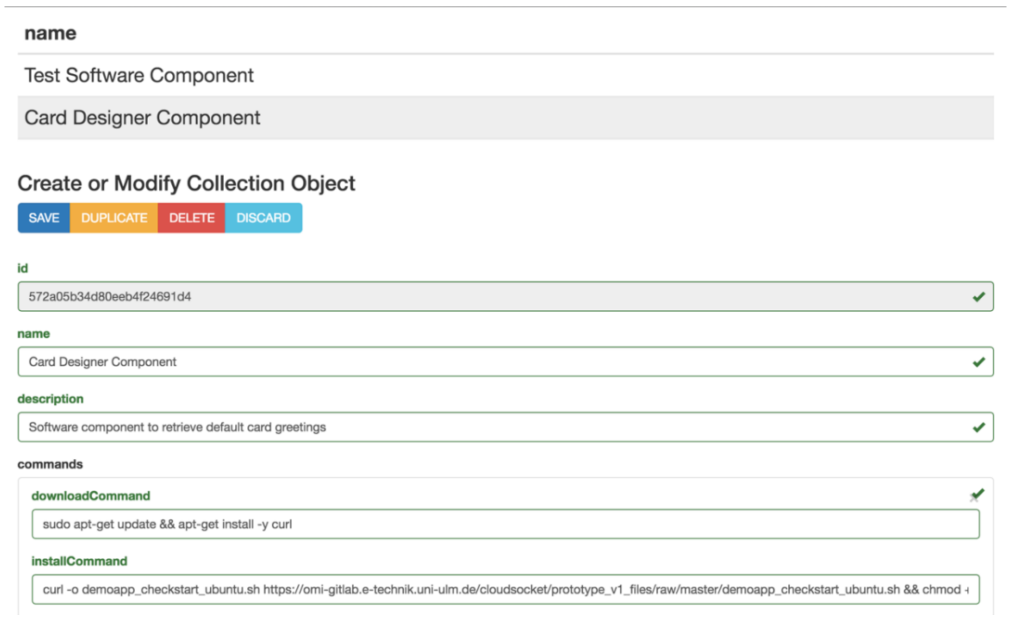
Table 8 - Functionalities of Repository Manager

Manuals

The manual, API description, handbooks are detailed in the alive wiki documentation: https://www.cloudsocket.eu/common-understanding-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 38 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 38 - 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/.

Credentials: available on demand.

4.3.3 BPaaS Execution Environment

The CloudSocket Execution Environment enables managing, monitoring and adapting the execution of the BPaaS bundles, which have been purchased in the Marketplace. The environment covers all the steps involved in the execution phase: (i) deploy and configure all components and monitoring sensors 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 the workflow of a BPaaS bundle has been deployed; (iii) visualise 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, which include:

- The *Workflow Engine,* which is responsible for deploying the executable workflows and executing the different workflow instances.
- The Cloud Provider Engine, which 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.
- The Monitoring Engine, which is responsible for cross-level BPaaS monitoring and for the correlation/aggregation of monitoring data coming from different levels, from infrastructure and software services up to the level of the workflow.
- The Adaptation Engine, which is responsible for the reconfiguration of the BPaaS deployment, possibly in a cross-level manner (e.g. by performing service replacement or component horizontal scaling), to resolve the problematic situations identified by adaptation rules.
- The *SLA Engine,* which is responsible for generating, storing and observing the formal documents describing electronic service-level agreements (SLAs) between the signatory parties (CloudSocket broker and BPaaS Customer).
- The *Process Data Mediator,* which 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 39 shows the detailed scheme of the internal architecture for the Execution Environment CloudSocket Reference Architecture, the location of the different components and their interactions with external components.

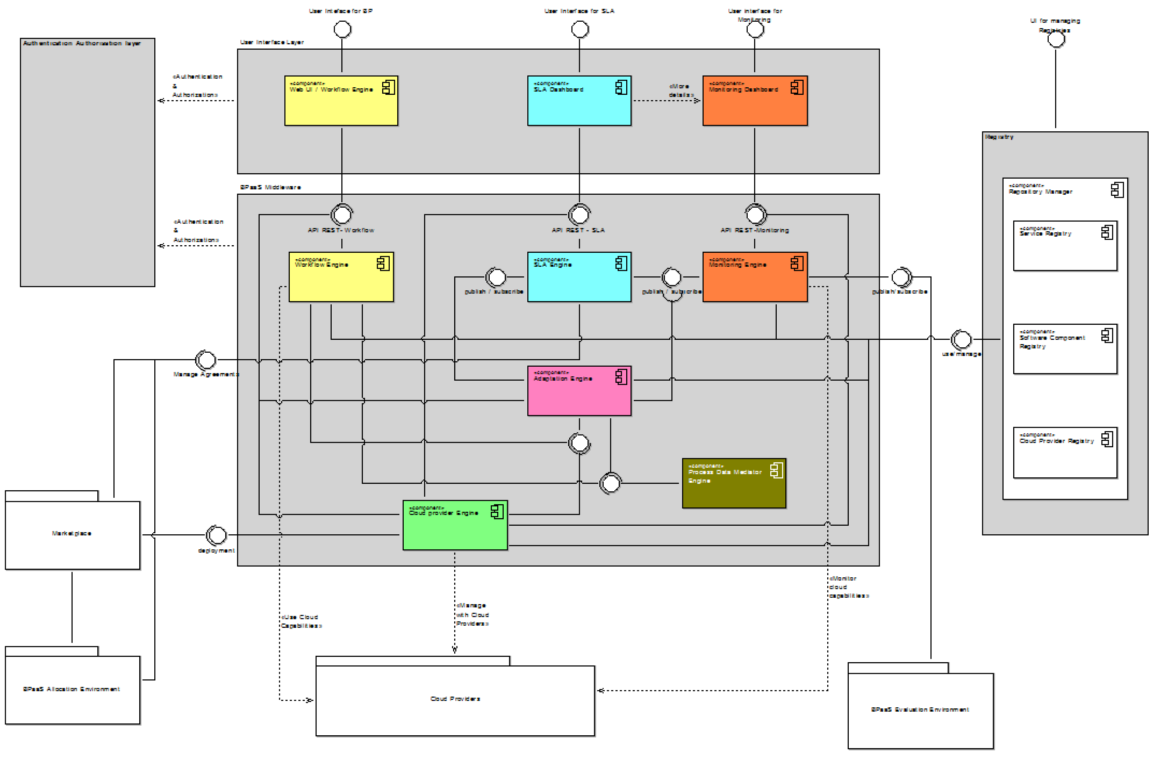

User Interface for BP
User Interface for SLA
User interface for Monitoring
UI for managing Registries
Authentication Authorization layer
User Interface Layer
«Authentication & Authorization»
«component» Web UI / Workflow Engine
«component» SLA Dashboard
«More details»
«component» Monitoring Dashboard
Registry
«component» Repository Manager
«component» Service Registry
«component» Software Component Registry
«component» Cloud Provider Registry
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

Figure 39 - 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 the Web interfaces of the Workflow Engine, the SLA dashboard and the Monitoring dashboard; (b) another including the core engines to cover the components 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, is not shown in Figure 39.

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 covered scenarios, can be found at the following URL: https://www.cloudsocket.eu/uml/3-ExecutionEnvironment/remotedocu/modeldocu/27012016104208/modelContentHTML, in which the corresponding UML diagrams (Figure 39) can be viewed. This information was also covered in the D4.2 deliverable [40].

In the following, we describe only those components that are part of this second 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.

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

4.3.3.1 Workflow Engine

It is responsible for managing the deployment, execution and management of the different workflow instances at the execution phase. This engine is multi-tenant, where tenant here means the customer organization that has purchased the BPaaS workflow. As such, it allows executing workflow instances on behalf of each tenant by also taking care of the corresponding workflow and organisation data level. Potentially it could also be exploited by the broker, who might act on behalf of a BPaaS customer, or the execution environment operator for administration purposes.

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 (i.e., start, suspend, resume, and stop) and monitor (the state of) the workflow instances, according to the workflow specification in BPMN.
- Interact with manual tasks of the workflow.
- Manage the workflow engine environment for multi-tenant, such as the defined roles, users and tenants, token access, 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 at the design phase (section 5.3.2.2). 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. The executable workflow is not deployed directly into the execution environment; it is necessary some actions to deploy it. Firstly, the allocation component (section 5.3.3) creates the BPaaS bundle for this executable workflow, which includes the deployment details for the services at several levels IaaS, PaaS and SaaS. When a bundle is defined, it can be published in the Marketplace. Then, the customer can purchase it and an automatic process is started by the cloud provider engine [ref] in order to orchestrate the deployment on cloud. Hence, the Cloud Provider Engine defines dynamically the self-contained executable when this workflow has to be deployed in the Workflow Engine.

The main roles to interact with the functionalities are:

- CloudSocket Customer, who wants to use the purchased BPaaS bundle(s), which has been deployed automatically in the cloud. The customer can interact via the use of different roles, which are defined in the Marketplace accounts: (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.
    - The Process Responsible will manage the different groups and internal roles associated to the business process, which are specific to the bundles. The management of internal groups have been delegated to an own component, avoiding to be covered at the level of the Marketplace accounts.

- 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 functionality of creating the executable workflows, which will be included in the BPaaS bundle during the design phase (section 5.3.2.2). 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 (section 2.5), who is responsible to designing the executable workflows using the Editor Workflow in order to include the necessary information to transform the abstract workflows (BPMN2.0 standard) to the executable workflows (including BPMN2.0 extensions to transform it to executable).

All the actors will interact directly with the graphical user interface and the system will manage automatically the authorisation of the respective functionalities invoked (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 9 - Details of the Workflow Engine

Comprises

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

Depends on

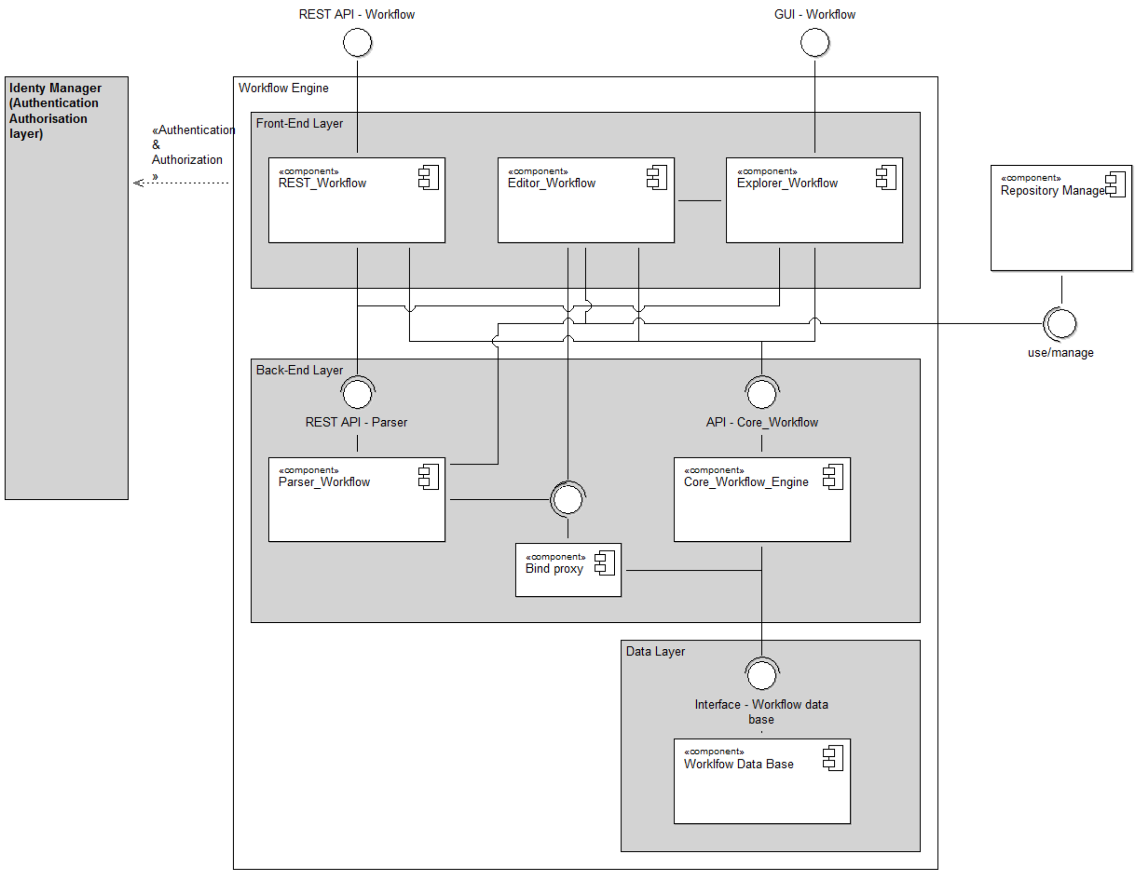
- IdM Marketplace
- Repository Manager

Architecture design

The general architecture of this component can be viewed by checking the components coloured in yellow, in Figure 39.

Figure 40 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 for providing all the functionalities and the respective data 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 corresponding actions needed.



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 40 - 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 4.3.3.3), which is responsible for orchestrating the deployment of the BPaaS bundle in the cloud environment, and the Adaptation Engine (section 4.3.3.4), which is responsible for modifying 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 actor is 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 executable workflows with the technology adopted in the workflow engine (extension of BPMN2.0 standard to cover the necessary executable attributes) in order to deploy and execute them correctly in the execution 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 & their instances, 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 for interacting with the data layer and for persisting 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, and 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 that 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 uses a persistence framework with support for custom SQL, stored procedures and advanced mappings; this allows to substitute the Workflow Data Base module by other

relational databases as long as the replacement database follows the standard connectors, since this persistence framework creates an abstraction layer between the entities and their persistence in the data base.

Functionalities

Table 10 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 engine environment	The component has to manage and follow up the complete workflow instances, the deployed BPaaS bundles, the groups, the tokens and the entities for the different tenants. It allows managing the engine	Yes (1st release)	Complete lifecycle
Design of Executable workflows	The component has to assist in the creation of the executable workflows, including the calls to the associated services.	Yes (2nd release)	Complete lifecycle
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.	Yes (2nd release)	Complete lifecycle. Integrated with the software component registry (section 5.3.2.2) and aligned with the semantic approach (section 3)
Monitor at the workflow level	The component has to provide the feedback for the defined raw metrics at the level of the workflow to the monitor engine.	Yes (2nd release)	Complete Lifecycle.

Centralized authentication and authorization	The component will delegate the authentication and authorization to an external Marketplace Identity Manager, following the standards (OAuth2 and SCIM). The functionality will manage the login and the logout of both the own component and also the Marketplace session.	Yes (2nd release)	Complete lifecycle.
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 component decides to modify a workflow execution to handle undesired situations. It is integrated with the Cloud Provider Engine (section 4.3.3.3) and the Adaptation Engine (section 4.3.3.4) to change the environment maintaining the quality of the services as a result of the research results (section 3)	Yes (2nd release)	Complete lifecycle.
Discover the most appropriate 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.	Partially (2nd release)	It is aligned with the research results (section 3)
Management of the customers and their groups by tenant	The Process Responsibles will manage the workflow usage of their company (tenant); moreover, they will assign the internal groups/roles for the different deployed bundles (workflows). On the other side, the component will synchronize automatically the users, the roles and the tenants of the Marketplace accounts.	Yes (2nd release)	Complete lifecycle
Internal user identification for the component.	The component will work with internal users to cover the platform operator role. It is necessary to have an alternative user identification based on the own component' data base in order to allow to manage the engine without depending on the external authentication. For example, if a problem occurs in the Marketplace IdM, it would impede to connect to the WFE and analyse the respective problematic situation.	Yes (2nd release)	Complete lifecycle

Table 10 Functionalities of the Workflow Engine

Manuals

The installation manual, the API description, unit test and handbooks are detailed in the alive wiki documentation https://www.cloudsocket.eu/common-understanding-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 also used [16] to cover the data layer.

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 and it is detailed in the D4.9 CloudSocket Integration Report [34].

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. Moreover, the automatic scripts to build and install the component, which are used in the continuous integration (T4.5), are available at: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/workflow-meta/tree/master (a UULM GitLab account is required in order to access the resources):

- run_build.sh is responsible for building and creating all the necessary artifact (see Figure 41),
- run_install.sh file is responsible for installing all the necessary software in an empty virtual machine.

Figure 41 indicates the generated artefacts and their dependencies.

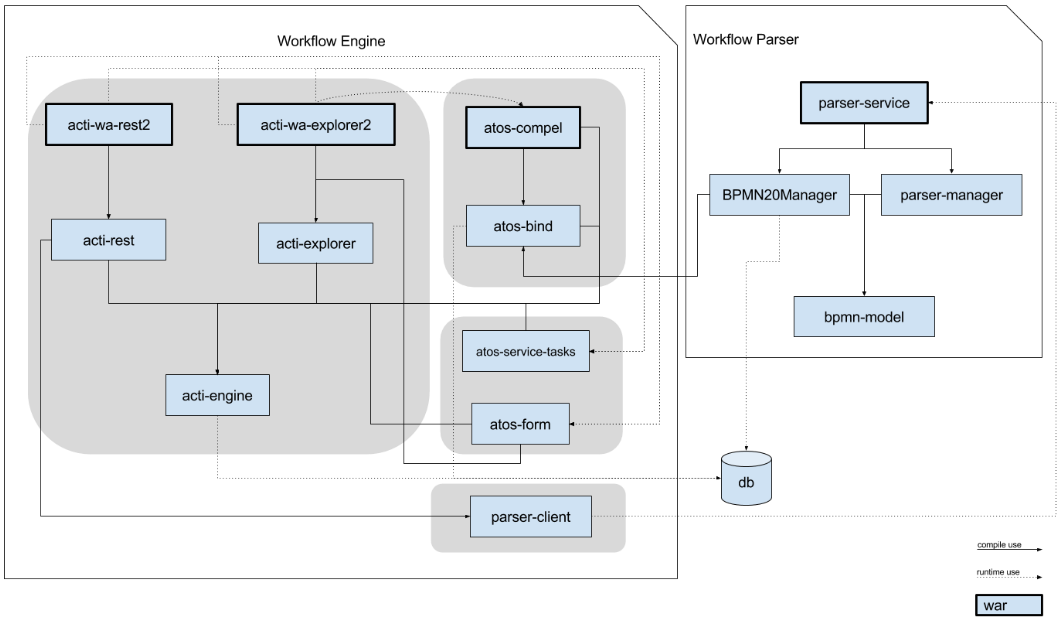


Figure 41- Artefacts of the Workflow Engine.

The automatic integration testings are also included in the GitLab repository https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/workflow-engine/tree/master/IntegrationTests (a UULM GitLab account is required in order to access the resources). These tests (deploymentTest.sh and adaptationTest.sh) cover the complete API functionalities to confirm that the behaviour is correct and the component works correctly with the Cloud Provider Engine.

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

- API specification: RESTful Interface is described to indicate the header, the method, the JSON structure of the request and the response. Besides, the associated unit tests plus the automatic integration testing for the continuous integration are included.

- Handbook is responsible for explaining the covered functionalities through the screenshots and description of the interface.
  - Handbook WFE is focused on the users of the component; hence the graphical user interface and the behaviour of these functionalities are detailed such as the different ways to be identified in the WFE (Figure 42).
  - Developer manual is focused on the developers; hence it is detailed how to create and work with the development environment and how to create and manage the different pieces of software, such as the custom fields for the workflow forms, the integration with the oAuth provider and so on.

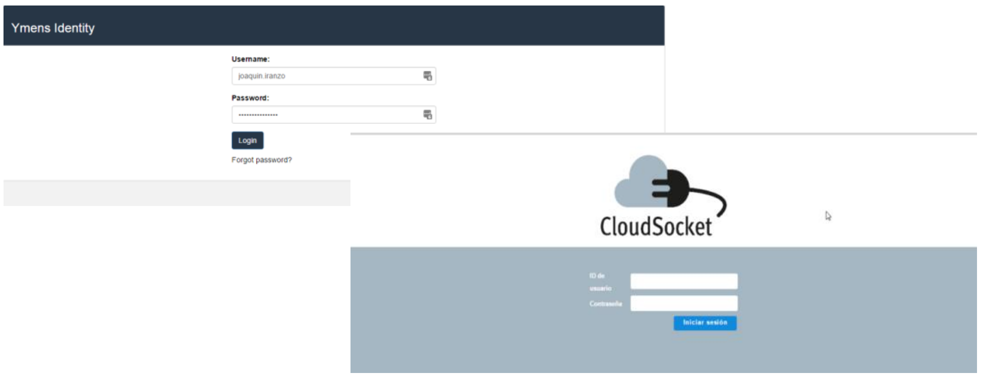


Figure 42 – OAuth2 and internal authentication of the Workflow Engine.

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: 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.

Credentials: A UULM GitLab account is required in order to access the resources.

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

Instances

There are different instances deployed on the UULM infrastructure to follow the continuous integration definition [34] for the second prototype. We have defined two environments: Integration and Demo. All the instances are published as a SaaS service for multitenancy in the cloud and they are completely integrated with the rest of components.

Integration Environment

- http://134.60.64.173/activiti-explorer/: this instance is used for testing the new releases and the research prototypes, before moving to the demo environment.

Demo Environment

- http://134.60.64.132/activiti-explorer: This instance is used by the brokers and the customers to manage the deployed bundles ("execution phase"). It is stable and only releases that have passed the complete testing, can be deployed on the demo environment.
- http://134.60.64.202/activiti-explorer/. This instance is responsible for designing and testing the executable workflows ("design phase"), before moving to the allocation phase. The artifacts are the same; however the designing of workflows consumes many resources (designing and testing the workflows) and is better to be isolated from any other instance that takes care of user/broker management of the workflows. In these terms, in order to also anticipate increased testing and load in usage, the component took the decision to have two instances of the engine, instead of one.

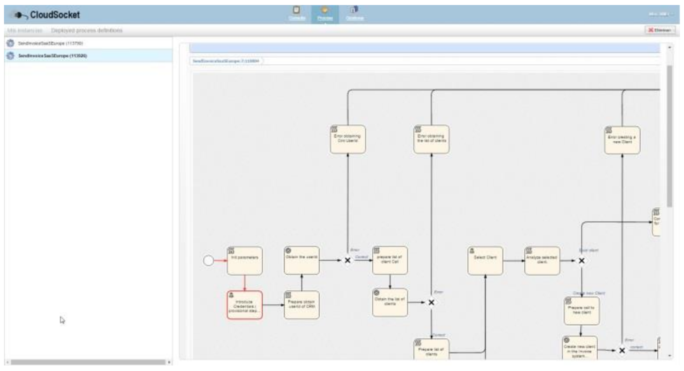


Figure 43 – Dashboard of the WFE.

4.3.3.2 Monitoring Engine

It is responsible to monitor a BPaaS and correlate/aggregate monitoring data from different levels, i.e., from the infrastructure and software service level up to the workflow level. 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 (assigned) 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 covers both raw metrics (direct measurements provided by deployed sensors or external measurement systems like PaaS providers) and aggregated metrics (formulas to exploit metrics are already implemented and produce the respective aggregated measurements).

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 REST API offered by Cloud Provider Engine and the TSDB engine (incorporated in the Monitoring Engine). This enables it to pull the monitored information stored on demand at periodic time intervals.

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);
- A Telnet interface for user-specific metrics, which are reported by custom sensors.
- 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 (system-based and user-specific sensors).
- Allow implementation and description of system-based and user-specific sensors;
- Monitor and aggregate metrics;
- Metric specification modification as well as new metric definition leading to changing the monitoring infrastructure/environment for an existing 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.

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 11 - Details of the Monitoring Engine

Comprises

- REST interface, also wrapped and used by the Cloud Provider Engine - see Section 4.3.3.3 (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

- 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 realise the Cloud Provider Engine. We will go into detail for this component in the architecture section of the Cloud Provider Engine - see Section 4.3.3.3.

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

Table 11 shows the status of the Monitoring Engine features with respect to the 1st and 2nd prototype.

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 (2nd release)	Distribution of raw monitoring data supported but not for composed monitoring data
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	Yes (2nd release)	Complete lifecycle
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 the levels of platform, software service and workflow	Partially (2nd release)	Basic sensors are available,e.g. availability sensor
Metric public/subscribed mechanism	Provide a way to subscribe to metrics	Yes (1st release)	Complete lifecycle
Third-party monitoring tool integration	Push interface to integrate third-party monitoring tools	Yes (1st release)	Complete lifecycle
Interfacing with the SLA Engine	Extend the reporting tool to create measurements in a compatible format for the SLA Engine	Yes (2nd release)	Complete lifecycle

Monitoring data storage abstraction	Abstract the reporting of monitoring data to support different storage back ends, such as TSDBs or log systems in order to, e.g. realize fault-tolerance	Yes (2nd release)	Support for InfluxDB available, support for Druid and Prometheus is ongoing

Table 12 - Functionalities of the Monitoring Engine

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: https://github.com/cloudiator/visor.git

The components are released here under the Apache 2.0 license.

Instances

As the Monitoring Engine is coupled with the Cloud Provider Engine, both are deployed on the same VM. In the scope of the BPaaS Bundle deployment the monitoring agent is deployed on each virtual machine hosting a software component. The monitoring agents gather the measurements. The REST interface of the Cloud Provider Engine is used to configure the Monitoring Engine. The Monitoring Engine instances for the Demo and Integration environments of CloudSocket are at the OpenStack cloud of UULM and are available at the following URLs: Demo Environment http://134.60.64.155:8080, Integration Environment http://134.60.64.166:8080.

Credentials: Available on demand

4.3.3.3 Cloud Provider Engine

This component is responsible for the complete deployment and lifecycle management of all the required components of a BPaaS, including software components, monitoring sensors and VMs across multiple clouds. These capabilities are 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 builds upon existing functionalities offered through the interfaces exposed by the cloud providers.

The BPaaS bundle deployment is managed by the Deployment Engine, namely Cloudiator [35], responsible for deploying, configuring and operating all appropriate software and VM components before deploying a workflow in the Workflow Engine. In essence, Cloudiator is responsible for executing the deployment plan included in the BPaaS bundle by orchestrating all necessary steps. This deployment plan, comprising the CAMEL model, the workflow file and SLA templates, covers the component deployment, SLA template 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 failure semantics, in terms of retrying the BPaaS Bundle deployment, takes place.

The Cloud Provider Engine will expose three interfaces: (a) an interface to enable performing re-deployment actions in order to interact with the Adaptation Engine component, (b) an interface to interact internally with the Deployment Engine for managing deployment, and (c) a web UI for the operators providing technical deployment information and advanced management capabilities.

Cloudiator sub-component comprises different plug-ins to connect to the different IaaS and PaaS 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 exposes:

- A REST interface to configure the deployment onto different cloud providers.
  - Input: BPaaS Bundle:
    - BPMN file (deployable workflow).
    - SLA agreement definition (including Service Level Objectives with conditions over quality metrics).
    - List of adaptation rules to drive the adaptation behaviour of the BPaaS Bundle.
    - Deployment plan (concrete deployment model in CAMEL), where further deployment actions are to be executed in the context of adaptation rules, which are controlled by the Adaptation Engine.
  - Output: The status and result of the current deployment plan can be viewed constantly on run-time.
- A REST interface to reconfigure the service instances during run-time.
- A web UI for the operator to perform advanced management operations (Figure 20).

The main functionalities are:

- Manage the complete BPaaS deployment.
- Manage and abstract from current IaaS capabilities and 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 13 - Description of the Cloud Provider Engine

Comprises:

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

Depends on:

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

Architecture design

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

The main part is the Colosseum component, which is the entry point to the framework which also 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 44 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 for components (via the use of a broker). Axe consists of the Adaptation 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. While Sword covered IaaS providers in the 1st prototype, in the 2nd protoype PaaS providers are covered as well by integrating the PaaS orchestration research prototype.

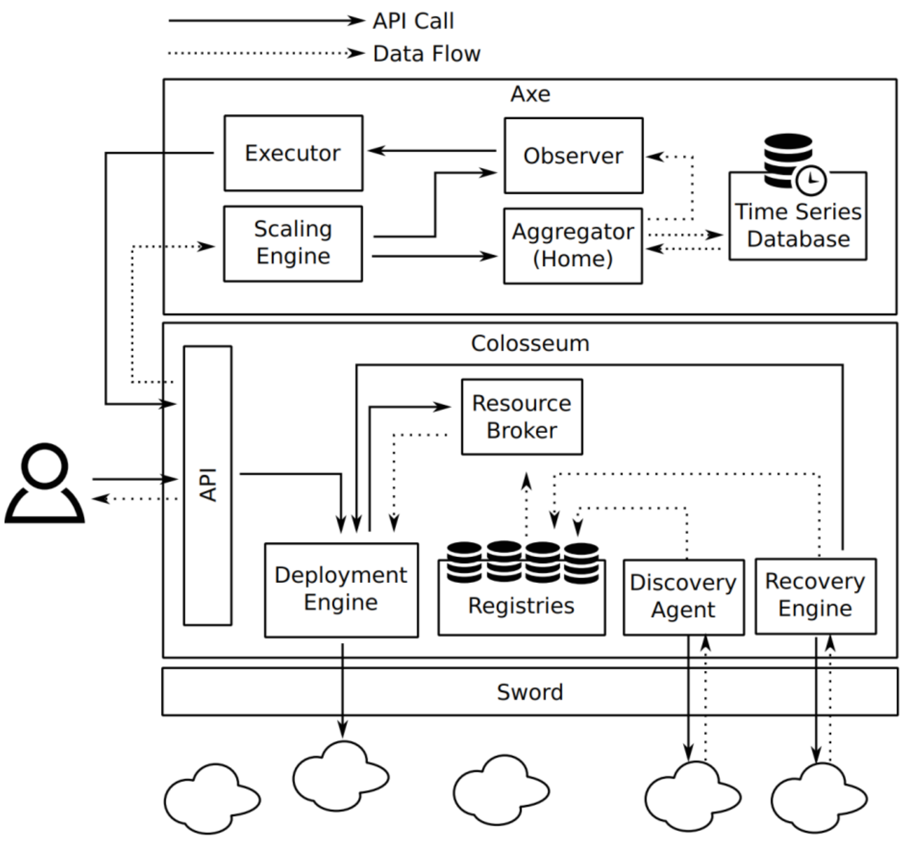


Figure 44 - Internal architecture for the Cloudiator framework of the home domain.

The remote domain is deployed along with the deployment of any virtual machine. Figure 45 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 scaling actions or stores aggregated values into a local TSDB.

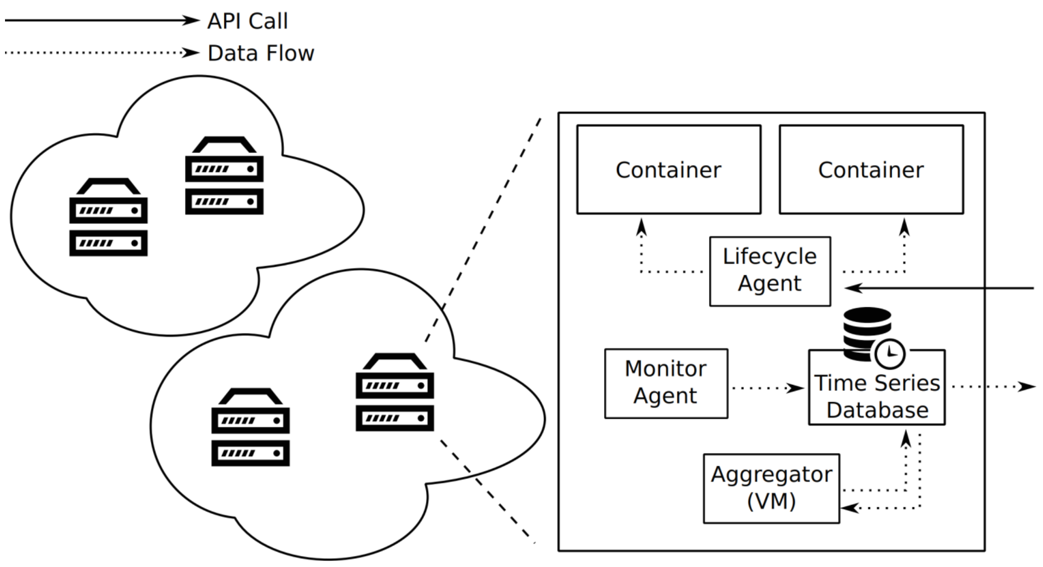


Figure 45 - Internal architecture for the Cloudiator framework of the remote domain.

Functionalities

Table 14 highlights the functionalities of the Cloud Provider Engine with respect to the 1st and 2nd prototype.

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)	Complete Lifecycle
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 level. For each level an abstraction layer should be provided.	Yes (2nd release)	Complete lifecycle, integrated Paas orchestration prototype [32]
Complex Life Cycle Actions	In addition to the generic imperative life-cycle actions, the Cloud Provider Engine should be open to more complex actions, i.e. a sequences of adaptation actions	Yes (2nd release)	Available in the research branch/environment [32]
Integration of well-known DevOps tools	The application component configuration should be additionally specified via Chef recipes or Puppet modules	partially (2nd release)	Custom Docker images are supported, further DevOps support is under development
Simplified access to the log files	The user should have an easy and supportive way to access the logs	Yes (1st release)	Complete lifecycle
Web GUI	In addition to the REST interface, a GUI should support the user towards the management of the components & VMs deployed	Yes (2st release)	Complete lifecycle
Interfaces for Adaptation Engine	Extended API to offer adaptation actions (on IaaS and PaaS level) and their analysis on the respective resource level	Yes (2st release)	PaaS adaptations are supported, cross-layer adaptations are enabled in the research add-ons [32]
Extended IaaS provider support	Supporting the deployment on the FIWARE infrastructure	Yes (2st release)	Complete lifecycle

Table 14 - Functionalities of the Cloud Provider Engine

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/common-understanding-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 source code for the Entry-point, as a wrapper for the Cloudiator toolset, is located at the GitLab of UULM and can be downloaded under: https://omi-gitlab.e-technik.uni-ulm.de/cloudsocket/execution-environment-simple-entrypoint.

Credentials: A UULM GitLab account is required in order to access the resources.

Instances

The Cloud Provider Engine, with the components entrypoint and colosseum, is deployed on separate VMs in the UULM OpenStack for each of the CloudSocket Environments as follows:

Demo environment:

Entrypoint: http://134.60.64.155:9012/job Colosseum: http://134.60.64.155:9000

Integration environment:

Entrypoint: http://134.60.64.166:9012/job Colosseum: http://134.60.64.166:9000

Cloud Provider UI:

Demo: http://134.60.64.155/cloud-provider-engine-ui

Integration/research: http://134.60.64.166/cloud-provider-engine-ui

Credentials: Available on demand

4.3.3.4 Adaptation Engine

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

The management of the adaptation rules is covered by a subcomponent called Rule Engine, which is responsible for taking decisions based on the environment variables and the performance levels encountered. So, it interacts 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 are 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 a sub-component of the Cloud Provider Engine, as it possesses the knowledge of the adaptation rules (covering scalability) and will 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 performs 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 are described in a form of a workflow which also dictates 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;
    - allow executing normal as well as personalized/customized adaptations.
  - Referencing of the actual adaptation workflow to allow its instantiation when a respective adaptation need arises.
  - Keep track of the result of the executed adaptation.

The main functionalities are:

- 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 15 - Description of the Adaption Engine

Comprises

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

Depends on

- Cloud Provider Engine
- Monitoring Engine
- Workflow 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 44.

Functionalities

Table 16 highlights the functionalities of Adaptation Engine with respect to the 1st and 2nd prototype.

Functionality	Description	Completed for release	Integrated (which level)
Basic Scaling Functionality	Provide a way to describe and enact basic threshold-based rules to trigger horizontal scaling actions.	Yes (1st release)	Complete lifecycle
Event / Measurement Registration	Register to Monitoring to obtain events for the evaluation of adaptation rules	Yes (1st release)	Complete lifecycle
Obtain and realise adaptation rules	Expose interface to retrieve adaptation rules from the Cloud Provider Engine, map events to adaptation rules and start their execution	Partly (2nd release)	Available as research add-on [32]
Adaptation plan	Provide a sophisticated way to define and enact the workflow of an adaptation action and its interdependence	Partly (1st release)	Available as research add-on [32]
Migration of stateful services	Ability to migrate stateful service to another cloud provider	Partially (2nd release)	Services need to provide an data transfer interface

Table 16 - Functionalities of the Adaption Engine

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 from GitHub. It features also an own branch for the CloudSocket-specific features, which are not needed for the general purpose of the Cloudiator project: https://github.com/cloudiator/axe-aggregator

The components are released here under the Apache 2.0 license.

Instances

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

Demo Environment:

http://134.60.64.155:9000/api/composedMonitor

http://134.60.64.155:9000/api/componentHorizontalOutScalingAction

http://134.60.64.155:9000/api/componentHorizontalInScalingAction

Integration Environment:

http://134.60.64.155:9000/api/composedMonitor

http://134.60.64.155:9000/api/componentHorizontalOutScalingAction

http://134.60.64.155:9000/api/componentHorizontalInScalingAction

Credentials: Available on demand

4.3.3.5 SLA Engine

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[2]. In CloudSocket, a BPaaS bundle includes the definition of an SLA template. The SLA template defines the service level to be exhibited by a BPaaS once it has been instantiated. Such an instantiation generates an SLA agreement, based on the SLA template of the BPaaS bundle, which is implicitly accepted by the BPaaS customer when the BPaaS bundle is bought. Several BPaaS Bundles may be generated from a single BPaaS Design Package (see Section 5.3.2 – BPaaS Design Environment), which include different SLAs with possibly different service levels and different cloud services being involved.

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 that specification. The specification has been extended where needed to cover some specific CloudSocket needs (see SLA Engine manuals)

This component exposes:

- A graphical user interface for checking the status of agreements,

[2] The SLAs between CloudSocket Broker and BPaaS Customer finally have not supported, but an analysis of such functionality can been found in the SLA Engine wiki.

- 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 generation of corresponding penalties when an SLO is violated.
- Notification of detected violations and incurred penalties to interested parties.

Only the graphical user interface is intended to be used by human users. These users can be employed by a:

- BPaaS Customer: such a user is employed by the BPaaS customer organisation, in order to check the SLAs of the BPaaS bundles purchased by the organisation.
- CloudSocket Broker: such a user is employed by the CloudSocket Broker in order to check the SLAs of the BPaaS bundles that have been purchased by the broker's customers.

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. [REF]
Reference community	Atos Research & Innovation (ARI) developers
Lead Partner	ATOS

Table 17 SLA component details

Comprises:

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

Depends on:

- Monitoring Engine
- Allocation Environment
- Marketplace
- Cloud Provider Engine

Architecture design

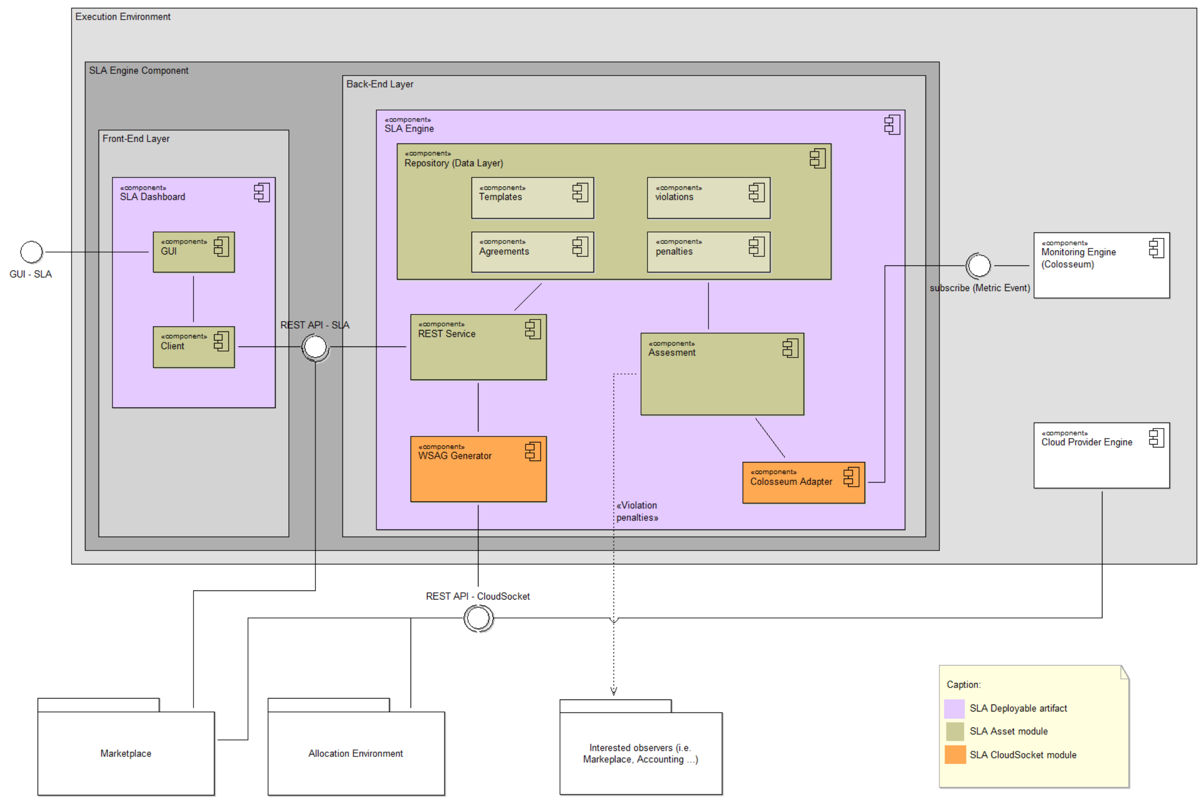


Figure 46 – SLA Framework architecture

The general architecture of this component can be viewed in Figure 46. The SLA Management in CloudSocket comprises two main components, and an optional Time Series Database:

- SLA Core. It relies on the Atos SLA Framework asset, with some CloudSocket-related modules. It exposes all the aforementioned 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 offered to the rest of the CloudSocket platform.
    - WSAG Generator. It generates WSAG-compliant templates according to the requirements of BPaaS Allocation Environment.
    - Colosseum Adapter. It is in charge of: (i) subscribing in the corresponding publish/subscribe mechanism of Colosseum for appropriate metric events; (ii) receiving events and translating 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.
- TimeSeries Database. An optional Time Series Database (currently, a KairosDB) stores the metric values for an agreement – received from the Monitoring Engine – and the corresponding violations if they apply. When the TimeSeries Database has been configured, this stored information can be later visualized in the SLA Dashboard.

The following CloudSocket components depend on the SLA Engine or SLA documents:

- Monitoring Engine: it reports the metric data to the SLA Core.
- Allocation Environment: it sends the SLA properties of a BPaaS bundle to the SLA Generator in order to obtain a template that describes the service level of a BPaaS Bundle and the penalties to be applied in case of violation of a service level.
- Marketplace: it shows a human-readable description of a BPaaS Bundle SLA to the customers on the BPaaS Bundle details page.
- Cloud Provider Engine: it provides the SLA to be stored and managed by the SLA Engine.

The sequence diagram in Figure 47 shows the relationships between all these components and the generation and states of SLA templates and SLA agreements. At design time, an SLA template is generated for a BPaaS Bundle, and included into it. When a BPaaS Bundle has been purchased, the Cloud Provider Engine communicates the deployment to the SLA Engine. The notification triggers the generation of the WSAG Agreement – based on the corresponding template - between the CloudSocket Customer and the CloudSocket Broker for this BPaaS Bundle, and the start of the SLA assessment. To perform the SLA evaluation, the SLA Engine subscribes to the appropriate events on the Monitoring Engine. At runtime, the received monitoring events are used to check the BPaaS Bundle instance satisfies the SLOs, raising a violation, which can imply a penalty, when an SLO is not satisfied.

In Figure 48 there is a screenshot of the SLA Dashboard showing the results of an agreement assessment.

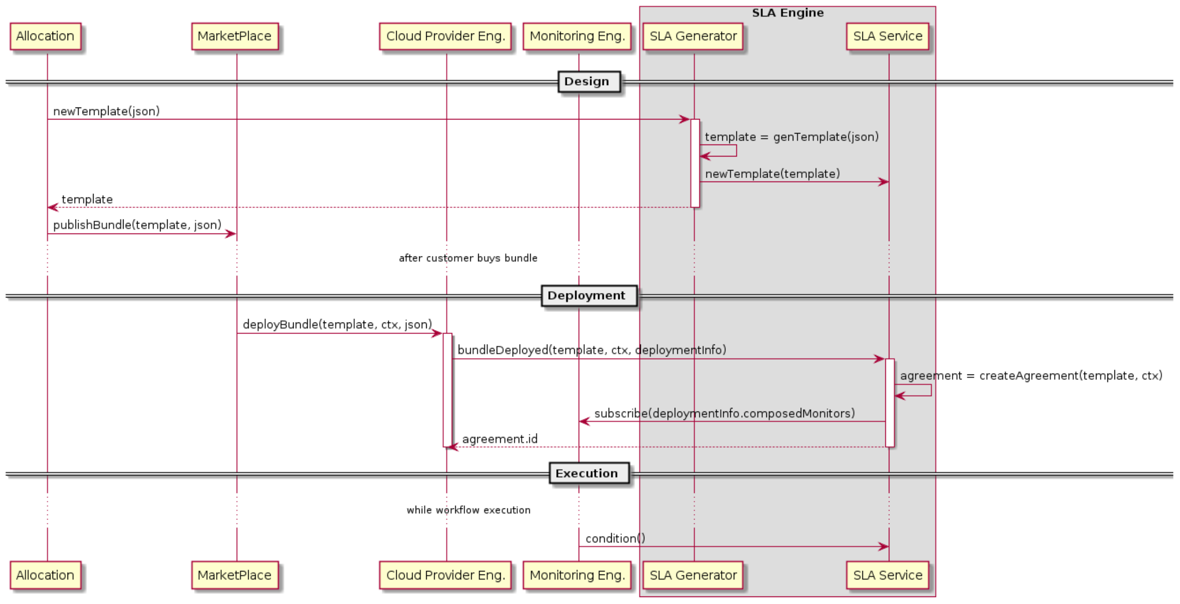
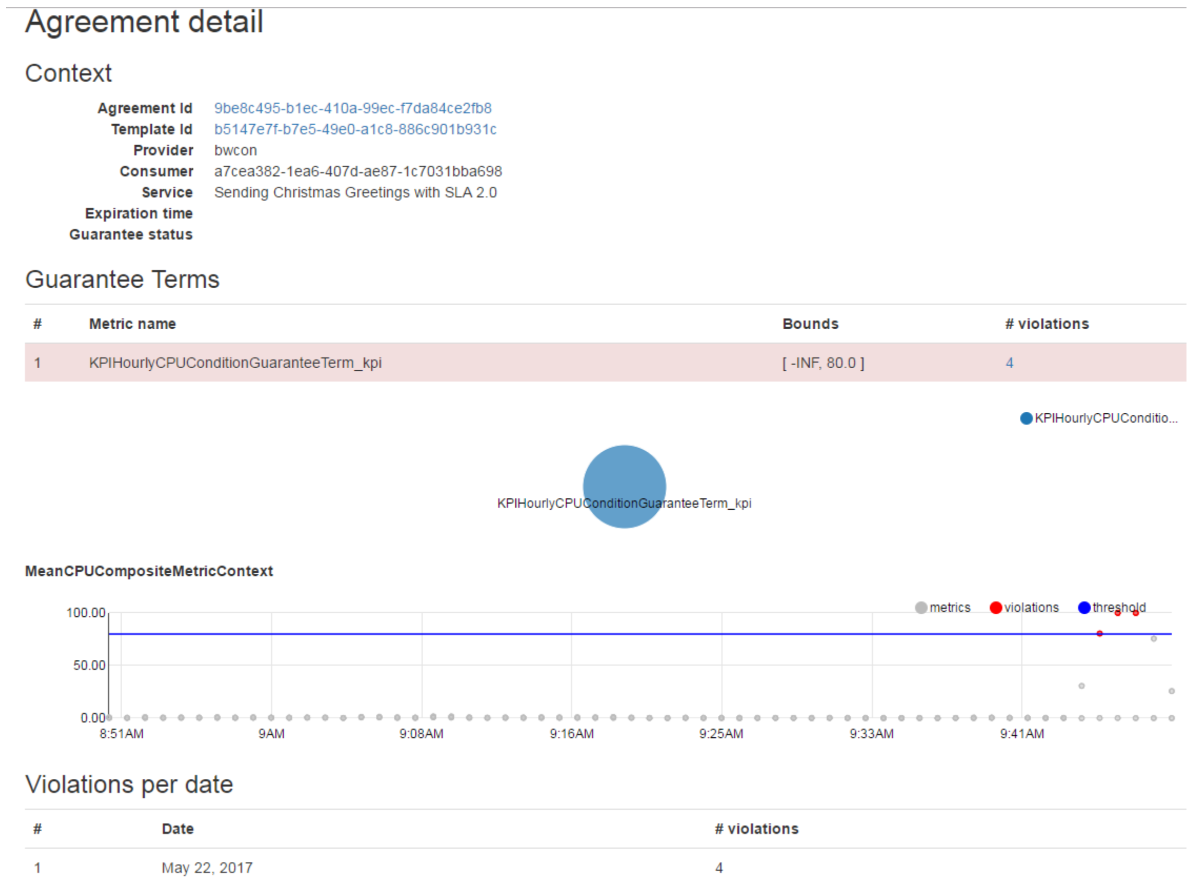

SLA Engine
Allocation
MarketPlace
Cloud Provider Eng.
Monitoring Eng.
SLA Generator
SLA Service
Design
newTemplate(json)
template = genTemplate(json)
newTemplate(template)
template
publishBundle(template, json)
after customer buys bundle
Deployment
deployBundle(template, ctx, json)
bundleDeployed(template, ctx, deploymentInfo)
agreement = createAgreement(template, ctx)
subscribe(deploymentInfo.composedMonitors)
agreement.id
Execution
while workflow execution
condition()

Figure 47 – SLA Engine Sequence Diagram



Agreement detail

Context

Agreement Id	9be8c495-b1ec-410a-99ec-f7da84ce2fb8
Template Id	b5147e7f-b7e5-49e0-a1c8-886c901b931c
Provider	bwcon
Consumer	a7cea382-1ea6-407d-ae87-1c7031bba698
Service	Sending Christmas Greetings with SLA 2.0
Expiration time	
Guarantee status	

Guarantee Terms

#	Metric name	Bounds	# violations
1	KPIHourlyCPUConditionGuaranteeTerm_kpi	[-INF, 80.0]	4

Violations per date

#	Date	# violations
1	May 22, 2017	4

Figure 48 – Agreement assessment

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)	Integrated
Generation of WS-Agreement models	Generates the SLA agreement according to the predefined template. An agreement is generated for each purchased BPaaS bundle.	Yes (2nd release)	Integrated
Integration with Monitoring Engine	Subscribes to Monitoring Engine in order to receive (measurement) data about a BPaaS bundle execution.	Yes (1st release)	Integrated
Cost model	Incorporates the CloudSocket cost model into the SLA documents. Penalties associated to violations map to this cost model.	Yes (2nd release)	Penalties are defined according to cost model defined by the BPaaS Allocation Environment.

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.	Analysed (see SLA Engine manuals) (2nd release)	Not integrated
Dashboard adaptation	Integrates the SLA Dashboard with the structure of the SLA agreements in CloudSocket.	Yes (1st release)	Integrated
Integration of dashboard with IdM	Integrates the SLA Dashboard with the Identity Management service used in CloudSocket.	Yes (2nd release)	Integrated

Table 18 SLA component functionalities

Manuals

The installation manual, API description, unit test and handbooks are detailed in the alive wiki documentation: https://www.cloudsocket.eu/common-understanding-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.

Credentials: A UULM GitLab account is required in order to access the resources.

Instances

There are different instances deployed on the UULM infrastructure to follow the continuous integration definition [34] for the second prototype. We have defined two environments: Integration and Demo. All the instances are published as a SaaS service for multitenancy in the cloud and are completely integrated with the rest of components. Both environments are deployed in the same VM due to the lack of available public IPs.

Integration Environment:

- http://134.60.64.232:8003/api: SLA Core instance.
- http://134.60.64.232:8005/: SLA Dashboard instance.

Demo Environment:

- http://134.60.64.232:8080/api: SLA Core instance.
- http://134.60.64.232:8000/: SLA Dashboard instance.

5 PROTOTYPE FROM BROKER PERSPECTIVE

5.1 Introduction

This demonstration showcases [30] how the Broker interacts with the CloudSocket BPaaS Design, Allocation and Evaluation Environments. The Broker aims to design (based on evaluation results) a new business process, define a corresponding workflow, allocate resources accordingly and publish 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 respective new BPaaS bundle.

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

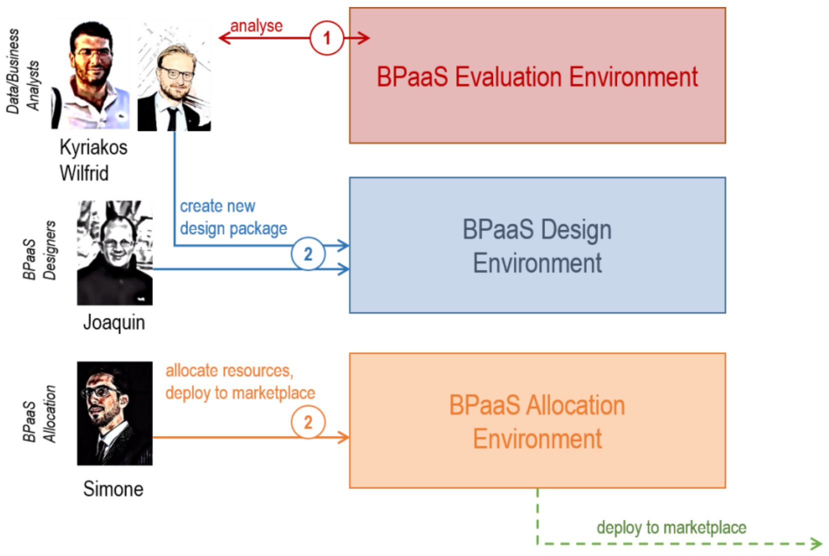


Figure 49 - Broker perspective

5.2 Demonstration from CloudSocket Broker Viewpoint

5.2.1 Involved Roles

The respective actors mapping to the roles defined in section 2.5, which 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 the BPaaS customer undertaking the role of a business engineer to search for a new process on his mobile device

5.2.2 Assessment of Indicators for Running BPaaS Bundles

While the user just sees a nice UI that enables him/her to explore the current status of the BPaaS performance, behind the scenes the implementation of the corresponding analysis functionality relied on a research prototype that was able to address the two following challenges: (a) harvesting of the analysis information from different components, which adopts different formats and structures; (b) coverage of all layers of abstraction for KPI analysis along with some flexibility on the way KPIs can be defined.

These two challenges were addressed by employing a semantic approach [32], in the context of WP3 and T3.3 in particular, which has been deemed as the most suitable for information integration purposes. In this approach, the harvested information, mainly from different REST APIs, is semantically enhanced and linked before it is stored in a semantic knowledge base. This semantic enhancement and linking relies on two ontologies: (a) an evaluation ontology that captures the dependencies involved in the hierarchy of a BPaaS; (b) a KPI ontology relying on OWL-Q for the semantic specification of KPIs and their constituent parts. Based on this approach, the semantic description of a KPI is exploited in order to produce a semantic SPARQL query, which is then issued on the semantic knowledge base in order to retrieve the corresponding KPI measurement. This approach also enables: (a) the exploration of the KPI metric space by allowing brokers to specify different KPI metric formulas and metric contexts and produce, based on them, the respective measurements. Such an exploration support the KPI metric determination process, which is considered as a difficult activity that requires some sort of creativity; (b) the drill-down of KPIs in order to support root cause analysis for KPI violations. In particular, based on a KPI dependency hierarchy, which has the KPI under examination on top, the proposed approach is able to produce a respective measurement tree which reflects the measurements mapping to the metrics of this hierarchy.

In this demo, by focusing on the Send Invoice BPaaS bundle, we showcase how the main outcome of the semantic approach, the Conceptual Analytics (REST) Service (see Section 5.3.1.2), supports the drill-down of the mean availability KPI for the whole BPaaS. Figure 50 depicts the KPI dependency hierarchy for the mean availability KPI, while Figure 51 shows the result of directly calling the REST service in order to perform the drill down.

The video at: https://youtu.be/P2Dl-v2nlxU shortly describes the aforementioned semantic approach and then attempts to explain the steps involved in directly calling the Conceptual Analytics Service. The drill-down result is also shown, which witnesses the measurement tree reflecting the KPI dependency hierarchy. As already stated, the BPaaS Design / Evaluation Environment will, in principle, call this service in the background so the user would not have to explore the REST service and make the calls there. The nice UI of this environment will just enable the user to ask for the drill-down execution by carefully selecting the most suitable KPI to be explored for a certain BPaaS.

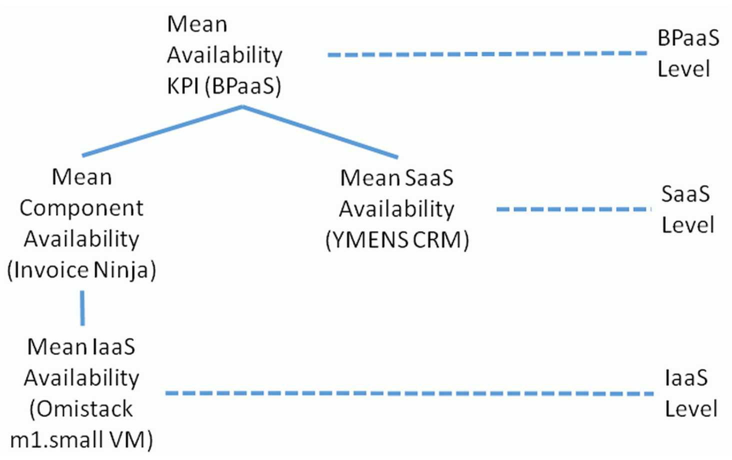
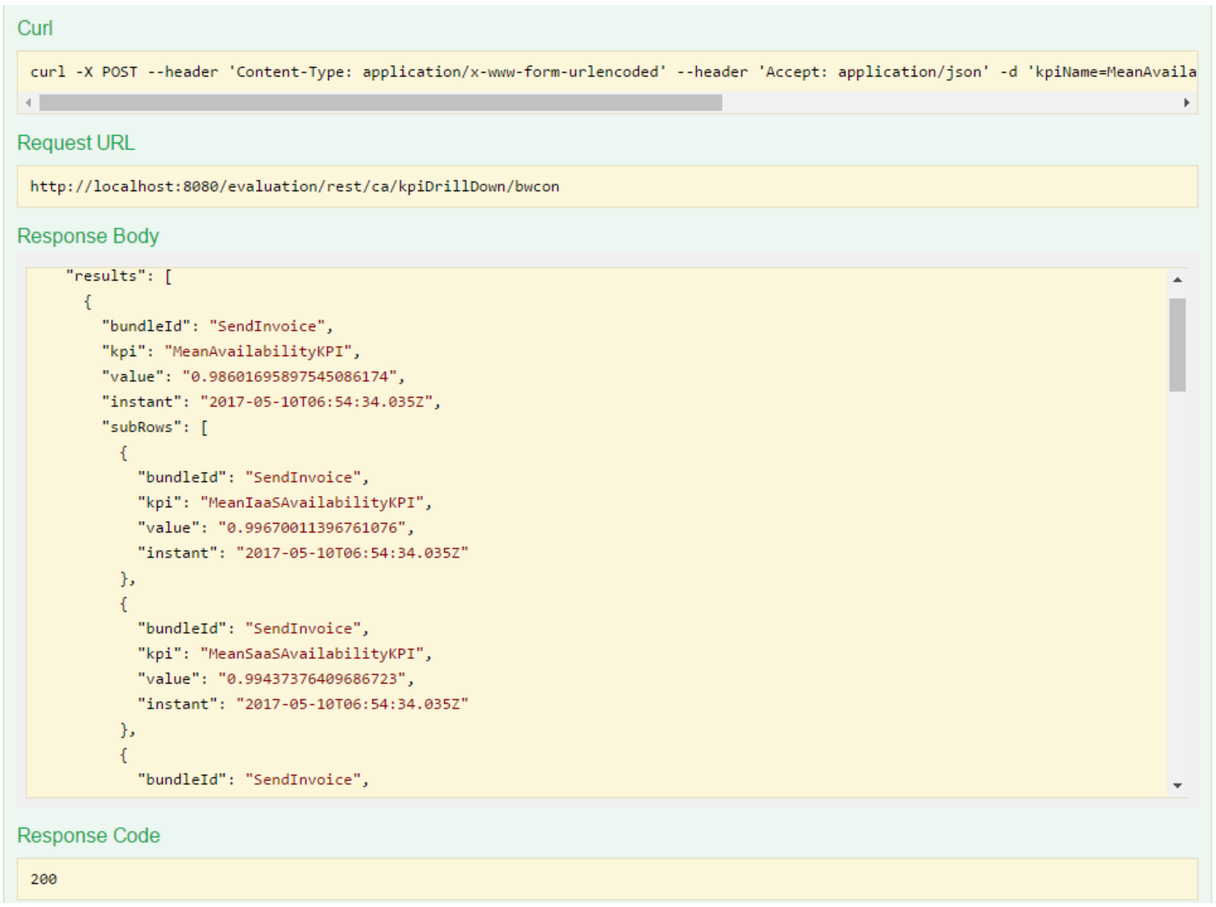

Mean Availability KPI (BPaaS)
BPaaS Level
Mean Component Availability (Invoice Ninja)
Mean SaaS Availability (YMENS CRM)
SaaS Level
Mean IaaS Availability (Omistack m1.small VM)
IaaS Level

Figure 50: The KPI hierarchy for the BPaaS availability KPI


Curl

```
curl -X POST --header 'Content-Type: application/x-www-form-urlencoded' --header 'Accept: application/json' -d 'kpiName=MeanAvaila
```

Request URL

```
http://localhost:8080/evaluation/rest/ca/kpiDrillDown/bwcon
```

Response Body

```
    "results": [
      {
        "bundleId": "SendInvoice",
        "kpi": "MeanAvailabilityKPI",
        "value": "0.98601695897545086174",
        "instant": "2017-05-10T06:54:34.035Z",
        "subRows": [
          {
            "bundleId": "SendInvoice",
            "kpi": "MeanIaaSAvailabilityKPI",
            "value": "0.99670011396761076",
            "instant": "2017-05-10T06:54:34.035Z"
          },
          {
            "bundleId": "SendInvoice",
            "kpi": "MeanSaaSAvailabilityKPI",
            "value": "0.99437376409686723",
            "instant": "2017-05-10T06:54:34.035Z"
          },
          {
            "bundleId": "SendInvoice",
```

Response Code

```
200
```

Figure 51: Snapshot of the results of the drill-down query for the Mean BPaaS Availability KPI

The CloudSocket 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 52) and the aforementioned semantic approach, the data retrieved from the monitoring interface is visually aggregated as KPIs, goals and perspectives (Figure 53).

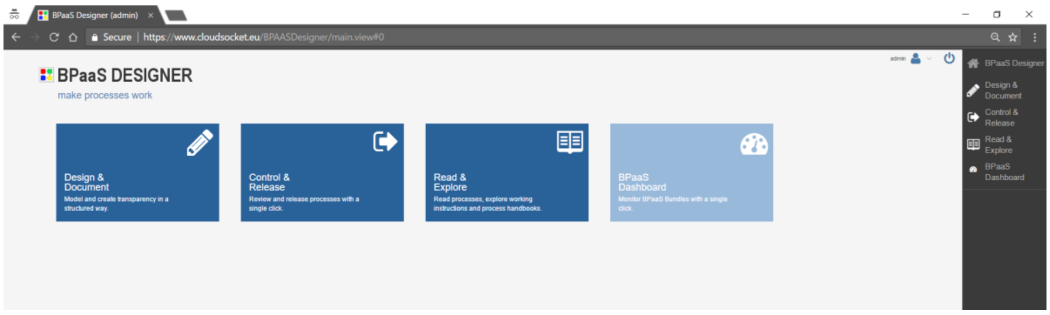
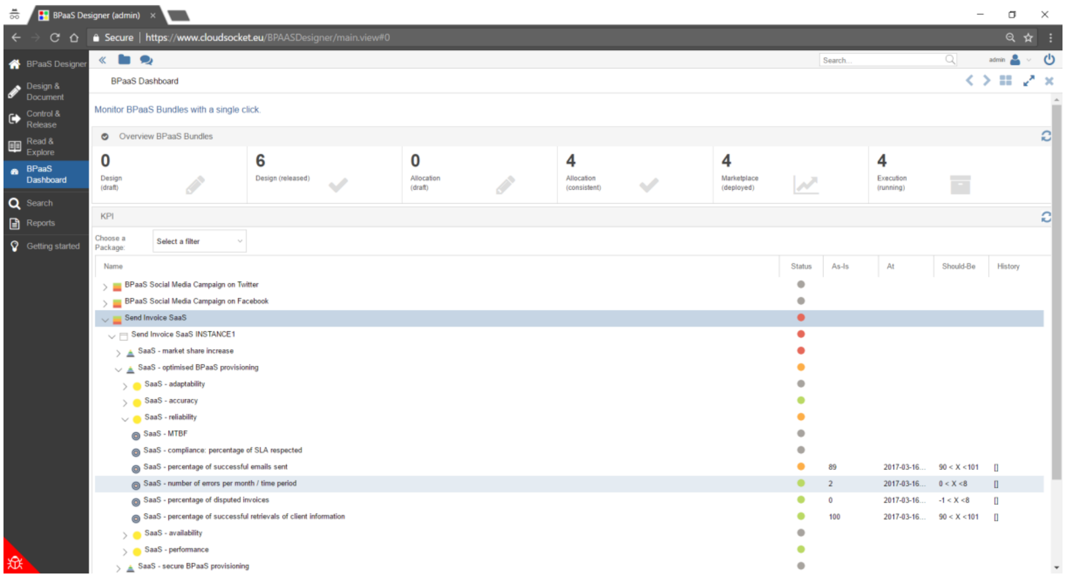

BPaaS Designer (admin)
Secure https://www.cloudsocket.eu/BPAASDesigner/main.view#0
BPaaS DESIGNER
make processes work
Design & Document
Model and create transparency in a structured way.
Control & Release
Review and release processes with a single click.
Read & Explore
Read processes, explore working instructions and process handbooks.
BPaaS Dashboard
Monitor BPaaS Bundles with a single click.
admin
BPaaS Designer
Design & Document
Control & Release
Read & Explore
BPaaS Dashboard

Figure 52 – Monitor & Assess



Monitor BPaaS Bundles with a single click.

Overview BPaaS Bundles

0	6	0	4	4	4
Design (draft)	Design (released)	Allocation (draft)	Allocation (consistent)	Marketplace (deployed)	Execution (running)

KPI

Choose a Package: Select a filter

Name	Status	As-Is	At	Should-Be	History
BPaaS Social Media Campaign on Twitter	⚪				
BPaaS Social Media Campaign on Facebook	⚪				
Send Invoice SaaS	🔴				
Send Invoice SaaS INSTANCE1	🔴				
SaaS - market share increase	🔴				
SaaS - optimised BPaaS provisioning	🟠				
SaaS - adaptability	⚪				
SaaS - accuracy	🟢				
SaaS - reliability	🟠				
SaaS - MTBF	⚪				
SaaS - compliance: percentage of SLA respected	⚪				
SaaS - percentage of successful emails sent	🟠	89	2017-03-16...	90 < X <101	[]
SaaS - number of errors per month / time period	🟢	2	2017-03-16...	0 < X <8	[]
SaaS - percentage of disputed invoices	🟢	0	2017-03-16...	-1 < X <8	[]
SaaS - percentage of successful retrievals of client information	🟢	100	2017-03-16...	90 < X <101	[]
SaaS - availability	⚪				
SaaS - performance	🟢				
SaaS - secure BPaaS provisioning	⚪				

Figure 53 – Hierarchal view for BPaaS bundle

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

5.2.3 Creation of New Design Package

The design is done using graphical models as an interaction means between business process and workflow/technical designers (see Figure 54). In the demonstration, we show the created models (Figure 55, Figure 56), and an overview of the features that the BPaaS Design Environment integrates in order to create correct models.

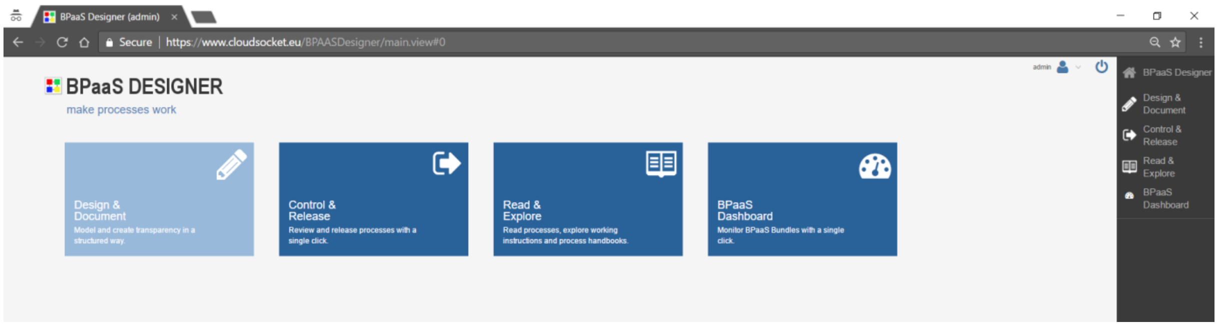

BPaaS Designer (admin)
Secure https://www.cloudsocket.eu/BPAASDesigner/main.view#0
BPaaS DESIGNER
make processes work
Design & Document
Model and create transparency in a structured way.
Control & Release
Review and release processes with a single click.
Read & Explore
Read processes, explore working instructions and process handbooks.
BPaaS Dashboard
Monitor BPaaS Bundles with a single click.
admin
BPaaS Designer
Design & Document
Control & Release
Read & Explore
BPaaS Dashboard

Figure 54 – Design & Document

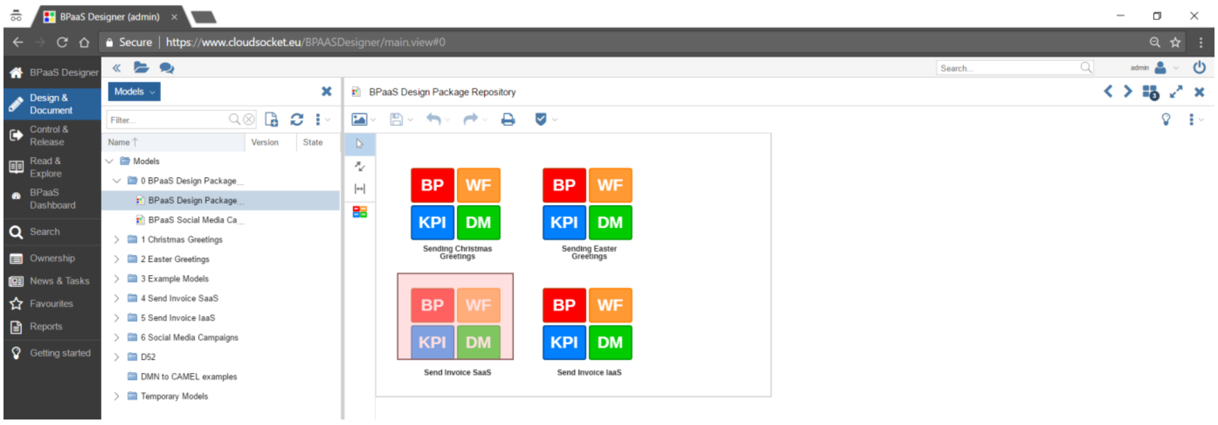


Figure 55 – Send Invoice SaaS Design Package

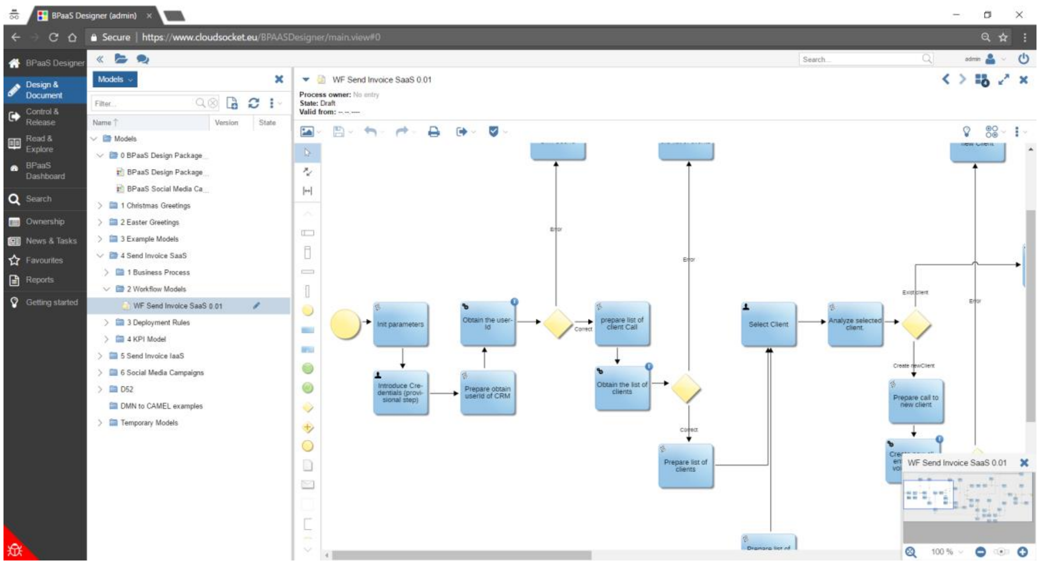
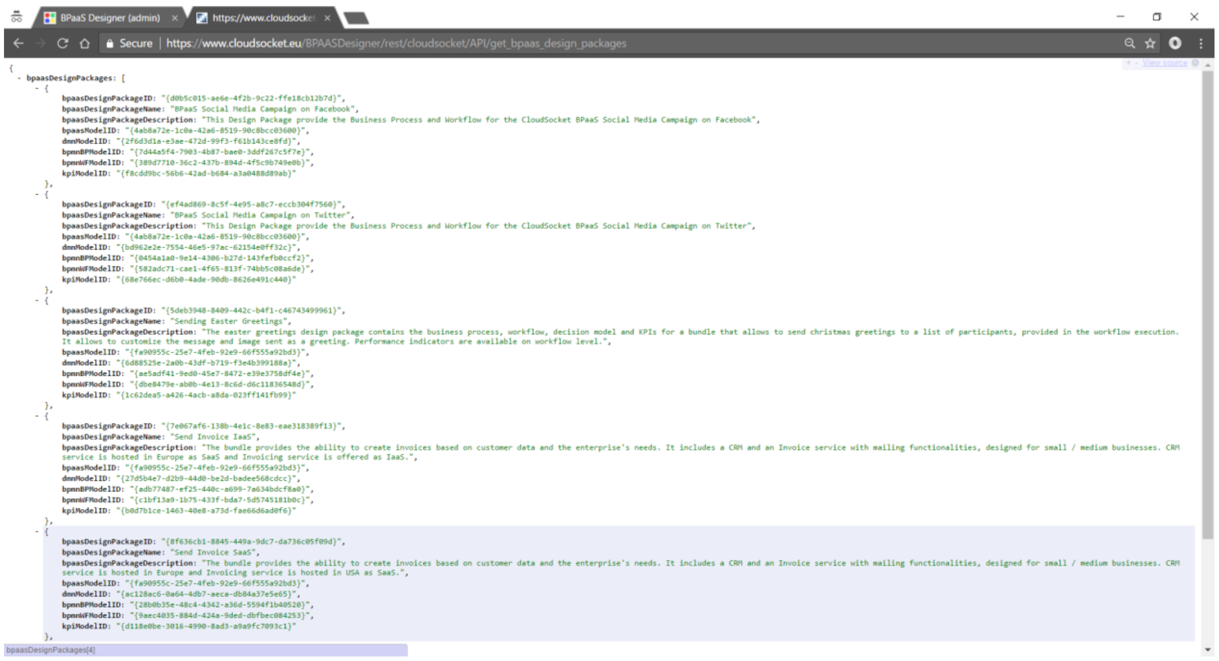


Figure 56 – Send Invoice SaaS (executable workflow)

The package is designed (Figure 55) 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 design package.

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

As a final step, the package is made available by releasing it in the Modeling Layer of the Allocation Environment. This makes the package also available via the BPaaS Design Environment API for further use in the Allocation Environment (Figure 57).


BPaaS Designer (admin)
https://www.cloudsocket
Secure
https://www.cloudsocket.eu/BPAASDesigner/rest/cloudsocket/API/get_bpaas_design_packages

```
{
 - bpaasDesignPackages: [
    - {
         bpaasDesignPackageID: "{d0b5c015-ae6e-4f2b-9c22-ffe18cb12b7d}",
         bpaasDesignPackageName: "BPaaS Social Media Campaign on Facebook",
         bpaasDesignPackageDescription: "This Design Package provide the Business Process and Workflow for the CloudSocket BPaaS Social Media Campaign on Facebook",
         bpaasModelID: "{4ab8a72e-1c0a-42a6-8519-90c8bcc03600}",
         dmnModelID: "{2f6d3d1a-e3ae-472d-99f3-f61b143ce8fd}",
         bpmnBPModelID: "{7d44a5f4-7903-4b87-bae0-3ddf267c5f7e}",
         bpmnWFModelID: "{389d7710-36c2-437b-894d-4f5c9b749e0b}",
         kpiModelID: "{f8cdd9bc-56b6-42ad-b684-a3a0488d89ab}"
      },
    - {
         bpaasDesignPackageID: "{ef4ad869-8c5f-4e95-a8c7-eccb304f7560}",
         bpaasDesignPackageName: "BPaaS Social Media Campaign on Twitter",
         bpaasDesignPackageDescription: "This Design Package provide the Business Process and Workflow for the CloudSocket BPaaS Social Media Campaign on Twitter",
         bpaasModelID: "{4ab8a72e-1c0a-42a6-8519-90c8bcc03600}",
         dmnModelID: "{bd962e2e-7554-46e5-97ac-62154e0ff32c}",
         bpmnBPModelID: "{0454a1a0-9e14-4306-b27d-143fefb0ccf2}",
         bpmnWFModelID: "{582adc71-cae1-4f65-813f-74bb5c08a6de}",
         kpiModelID: "{68e766ec-d6b0-4ade-90db-8626e491c440}"
      },
    - {
         bpaasDesignPackageID: "{5deb3948-8409-442c-b4f1-c46743499961}",
         bpaasDesignPackageName: "Sending Easter Greetings",
         bpaasDesignPackageDescription: "The easter greetings design package contains the business process, workflow, decision model and KPIs for a bundle that allows to send christmas greetings to a list of participants, provided in the workflow execution.
         It allows to customize the message and image sent as a greeting. Performance indicators are available on workflow level.",
         bpaasModelID: "{fa90955c-25e7-4feb-92e9-66f555a92bd3}",
         dmnModelID: "{6d88525e-2a0b-43df-b719-f3e4b399188a}",
         bpmnBPModelID: "{ae5adf41-9ed0-45e7-8472-e39e3758df4e}",
         bpmnWFModelID: "{dbe8479e-ab0b-4e13-8c6d-d6c11836548d}",
         kpiModelID: "{1c62dea5-a426-4acb-a8da-023ff141fb99}"
      },
    - {
         bpaasDesignPackageID: "{7e067af6-138b-4e1c-8e83-eae318389f13}",
         bpaasDesignPackageName: "Send Invoice IaaS",
         bpaasDesignPackageDescription: "The bundle provides the ability to create invoices based on customer data and the enterprise's needs. It includes a CRM and an Invoice service with mailing functionalities, designed for small / medium businesses. CRM
         service is hosted in Europe as SaaS and Invoicing service is offered as IaaS.",
         bpaasModelID: "{fa90955c-25e7-4feb-92e9-66f555a92bd3}",
         dmnModelID: "{27d5b4e7-d2b9-44d0-be2d-badee568cdcc}",
         bpmnBPModelID: "{adb77487-ef25-440c-a699-7a634bdcf8a0}",
         bpmnWFModelID: "{c1bf13a9-1b75-433f-bda7-5d5745181b0c}",
         kpiModelID: "{b0d7b1ce-1463-40e8-a73d-fae66d6ad0f6}"
      },
    - {
         bpaasDesignPackageID: "{8f636cb1-8845-449a-9dc7-da736c05f09d}",
         bpaasDesignPackageName: "Send Invoice SaaS",
         bpaasDesignPackageDescription: "The bundle provides the ability to create invoices based on customer data and the enterprise's needs. It includes a CRM and an Invoice service with mailing functionalities, designed for small / medium businesses. CRM
         service is hosted in Europe and Invoicing service is hosted in USA as SaaS.",
         bpaasModelID: "{fa90955c-25e7-4feb-92e9-66f555a92bd3}",
         dmnModelID: "{ac128ac6-0a64-4db7-aeca-db84a37e5e65}",
         bpmnBPModelID: "{28b0b35e-48c4-4342-a36d-5594f1b40520}",
         bpmnWFModelID: "{9aec4035-884d-424a-9ded-dbfbec084253}",
         kpiModelID: "{d118e0be-3016-4990-8ad3-a9a9fc7093c1}"
      },
bpaasDesignPackages[4]
```

Figure 57 – BPaaS Design Environment API

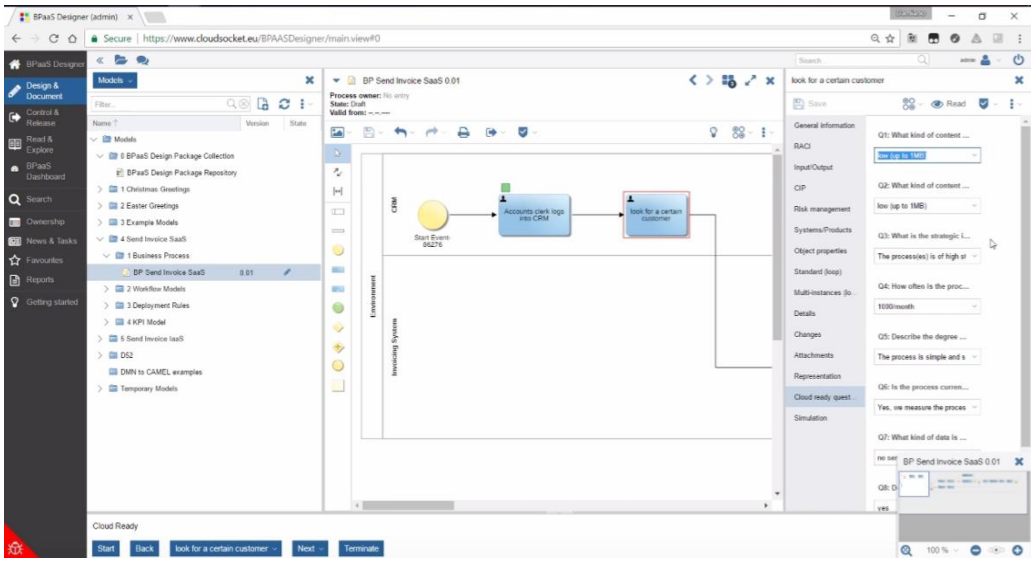
The following video shows the design process and interaction to create the new BPaaS Design package https://youtu.be/Ce_KX3nC9Y8.

After the design package and, in particular, the business process and the executable workflow have been designed, the environment gives the possibility to perform some quality checks in order to evaluate the cloud readiness level of the business process, check the presence of unexpected behaviour in the workflow, estimate execution times and costs using a simulation, and semantically enrich the workflow in order to support the discovery of best matching services.

In the following sections, we show an overview of all these capabilities.

5.2.3.1 *Cloud Readiness Quality Check*

From the BPaaS Design Environment, once a business process model is selected, it is possible to check its cloud readiness. The check is based on a questionnaire that the modeller has to answer. This questionnaire is related both to the whole business process and to every specific activity over the modelled business process flow (Figure 58).



look for a certain customer

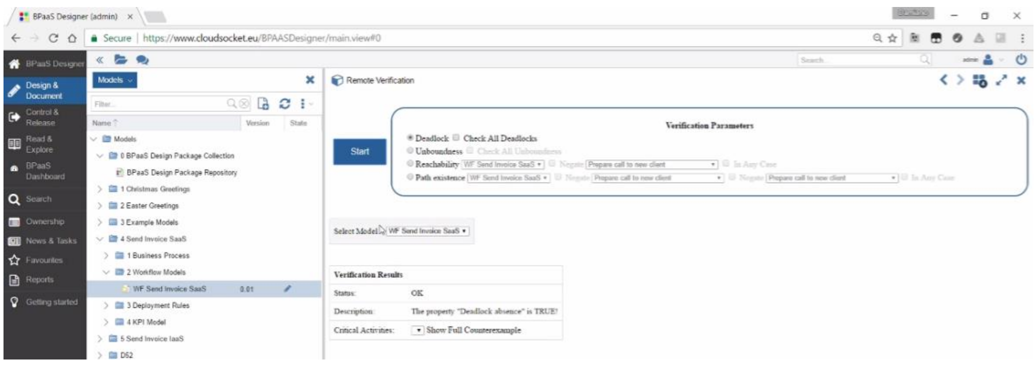
Q1: What kind of content ...	low (up to 1MB)
Q2: What kind of content ...	low (up to 1MB)
Q3: What is the strategic i...	The process(es) is of high st
Q4: How often is the proc...	1000/month
Q5: Describe the degree ...	The process is simple and s
Q6: Is the process curren...	Yes, we measure the proces
Q7: What kind of data is ...	no ser
Q8: D	yes

Figure 58 – BPaaS Design Environment Cloudreadiness

The following video show the whole process of cloud readiness quality check: https://youtu.be/OY-Iq7orNZU

5.2.3.2 Business Process and Workflow Verification

Once a business process or a workflow has been modelled, it is possible to formally check its correctness in terms of expected behaviour. In particular, via the verification process it is possible to evaluate the presence of deadlocked paths, unbounded tasks or the existence of a specific behaviour in the Business Process (Figure 59).



Verification Parameters

- [✓ Deadlock] Check All Deadlocks
- Unboundness Check All Unboundness
- Reachability WF Send Invoice SaaS Negate Prepare call to new client In Any Case
- Path existence WF Send Invoice SaaS Negate Prepare call to new client Negate Prepare call to new client In Any Case

Select Model: WF Send Invoice SaaS

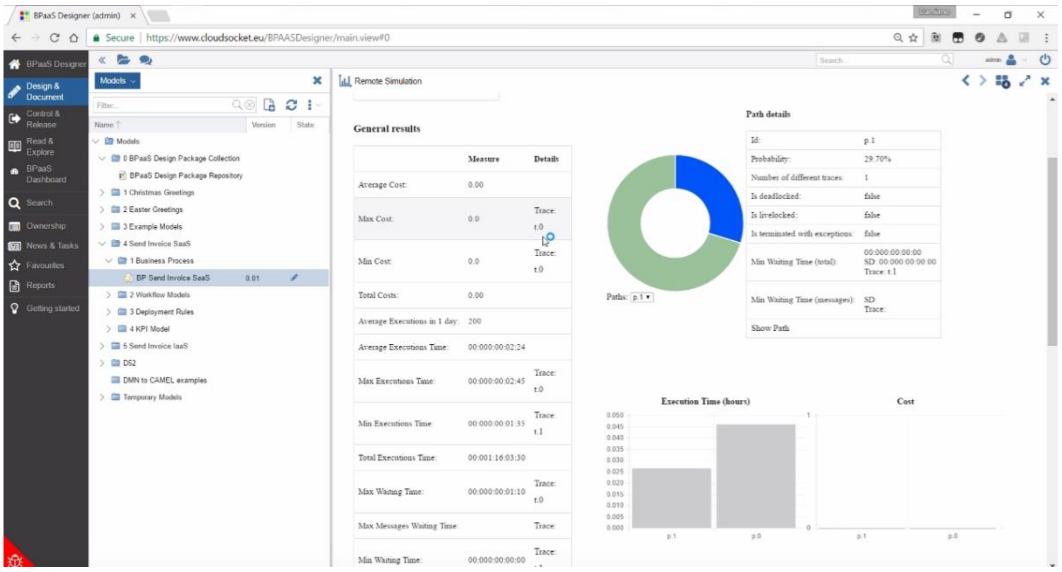
Verification Results	
Status:	OK
Description:	The property "Deadlock absence" is TRUE!
Critical Activities:	Show Full Counterexample

Figure 59 BPaaS Design Environment Formal Verification

The following videos show how the verification takes place for the business process and for the workflow respectively: https://youtu.be/GH6Yfu-tlY4 and https://youtu.be/Td_cRHl-dqU.

5.2.3.3 Business Process and Workflow Simulation

The last step available to evaluate business process and workflow behaviour is a simulation in order to estimate costs, times and other relevant parameters. The simulation is available in the BPaaS Design Environment as an additional feature, like the aforementioned formal verification one, and obtained simply with a right click on the model (Figure 60).


Remote Simulation

General results

	Measure	Details
Average Cost:	0.00	
Max Cost:	0.0	Trace: t.0
Min Cost:	0.0	Trace: t.0
Total Costs:	0.00	
Average Executions in 1 day:	200	
Average Executions Time:	00:000:00:02:24	
Max Executions Time:	00:000:00:02:45	Trace: t.0
Min Executions Time:	00:000:00:01:35	Trace: t.1
Total Executions Time:	00:001:16:03:30	
Max Waiting Time:	00:000:00:01:10	Trace: t.0
Max Messages Waiting Time:		Trace:
Min Waiting Time:	00:000:00:00:00	Trace:

Path details

Id:	p.1
Probability:	29.70%
Number of different traces:	1
Is deadlocked:	false
Is livelocked:	false
Is terminated with exceptions:	false
Min Waiting Time (total):	00:000:00:00:00 SD: 00:000:00:00:00 Trace: t.1
Min Waiting Time (messages):	SD: Trace:
Show Path	

Figure 60 - BPaaS Design Environment Simulation

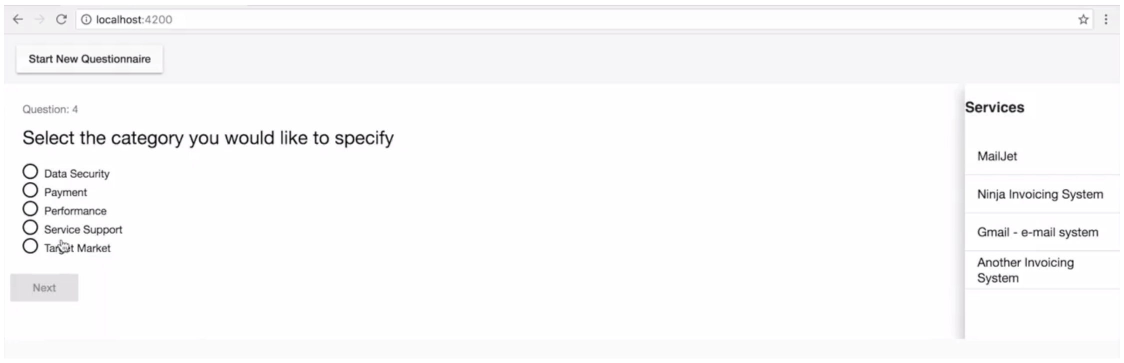
The following videos show how the simulation takes place for the business process and for the workflow, respectively: https://youtu.be/1XPWexBit3c and https://youtu.be/bpWmgsuRCqY.

5.2.3.4 Supporting the Discovery and Selection of Services and Workflows

Identifying appropriate cloud services as well as their orchestrations (i.e., workflows) is an arduous task. This is mainly true in enterprises where IT expertise is scarce or the Business side is poorly aligned with the IT side. We support this task by proposing an environment that enables both the human and machine interpretation. The former allows specifying requirements from a business perspective. The machine interpretability, on the other side, is made possible through semantic technologies that support the matching between the specified requirements and the cloud services / workflows descriptions.

Within this context, we have developed two research items – the Semantic Lifting and the Context-Adaptive Questionnaire. These addressed the following research questions, respectively:

1) How can a modeling environment support consistency between human and machine interpretation of BPaaS models for the Business-to-IT alignment?
2) How can semantic lifting and service discovery be performed more intuitively and smartly?


localhost:4200

Start New Questionnaire

Question: 4

Select the category you would like to specify

- Data Security
- Payment
- Performance
- Service Support
- Target Market

Next

Services

- MailJet
- Ninja Invoicing System
- Gmail - e-mail system
- Another Invoicing System

Figure 61 – Context-Adaptive Questionnaire

A video that introduces the three prototypes and how to use them can be found under the following link: https://youtu.be/f_9i6fSlc_A.

5.2.4 Allocation and Deployment of the New Bundle

5.2.4.1 Demonstration of Allocation and Deployment

Once the CloudSocket Broker completes a new Design Package using the BPaaS 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 CloudSocket Broker can involve a sales consultant and an allocation expert in order to delegate to them the main responsibilities of bundle creation.

In order to create a new bundle, the allocation expert accesses the Allocation Tool (Figure 62).

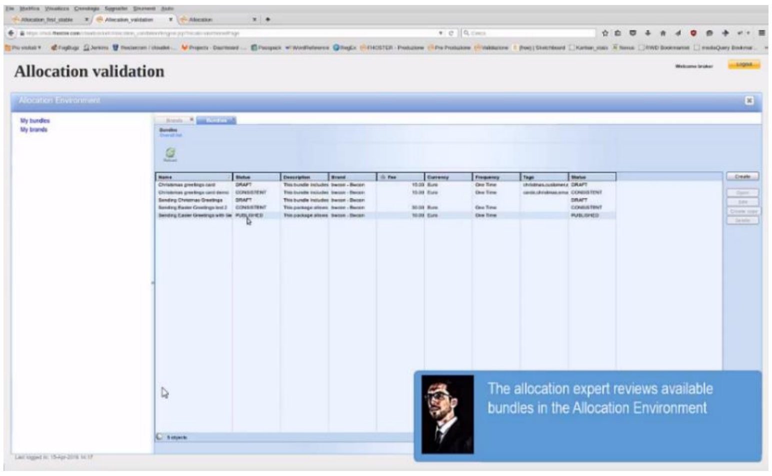

Allocation validation
My bundles
My brands
The allocation expert reviews available bundles in the Allocation Environment

Figure 62 – 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 BPaaS Design Environment (Figure 63). The CloudSocket 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 BPaaS 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 concrete atomic services to be invoked by it.

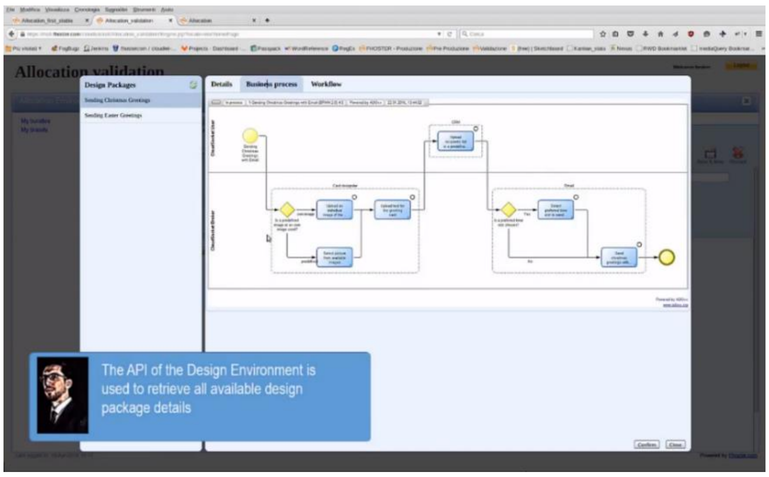
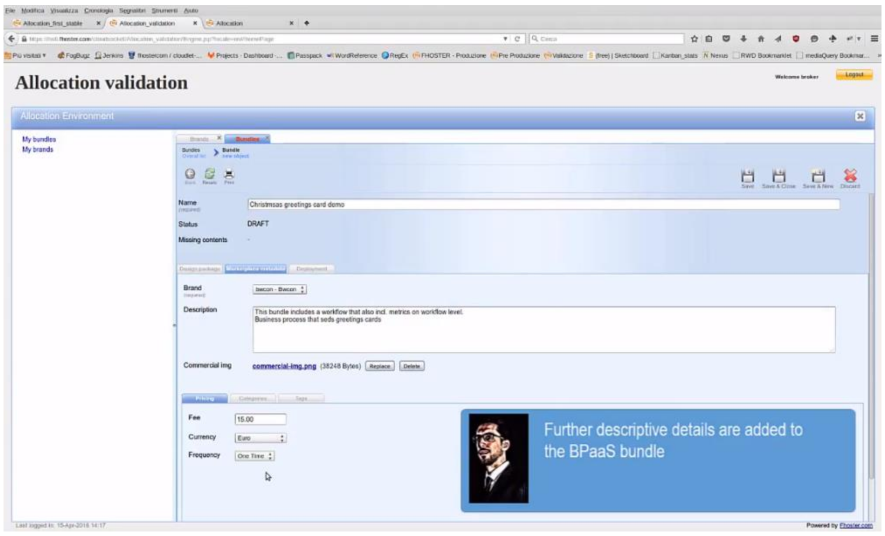

Allocation validation
Design Packages
Sending Christmas Greetings
Sending Easter Greetings
Details
Business process
Workflow
The API of the Design Environment is used to retrieve all available design package details
Confirm
Close

Figure 63 – 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 64).



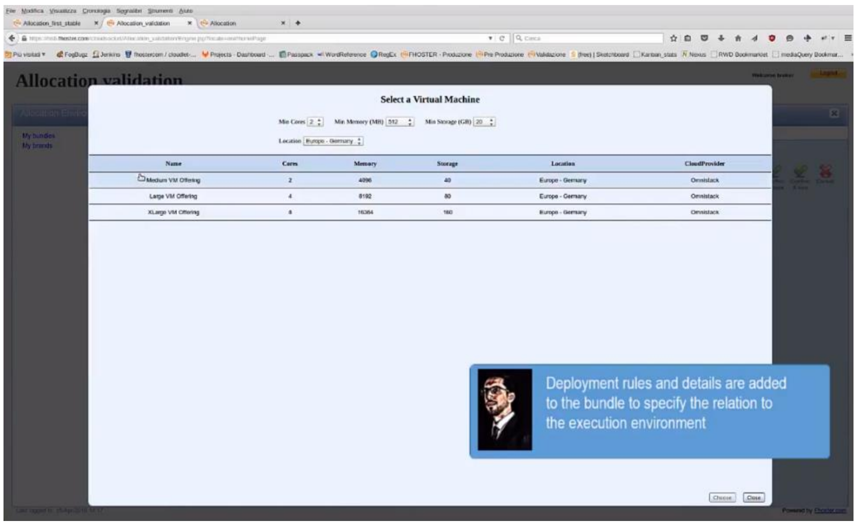
Name: Christmsas greetings card demo
Status: DRAFT
Missing contents: -
Brand: becon - Bwcon
Description: This bundle includes a workflow that also incl. metrics on workflow level.
Business process that seds greetings cards
Commercial img: commercial-img.png (38248 Bytes) Replace Delete
Fee: 15.00
Currency: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

Figure 64 – Definition of the BPaaS bundle

The allocation expert takes then over. He 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 or Platform as a Service Offering that will host it. The Allocation Tool will help in such a selection by filtering the offerings space for each software component by considering the component's minimum hardware requirements and its compatible Operating System(s). As such, the Infrastructure / Platform as a Service selection / allocation scenario is covered. Then the allocation expert needs to allocate also all atomic services (currently these maps to the selection of one endpoint from those available for each atomic service). To complete the allocation phase, the allocation expert could model the Key Performance Indicators, as well as the KPI conditions and associate some of them to SLOs in order to define the overall Service Level Agreement. The KPI conditions can also be used in order to define some useful adaptation rules, for example, to replace some services if they are not performing as expected.

The allocation decisions are translated by the Allocation Tool into a deployment plan defined using the CAMEL[3] language (Figure 65).

[3] www.camel-dsl.org


Select a Virtual Machine

Min Cores 2 Min Memory (MB) 512 Min Storage (GB) 20

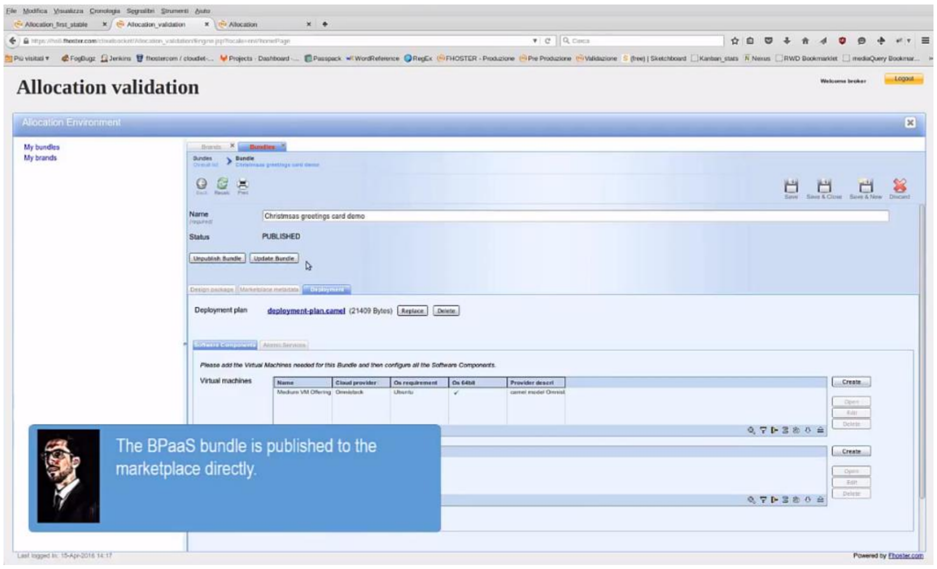
Location Europe - Germany

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 65 – 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 66).


Allocation validation

Name: Christmsas 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.

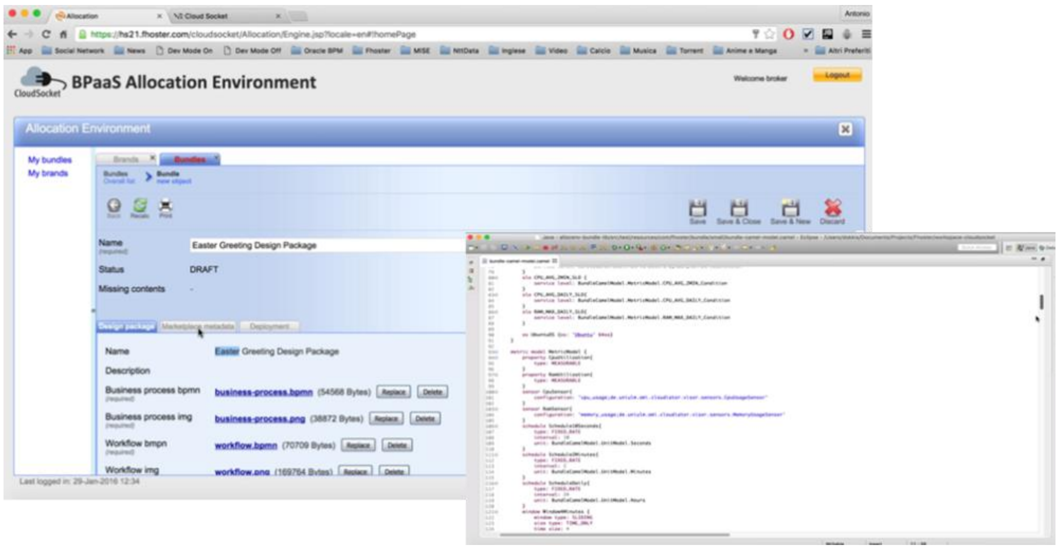
Virtual machines

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 66 – BPaaS bundle ready to be published

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



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

Figure 67 - Allocation environment

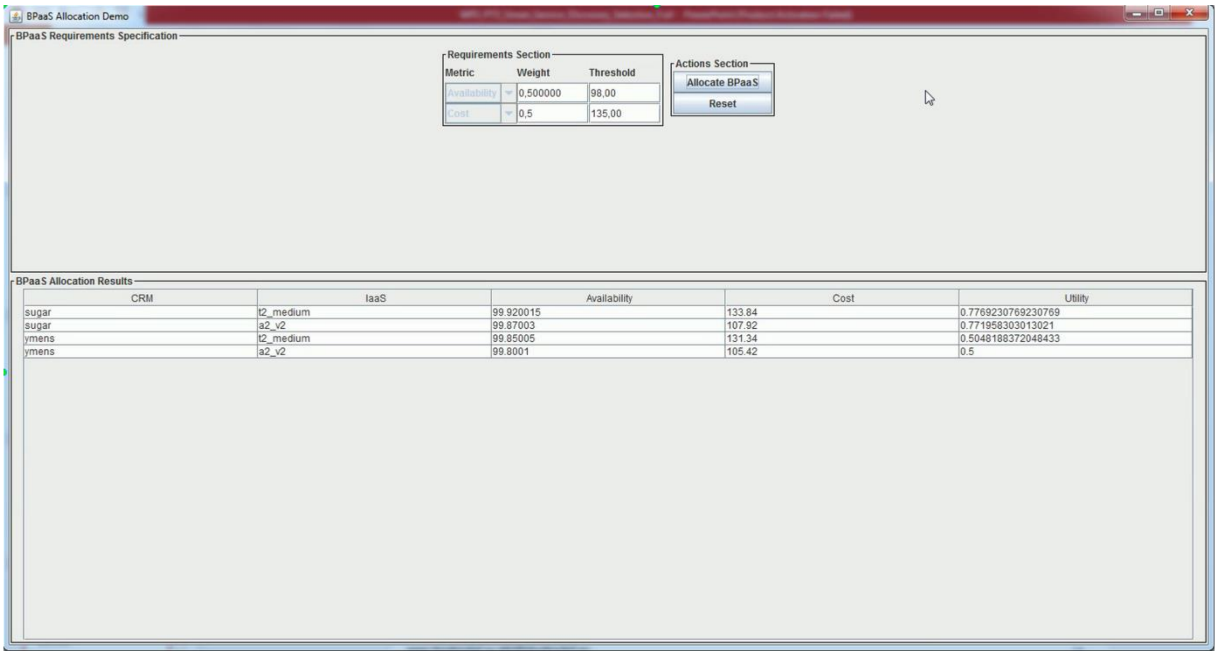
5.2.4.2 Demonstration of Research Prototype - Automatic BPaaS allocation

The previous part showed a bundle editing tool, which enables the broker to manually allocate cloud services to a certain BPaaS and thus produce a corresponding BPaaS bundle. This manual allocation could be automatised and this would certainly accelerate the BPaaS bundle development time. In this respect, the BPaaS allocation tool could benefit from an automatic approach that is able to confront the following problems that relate to the manual allocation limitations as well as disadvantages of related work: (a) potentially huge solution space, which makes it hard for the broker to inspect it in order to make the appropriate selections; (b) existence of multiple levels, which raise the complexity of the allocation problem and whose dependencies have been neglected by the literature; (c) different selection options even for the same BPaaS task: in particular, there can be cases where there exist both internally-developed software components and externally-available SaaS that can realise the functionality of a certain BPaaS task. In this respect, the broker would have to select either the software component (which would then map to having to also select corresponding IaaS services able to support its execution) or one from the alternative SaaS services available which would make his/her allocation task even harder.

Such an automatic approach, which automates the allocation task and address the aforementioned challenges, has been developed in the context of WP3 (and T3.2 in particular) and has been mapped to a certain Smart Service Discovery and Selection tool. This tool mainly offers its services in the form of two REST services: (a) a service discovery one; (b) a service selection one, while a nice but limited UI has been realised on top of it, which works on a particular BPaaS and only for a small set of non-functional properties that constitute corresponding technical requirements for the actual selection. However, due to the followed REST-based architecture, this UI can be interchanged with the much nicer and complete UI of the BPaaS Allocation Environment.

The semantic tool exploits: (a) prominent semantic functional and non-functional service discovery algorithms, which have been integrated in a uniform semantic service discovery framework; (b) a service selection algorithm, which: (i) takes into account the broker non-functional requirements, the structure of the BPaaS and the cloud services available to realise the functionality or support the execution of the BPaaS workflow tasks, (ii) produces an optimisation problem out of them, and (iii) then employs a Constraint Programming engine for solving it. This tool also exhibits various other benefits, apart from those mapping to the aforementioned problems that have been appropriately solved, which can be inspected in deliverable D3.4 [33].

In this demonstration, the focus is again on the Send Invoice BPaaS, and especially on how to automatically allocate it based on the offered automatic service discovery and selection tool. The demonstration, which can be found at: https://youtu.be/77hNj0eyvtQ, showcases that, by exploiting the simple but straightforward UI of the tool, the broker just provides some preferences and constraints over two non-functional metrics (availability & price) and then he / she is presented with a set of ranked service combinations that satisfy these preferences and constraints. In fact, the broker can modify his/her input, like making the constraint stricter or changing his/her preferences, in order to obtain an even more plausible outcome. The service combinations are produced rapidly so the whole service discovery and selection process is quite transparent to the broker. In this respect, the benefits of the tool are quite obvious and important while its design has been quite flexible and this can enable integrating it with the BPaaS Allocation UI in order to enhance the broker experience and make him/her explore in a more complete manner all suitable possibilities. The following screenshot of the UI of the service discovery and selection tool shows one result (ranked service combinations) that has been produced based on the posed broker preferences and requirements, which are also depicted.



Metric	Weight	Threshold
Availability	0,500000	98,00
Cost	0,5	135,00

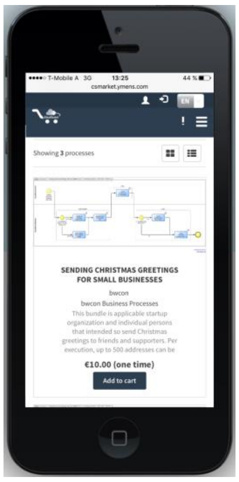
CRM	IaaS	Availability	Cost	Utility
sugar	t2_medium	99.920015	133.84	0.7769230769230769
sugar	a2_v2	99.87003	107.92	0.771958303013021
ymens	t2_medium	99.85005	131.34	0.5048188372048433
ymens	a2_v2	99.8001	105.42	0.5

Figure 68: The UI of the Smart Service Discovery and Selection Tool

5.2.5 Publication of the New Bundle on the Marketplace

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 69) and buy it as it is described in the perspective of the BPaaS Customer Section 4.



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

Figure 69- 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/dwAi7wTf47k

5.3 Required Environments

5.3.1 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 are supported: (a) KPI assessment; (b) KPI drill-down; (c) best deployment discovery for BPaaS; (d) 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 5.2.2. All of the analysis functionality offered has been realised in the context of WP3 and task T3.3 in particular. As such, this environment represents the one across the whole CloudSocket architecture from which most of the components were derived from research prototypes.

In order to support the aforementioned visualisation and analysis functionality, the CloudSocket BPaaS Evaluation Environment comprises the following components:

- *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 (not yet completed).

- *Harvesting Engine*: Responsible for drawing information from other environments, semantically lifting and integrating it, and finally storing it in the *Semantic Knowledge Base*.
- *Conceptual Analytics Engine*: Provides a REST API through which the (a)-(b) types of analysis can be performed, and implements the underlying functionality needed.
- *Deployment Discovery Engine*: Provides a REST API via which the best deployment(s) for a BPaaS can be discovered.
- *Process Mining Engine*: Provides a REST API via which process mining analysis tasks can be performed over the semantic repository.
- *Semantic Knowledge Base* (*Semantic KB*): Supplies a REST management interface, called *Semantic KB Service*, over a semantic repository realised via a *Triple Store*. This interface enables to import, update and export semantic data from the *Triple Store* as well as to pose semantic SPARQL queries that can assist in the realisation of the different types of analysis supported
- *Meta-Model Platform*: provides access to various information 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.

Figure 70 depicts the component diagram of the BPaaS Evaluation Environment. As it can be seen, there are three levels involved, covering the visualisation, analysis and the underlying storage and management of the (semantic BPaaS evaluation) data. In the context of conducting a certain analysis kind, components in each level interact with each other to properly deliver and visualise the respective analysis functionality. More information about the way these interactions are performed and what are the respective scenarios covered can be found at: https://www.cloudsocket.eu/uml/4-EvaluationEnvironment/remotedocu/modeldocu/27012016104756/modelContentHTML where the corresponding UML diagrams [2] can be viewed. This information was also covered in the D4.1 deliverable [37].

In the following, we describe all the aforementioned components, which are part of this final release of the CloudSocket prototype in separate sub-sections.

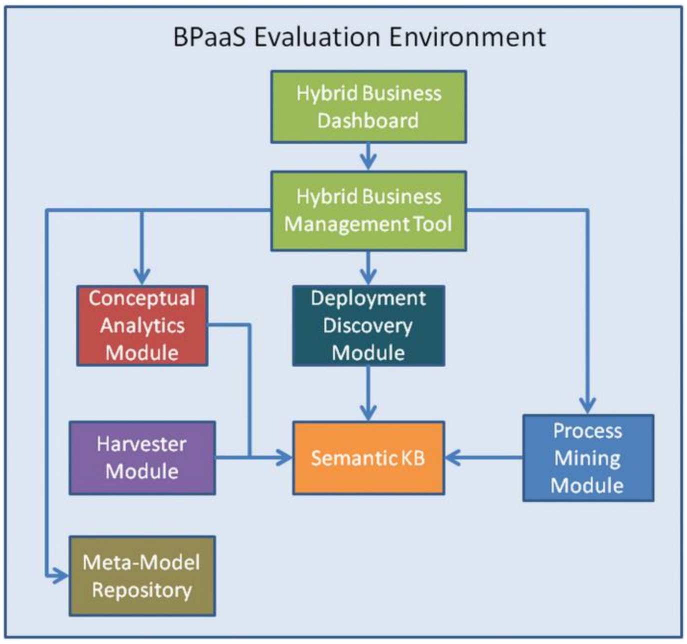

BPaaS Evaluation Environment
Hybrid Business Dashboard
Hybrid Business Management Tool
Conceptual Analytics Module
Deployment Discovery Module
Harvester Module
Semantic KB
Process Mining Module
Meta-Model Repository

Figure 70: The architecture of the BPaaS Evaluation Environment

5.3.1.1 Hybrid Business Dashboard

The implementation of the Hybrid Business Dashboard is available as a feature of the BPaaS Design Environment described in Section 5.3.2.1. 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 green rectangles in the component diagram of the BPaaS Evaluation Environment (See Figure 70).

5.3.1.2 Harvesting Engine

The *Harvesting Engine* is the component responsible for the harvesting of information stored in other environments, its semantic enrichment and linking, and its final storage in the *Semantic KB*. In this respect, it is a very critical component that affects the success of all the analysis features offered by the BPaaS Evaluation Environment. This component is shortly described here, but more details can be found in deliverables D3.5 & D3.6 [36][38].

The semantic linking in this component is performed via exploiting the evaluation and OWL-Q (KPI extension) ontologies (see [36]). Multi-tenancy is supported by carefully splitting the content harvested into tenant-specific parts, which are stored in corresponding tenant-specific graphs in the *Semantic KB*. In this respect, this is the main component that realises multi-tenancy in this environment.

The *Harvesting Engine* is not offered in the form of a REST service. On the contrary, it is a Java-based component spawned as a thread that harvests information periodically in order not to overwhelm the CloudSocket system. As such, this component's capabilities are not exposed to the outer world. The main functionalities realised by this component are the following:

(a) harvesting of information from different CloudSocket components - currently, there is support to extract information from the *Cloud Provider Engine*, the *Workflow Engine*, the *Monitoring Engine*, and the (service & software component) registries of the *Repository Manager*;
(b) semantic transformation and linkage of the harvested information based on the Evaluation & OWL-Q ontologies;
(c) multi-tenant-oriented storage of semantically enhanced information into the *Semantic KB*.

The *Harvester* communicates with all components mentioned in the three main functionalities, i.e., the *Cloud Provider Engine*, *Workflow Engine*, *Monitoring Engine*, *Registries-Repository Manager* and the *Semantic KB*. In particular, it only retrieves information from the first four components, while for the last it also stores this information.

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 19 - Details of the Process Mining Engine

Depends:

- *Semantic KB*
- *Workflow Engine*
- *Cloud Provider Engine*
- *Monitoring Engine*
- *Abstract Service, Concrete Service & Software Component Registries*

Architecture design

The *Harvesting Engine* maps to the purple rectangle in the component diagram of the BPaaS Evaluation Environment (see Figure 70). This component has the internal architecture depicted in Figure 71. As it can be seen, the component includes 9 main sub-components: (a) the *Harvester* core, which is the java thread that orchestrates the execution of the rest of the components; (b) the *Deployment Extractor,* which harvests information from the *Cloud Provider Engine*; (c) the *Registry Extractor,* able to harvest service/software component-related information from the respective registries of the *Repository Manager*; (d) the *Workflow Engine Extractor,* responsible for extracting the BPaaS workflow type and instance information from the *Workflow Engine*; (e) the *BPMN Parser,* exploited by the *Workflow Engine Extractor* for the parsing of BPaaS BPMN workflows; (f) the *CAMEL Parser,* which enables to parse and extract valuable information from the BPaaS CAMEL model as well as to transform its requirement and monitoring part into an OWL-Q model; (g) the *Monitoring Extractor,* responsible for harvesting measurement information from the *Monitoring Engine*; (h) the *Marketplace Extractor,* responsible for the harvesting of organisational-oriented information; (i) the *Triple Importer,* which takes care of creating linked data (RDF triples) out of the information harvested that is stored in the *Semantic KB*. More details about the internal architecture of the *Harvesting Engine* can be found at deliverable D3.6 [38].

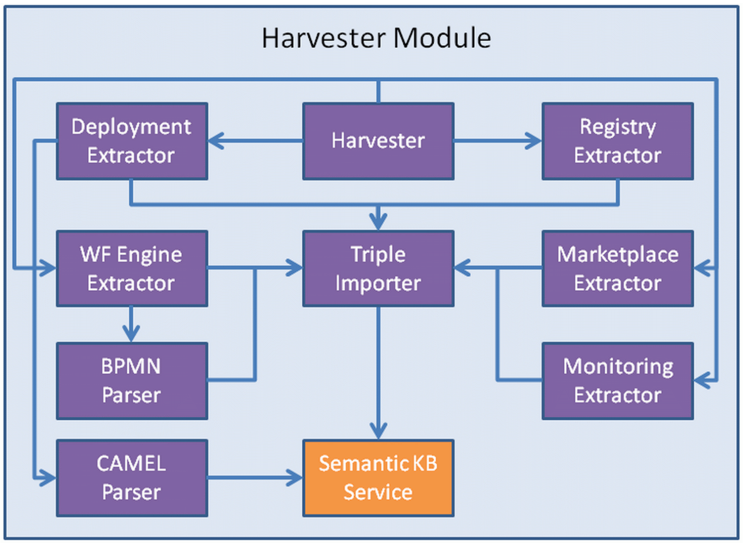


Figure 71: The architecture of the Harvesting Engine

The following table highlights the main functionalities implemented in the Harvesting Engine, while details about the workitems and the respective functionality realisation status can be found at the shared excel file, which has been referenced in Section 2.2. Moreover, details about the interactions required to realise this functionality can be found in D4.1 [37].

In our opinion, this component is not easily replaceable, as it is the glue point between BPaaS evaluation-related and many other CloudSocket components in other environments. As such, the internal logic of that component could be quite hard to reproduce unless there is a deep knowledge of all the components from which this component has to extract information. In addition, there is the need to possess a good level of expertise in semantic technologies to realise this component, due to the requirement to structure and link the harvested information based on the two ontologies considered. However, as this component does not offer any external interface, once such knowledge is obtained and the respective needed logic realised, then it could be easily interchanged.

Functionalities

Functionality	Description	Completed for release	Integrated (which level)
Information harvesting	The component harvests information from multiple components, including the *Cloud Provider Engine*, the *Monitoring Engine*, the *Workflow Engine* and the *Registry*.	Yes (2nd release)	90% but still allowing to fully implement the actual analysis functionalities intended to be delivered by the BPaaS Evaluation Environment
Semantic information transformation and linking	The component semantically transforms the harvested information and links it by considering the Evaluation and OWL-Q ontologies	Yes (2nd release)	Fully Integrated
Tenant-specific storage of the semantically-enhanced information	The component stores in a tenant-specific space / graph the semantically-enhanced information by exploiting the import mechanism of the *Semantic KB Service* in the Semantic KB.	Yes (2nd release)	Almost Integrated

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/common-understanding-wiki/-/wiki/Main/Harvesting+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/harvester/repository/archive.zip?ref=master.

Credentials: A UULM GitLab account is required in order to access the resources.

Instance

The component has been currently deployed at the same VM where the *Semantic KB* has been deployed at the Omistack cloud of UULM and runs as a Java thread, which continuously and timely retrieves information mainly from the Execution Environment, and populates with it the *Semantic KB*. Such a deployment has been done to reduce the latency in the communication between the components and thus accelerate the storage of the harvested information in the *Semantic KB*.

5.3.1.3 Conceptual Analytics Engine

The *Conceptual Analytics Engine* supports the analysis of KPIs by exploiting the semantic information already provided by the *Semantic KB*. Two types of KPI analysis are supported: (a) KPI evaluation and (b) KPI drill-down.

This component is multi-tenant. This means that it is able to serve multiple brokers. This feature has been enabled by carefully separating the content of the *Semantic KB* into tenant-specific parts. In addition, this component supports two modes of KPI evaluation: (a) KPI analysis over raw measurement data spanning different time periods; (b) querying of KPI history to obtain the respective KPI measurements. This has been enabled again via appropriate structuring of the tenant-specific part of the *Semantic KB*. In particular, raw measurement data are inserted into one graph, while specific KPI metric measurements derived by this component are stored into a history-based graph. This separation of evaluation modes has been mainly performed to separate the normal KPI metric querying over the KPI metric history from the exploration of the KPI metric space over raw measurement data, which can be used to derive the most suitable metric for a certain KPI.

This component mainly maps to the functionality of a REST service that provides an API offering the following methods:

(a) one enabling conducting KPI evaluation based on the name of the required KPI and a specific evaluation period - this method enables to store the KPI measurements derived into the history-based graph;
(b) one enabling conducting KPI evaluation based on the specification of a certain KPI metric and its (measurement context) - this method enables the exploration of the KPI metric space to find the most suitable metric for a certain KPI;
(c) one enabling to perform KPI drill-down based on the name of the required KPI and a specific evaluation period;
(d) one enabling to query the history-based graph in order to obtain the measurements that have been derived for a certain KPI over a specific evaluation period;
(e) one enabling the retrieval of those tenants (BPaaS clients) for which we do have measurements for a respective BPaaS and KPI;
(f) one enabling the retrieval of raw KPI metrics from which KPI metric formulas can be constructed and evaluated based on the method in (b);
(g) one enabling to retrieve all KPIs that can be assessed for a certain BPaaS.

The communication between the *Conceptual Analytics Engine* and the *Semantic KB* mainly relies on the posing of semantic SPARQL queries, via a particular method of the *Semantic KB Service*, over the semantic data stored which result in the direct retrieval of the KPI assessment value. In addition, the semantic data import method of the *Semantic KB Service* has been also exploited to insert KPI assessment values into the history-based graph of the current tenant / broker at hand. 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

- *Semantic KB*

Architecture design

This component maps to the yellow rectangle in the component diagram of the BPaaS Evaluation Environment (see Figure 70). This component has the internal architecture depicted in Figure 72. As it can be seen from this figure, the component includes multiple components spanning: (a) one REST service called *Conceptual Analytics Service*, offering the aforementioned API. This component is responsible for calling the *Query Creator* and *Drill-Down Handler* components to initiate the suitable KPI analysis functionality as requested by the broker; (b) the *KPI Handler,* which is responsible for extracting the OWL-Q specification of KPIs and metrics from the Semantic KB; (c) the *Query Creator,* which is responsible for the creation of SPARQL queries to respond to various kind of user requests (e.g. tenant queries, KPI evaluation, and metric evaluation). This component is also responsible for calling the other components involved in the execution path corresponding to the creation of a SPARQL query out of a metric specification or KPI name; (d) the *Resource Accessor,* which is responsible for the retrieval of information from external sources before the KPI metric-to-SPARQL-query transformation takes place; (e) the *SPARQL Transformer,* which is responsible for transforming the OWL-Q KPI metric specification into a SPARQL query; (f) the *Drill-Down Handler,* which is responsible for the answering of KPI Drill-Down requests and exploits the query creation facilities of the *Query Creator*. More details about the internal architecture of the *Conceptual Analytics Engine* can be found at deliverable D3.6 [38] as well as the project wiki[4].

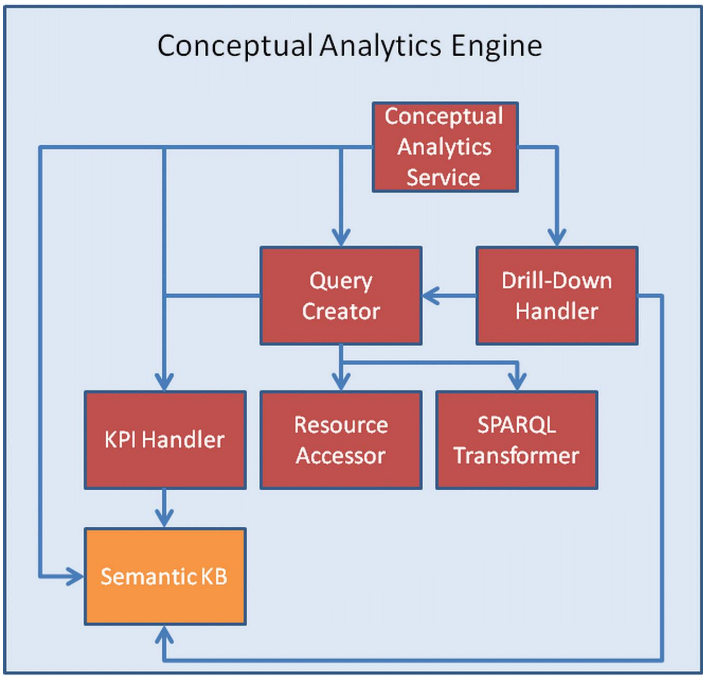


Figure 72: The architecture of the Conceptual Analytics Engine

This component originally was intended to cover all analysis functionalities apart from the process mining one. However, based on the developments in T3.3, it has been decided to be split into three main parts mapping to the 3 different types of analysis intended to be delivered originally by this component, i.e., (a) KPI analysis; (b) best

[4] https://www.cloudsocket.eu/common-understanding-wiki/-/wiki/Main/Conceptual+Analytics+Engine

BPaaS deployment discovery; (c) event pattern detection. As such, the first type of analysis is now supported by this final version of the *Conceptual Analytics Engine*, while the other two types of analysis would have been supported by the *Deployment Discovery Engine* and the *Pattern Detection Engine* (see D3.6 [38]). Due to resource limitations and the corresponding effort needed to integrate the latter engine, in the end, only the *Deployment Discovery Engine* was integrated in the final version of the BPaaS Evaluation Environment.

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 just one from the rest of the BPaaS Evaluation Environment components, the Semantic KB Service, as it was already stated.

Functionalities

Functionality	Description	Completed for release	Integrated (which level)
KPI evaluation (current & historical values)	The component enables the evaluation of a KPI either with respect to the current moment or with respect to a particular time range (thus covering historical KPI assessment information). Evaluation can be done either by the KPI name or by supplying a KPI metric description.	Yes (2nd release)	Fully Integrated
KPI drill-down	The component enables a KPI drill-down analysis in order to discover which low-level KPIs are to blame for the violation of high-level ones	Yes (2nd release)	Fully Integrated
Raw Metric information retrieval	The component enables to retrieve the information of raw metrics which can be used to build up KPI metric formulas (to be used for KPI evaluation purposes)	Yes (2nd release)	Fully Integrated
Tenant information retrieval	The component enables to retrieve those tenants / customers which have purchased a certain BPaaS	Yes (2nd release)	Fully integrated
KPI retrieval	The component will enable the retrieval of all KPIs specified for a certain BPaaS	Yes (2nd release)	Fully integrated

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/common-understanding-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/repository/archive.zip?ref=master.

Credentials: A UULM GitLab account is required in order to access the resources.

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/evaluation/. 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.

Credentials: available on demand.

5.3.1.4 Deployment Discovery Engine

The *Deployment Discovery Engine* aims at discovering the best deployment for a BPaaS based on its execution history or the execution history of other BPaaS that are similar or equivalent to it. This component has been obtained from WP3 (see D3.6 [38]) and has been integrated in this final prototype. More details about this component can be found at deliverables D3.5 & D3.6 [36][38].

This component is also multi-tenant and exposes its functionality in the form of a REST service, called *Deployment Discovery Service*, which provides an API with the following main method: discovery of best deployments for a BPaaS, which is given as input along with information about the period over which the analysis can be performed.

The *Deployment Discovery Engine* communicates mainly with the *Semantic KB* in order to pose queries aiming at: (a) obtaining all workflows and their tasks to assist in the matching of tasks/workflows for discovering the similarity between BPaaSes; (b) retrieving the execution history for all the relevant BPaaSes (the current one and all those that are similar / equivalent to it).

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 22 - Details of the Process Mining Engine

Depends

- *Semantic KB*

Architecture design

The *Deployment Discovery Engine* maps to the green rectangle in the component diagram of the BPaaS Evaluation Environment (see Figure 70). This component has the internal architecture depicted in Figure 73. As it can be seen, this engine comprises the following 5 main components: (a) the *Deployment Discovery Service,* which, as indicated, is a REST service exposing the best BPaaS deployment discovery functionality; (b) the *Deployment Discovery Orchestrator*, the core component in this architecture, which takes care of orchestrating the functionality exposed by other components in order to finally produce the ranked list of deployments for a certain BPaaS; (c) the *Semantic Information Extractor,* which poses queries over the *Semantic KB* in order to obtain workflow information as well as the execution history of the BPaaS; the first information kind is transformed it into a set of facts, which are entered in the *KB*; (d) the *KB,* which is responsible for the firing of rules over its content in order to produce the BPaaS (workflow & task) similarity facts; (e) the *Utility Calculator,* which is responsible for computing the utility for each relevant BPaaS (workflow) deployment, according to its execution history extracted by the *Semantic Information Extractor*, as well as ranking all deployments based on this utility.

Please note that in the corresponding research prototype from WP3 that has been integrated in the BPaaS Evaluation Environment, there was also another component called *Smart Service Discovery & Selection Tool*. Such a component could be integrated into the final architecture of this *Deployment Discovery Engine* while it could also be used in the BPaaS Allocation Environment for automatic service selection purposes. However, due to some technical issues and the lack of time and resources, this component was not finally integrated and this is the main reason that it has been left out from the architectural image below.

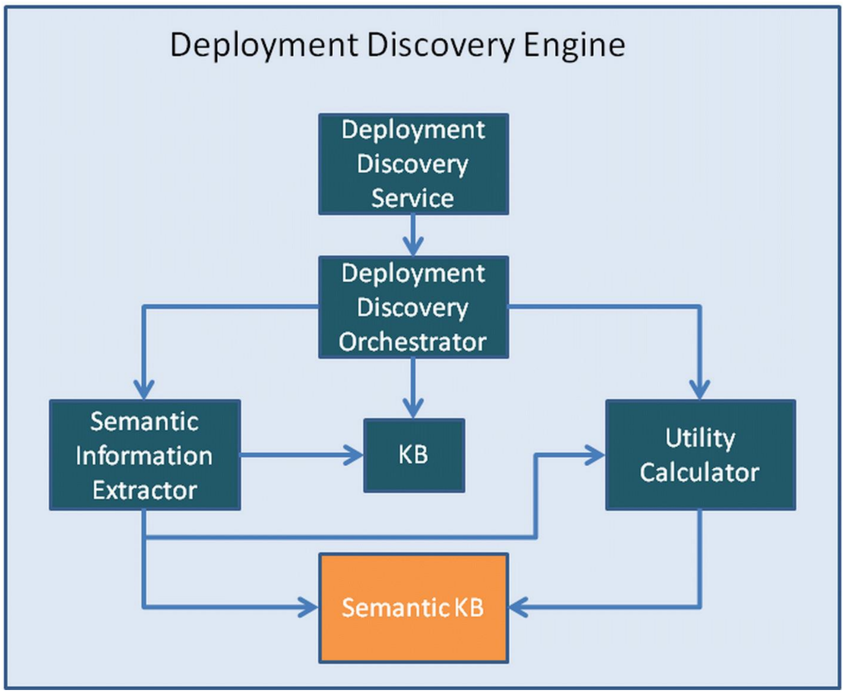

Deployment Discovery Engine
Deployment Discovery Service
Deployment Discovery Orchestrator
Semantic Information Extractor
KB
Utility Calculator
Semantic KB

Figure 73: The architecture of the Deployment Discovery Engine

Functionalities

Functionality	Description	Completed for release	Integrated (which level)
Deployment Discovery	The component enables to find all deployments for a certain BPaaS and rank them in descending order based on their overall utility	Yes (2nd release)	Fully Integrated

Table 23 - Functionalities of the Deployment Discovery 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/common-understanding-wiki/-/wiki/Main/Deployment+Discovery+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-dd/repository/archive.zip?ref=master.

Credentials: A UULM GitLab account is required in order to access the resources.

Instance

The component has been currently deployed in a VM at the Omnistack cloud of UULM and is available at the following URL: http://134.60.64.222:8080/evaluation-dd/. 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.

Credentials: available on demand.

5.3.1.5 Process Mining Engine

The *Process Mining Engine* has the goal of exploiting process mining algorithms over the execution history of a BPaaS workflow in order to find discrepancies between what has been designed and what is being exhibited in practice. This component has been obtained from WP3 (see D3.6 [38]) and has been integrated in this final prototype.

Similarly to the case of the *Conceptual Analytics Engine*, the *Process Mining Engine* is a multi-tenant component where, again, tenant means a broker. This has been enabled thanks to the *Harvesting Engine* component, as has been stated in Section 5.3.1.2.

This component mainly maps to the functionality of a REST service that provides an API offering the following main methods:

(a) One enabling to retrieve the process mining algorithms that can be exploited.
(b) One actually enabling to perform the process mining via exploiting a certain algorithm, whose name is given as input, over the history of a certain BPaaS (whose name is also given as input). The broker has

the capability also to define the period of analysis / process mining (from now until a specific time point in the past), to restrain the analysis over a specific tenant (whose name can be given as input), and to explicate whether the semantic label or the task name should be used for the process mining, where the first case maps to supporting a sort of semantic process mining.

The *Process Mining Engine* communicates with: (a) the *Semantic KB* in order to pose SPARQL queries over it, aiming at drawing the corresponding execution history information needed for the mining; (b) the *Hybrid Business Dashboard,* which calls one of its two API methods available (mapping to the two aforementioned functionalities).

Details of the component:

Type of ownership	Creation
Original tool	ProM[5] Process Mining Framework
Planned OS License	Mozilla Public Licence (MPL) 2.0. [28]
Reference community	ADOxx community
Lead Partner	FORTH

Table 24 - Details of the Process Mining Engine

Depends:

- *Semantic KB*

Architecture design

The *Process Mining Engine* maps to the blue rectangle in the component diagram of the BPaaS Evaluation Environment (see Figure 70). This component has the internal architecture depicted in Figure 74. As it can be seen, the component comprises 5 main sub-components: (a) one REST service called *Process Mining Service*, offering the aforementioned API; (b) the *Process Mining Orchestrator,* which takes care of orchestrating the process mining work by invoking other components in the architecture with the right order; (c) the *Query Creator,* which is responsible for extracting the BPaaS workflow execution history from the *Semantic KB*; (d) the *Log Creator,* which is responsible for transforming the execution history extracted into a process log conforming to a particular format; (e) the *Process Mining Library,* which includes the implementation of certain process mining algorithms that can be executed on-demand. More details about the internal architecture of the *Process Mining Engine* can be found at deliverable D3.6 [38].

[5] www.promtools.org

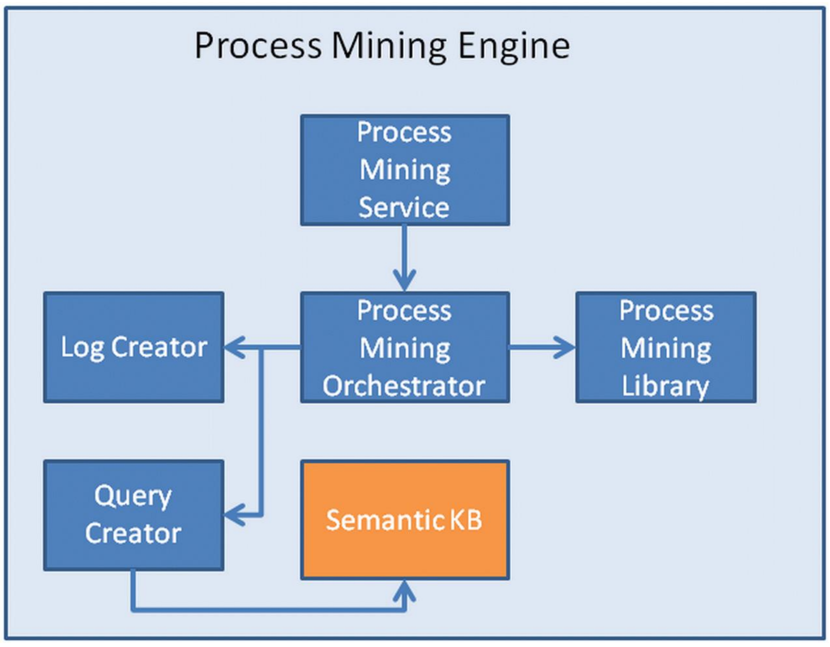


Figure 74: The architecture of the Process Mining Engine

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 sole component it connects to. In particular, as already stated, it just retrieves the BPaaS workflow execution information from the *Semantic KB* and then transforms it into a log that can be exploited by one or more process mining algorithms. As the *Semantic KB* exposes a REST-based API that abstracts away from any implementation particularities, the replacement of the *Process Mining Engine* component would only have to re-use this API in order to realise accordingly its functionality.

Functionalities

Functionality	**Description**	**Completed for release**	**Integrated (which level)**
Process Mining	The component enables to execute a certain process mining algorithm over the execution log of a particular BPaaS	Yes (2nd release)	Fully Integrated
Process mining algorithm retrieval	The component enables to retrieve the information of the process mining algorithms supported for execution over process logs.	Yes (2nd release)	Fully Integrated

Table 25 - Functionalities of the Process Mining 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/common-understanding-wiki/-/wiki/Main/Process+Mining+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-pm/repository/archive.zip?ref=master.

Credentials: A UULM GitLab account is required in order to access the resources.

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/evaluation-pm. 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.

Credentials: available on demand.

5.3.1.6 Semantic Knowledge Base

The *Semantic Knowledge Base* (*Semantic KB*) hosts semantic data that can be used for BPaaS analysis purposes. It has been implemented in the form of a REST service on top of a semantic *Triple Store* realised via the Virtuoso[6] open-source component. This service enables the complete management of such semantic data via exploiting the Sesame interface for semantic data management over Virtuoso. Such management is also facilitated via certain capabilities of the Virtuoso Triple Store Server which span the following:

- support for the SPARQL query language
- support for the updating of semantic data via SPARUL statements
- semantic data import and export support via the offering of various built-in functions

Details of the component:

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

Table 26 - Details of the Semantic Repository

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

Depends

- This is a totally independent component from the rest of the components of the BPaaS Evaluation Environment

Architecture

This component maps to the red rectangle in Figure 70 (BPaaS Evaluation Environment component diagram). As already explained, this component is mapped to two sub-components, namely the Semantic KB Service and the (Virtuoso) Triple Store. This is also depicted in Figure 75.

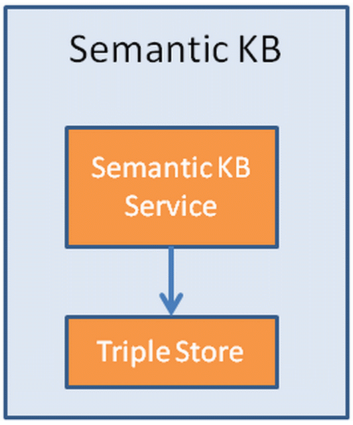


Figure 75: The architecture of the Semantic KB

The component has been fully integrated in the CloudSocket prototype as: (a) it can now be completely populated with all the necessary information by the *Harvesting Engine*; (b) it is exploited by the *Conceptual Analytics Engine*, the *Deployment Discovery Engine* and the *Process Mining Engine* in order to support the various types of analysis offered by these engines. In principle, its query functionality could be exploited by the *Hybrid Business Dashboard* in case an experienced user with knowledge of the semantic technology desires to explore its content in order to, e.g. see what kind of information pieces could be suitable for the realisation of a certain KPI metric.

This component can be easily replaced by another component developed externally with respect to the CloudSocket organisation. The only requirement for such a replacement is that the same REST interface should be realised as it is currently exploited by many other components in the BPaaS Evaluation Environment architecture (including the *Harvester*, the *Process Mining, Deployment Discovery* and *Conceptual Analytics* engines).

By considering the internal architecture of this component, also its internal components could be replaced. The first could be replaced by another REST service that exposes the same interface. The second, by another Triple Store implementation that supports the Sesame interface. The latter requirement (about the Sesame interface) would of course hold if we replace just the second and not also the first component. In a full overall component replacement, in principle, the respective developing organisation would have the freedom to select the most suitable semantic technologies that suit its own purposes.

Functionalities

Functionality	Description	Completed for release	Integrated (which level)
Management of semantic data	The component enables the management (query, update, import, and export) of the semantic data stored into its underlying Triple Store by offering different interfaces mapping to the realisation of this management functionality.	Yes (1st release)	

Table 27 - Functionalities of the Semantic Repository

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/common-understanding-wiki/-/wiki/Main/Semantic+Knowledge+Base.

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/skb/repository/archive.zip?ref=master.

Credentials: A UULM GitLab account is required in order to access the resources.

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/skb. Internally, as being a REST service, a servlet container (tomcat) has also been deployed in the same VM in order to host it.

Credentials: available on demand.

5.3.2 BPaaS Design Environment

The Design Environment allows the CloudSocket Broker to construct the BPaaS Design Packages for the Business Process services 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) and semantic annotations.

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

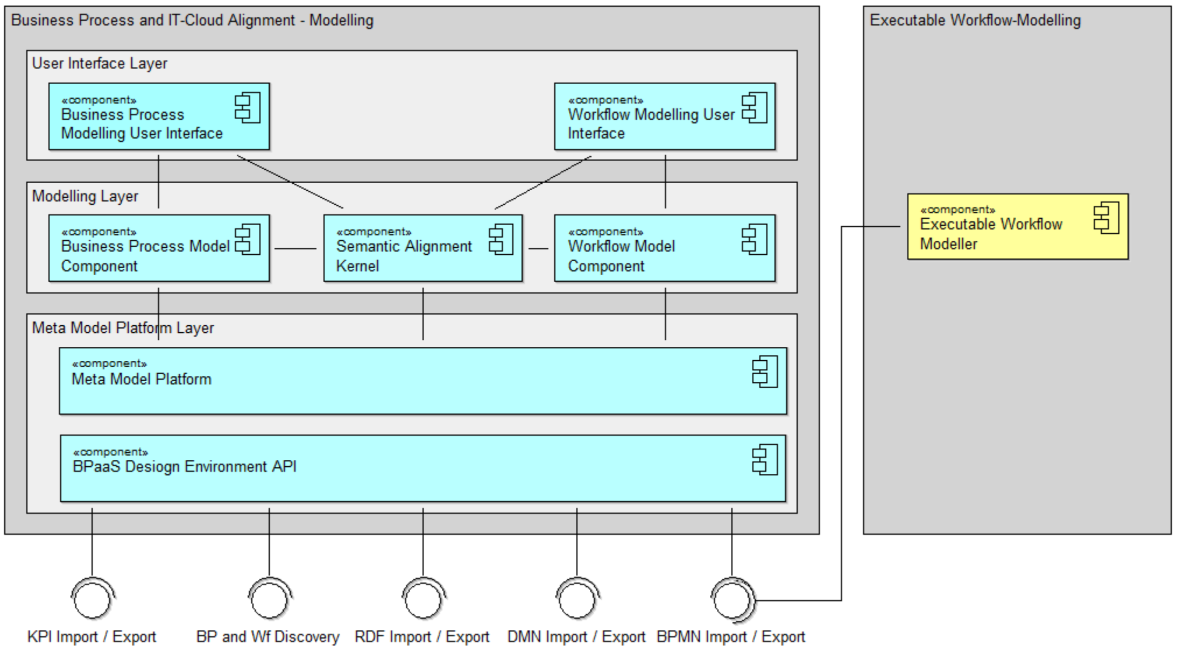

Business Process and IT-Cloud Alignment - Modelling
User Interface Layer
«component» Business Process Modelling User Interface
«component» Workflow Modelling User Interface
Modelling Layer
«component» Business Process Model Component
«component» Semantic Alignment Kernel
«component» Workflow Model Component
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 76 - The component diagram of the BPaaS Design Environment

5.3.2.1 BPaaS Design Tool

The BPaaS design Tool has been created on the basis of the CloudSocket meta-model specified in D3.1 [24]. This meta-model defines all the aspects required in order to the BPaaS design functionality, which, in this specific context, are:

- Domain-specific Business Processes applying the BPMN 2.0 standard.
- Execution Workflows applying the BPMN 2.0 standard but allowing also custom extensions.
- Decision Models applying the DMN 1.1 standard.
- The company structure as defined in D3.1 (no standards exist for that).
- Explanatory 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, which 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 the 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 a service (SaaS)
Reference community	ADOxx
Lead Partner	BOC

Table 28 - Details of the Design Environment

Comprises:

- User inteface for the modeling of the BPaaS Design Package: the tool is a web based solution built on top of the ADOxx environment.
- Hybrid Business Dashboard: as a result of the CloudSocket Evaluation Environment.
- REST API in order to allow other environments to interact with the BPaaS Design Environment.

Architecture design

The general architecture of this component can be viewed by checking the components coloured in blue, in the Figure 76 above as the BPaaS 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 their 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 modelling component.
- The corresponding (Web-)GUI realizes the user interaction features, to manipulate a model.
- The Hybrid Business Dashboard is a web portal that gives the possibility to monitor the current status of all the KPIs defined in a BPaaS Design Package. A user-friendly interface has been provided in order to explore the KPIs and check if they are in line with the allowed value ranges or not. In the background, the *Conceptual Analytics Engine* in the BPaaS Evaluation Environment (see Section 5.3.1.3) is exploited in order to derive the actual KPI metric values to be examined.
- API Interfaces enable the access to the BPaaS Design Environment, and 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.

In the following, a more detailed explanation of the components is provided:

- 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, thus providing the possibility to create models for the defined meta-models. This component is also responsible for efficiently storing and retrieving models (of a BPaaS Design Package) defined in the BPaaS Design Environment.

- 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. A model can be listed and exported in different formats in relation to its respective 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 any other kind supported.
  - Image export: gives 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: enables to export a Business process model or a workflow in standard BPMN 2.0 format.
  - BPMN import: gives the possibility to import a BPMN 2.0 standard model into the BPaaS Design Environment.
  - XML export: enables to export any kind of model in a generic xml format specifically relevant for the BPaaS Design Environment.
  - XML import: gives the possibility to import into the BPaaS Design Environment a model described in its specific XML format.
  - DMN export: enables to export a decision tables model in standard DMN format.
  - RDF import: gives the possibility to import a model defined in standard RDF format into the BPaaS Design Environment.
  - BPaaS Design Package export: enables the retrieval of individual BPaaS design packages.

  This REST API is available at the following URL:
  https://www.cloudsocket.eu/BPAASDesigner/rest/cloudsocket/API/

- The Business Process Modelling User Interface is the entry point for the Business Process Designer in order to model a BPaaS Package. The product version is a web app that can be accessed with any browser. In this version, all the features of the research items have been replicated and new ones (like the new simulation engine, the formal verification and the cloud readiness quality check) were also introduced. In particular, the following features have been introduced:
  - Cloud readiness check: This feature gives the possibility to evaluate the readiness of the business process for being deployed in a cloud environment. It is based on a questionnaire that the modeller must answer for the whole process and for every task in the process so as to return a score over them. This feature is also present in the Workflow Modelling User Interface.
  - Formal Verification check: This feature maps to applying formal verification methods over the business process in order to verify structural properties, like the absence of deadlock or livelocks, and the presence of a specific path. In particular, it uses a state-of-the-art model-checking technique over a petri-net representation of the business process with properties specified in Linear Temporal Logic. This feature is also present in the Workflow Modelling User Interface.
  - Simulation: This feature enables to estimate costs, times and other relevant parameters of a business process performing a simulation over it. This feature is also present in the Workflow Modelling User Interface.

- 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 of the workflow is performed in the Workflow Modelling Component, while its configuration as an executable workflow is performed in a separate, but well integrated, “Executable Workflow Designer”, described in Section 5.3.2.2. 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 workflow is so first designed and semantically annotated in the Workflow Modelling Component, and then exported in the separate tool for adding all the references to executable services. Hence, this component provides user management, model management and model design features for workflows.

- The Semantic Alignment Kernel is a light-weight component for annotating modelling elements with the BPaaS Ontology.

Functionalities

The Table 29 indicates the covered functionalities and their status:

Functionality	Description	Completed for release	Integrated (which level)
Create/Edit BPaaS packages	This functionality allows a broker to create new BPaaS packages or edit his existing packages by creating/changing models of – Business Processes – Workflows – Decisions – Key Performance Indicators and cause/effect relations – Bundle/Design package overviews	Yes (1st release)	Complete lifecycle
Hybrid Dashboard	This functionality allows a designer to monitor the status of the KPIs defined on every BPaaS Package.	Yes (2nd 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 designed models in different formats like BPMN, DMN or RDF (according to their respective formats) and to generate images for them.	Yes (1st release)	Complete lifecycle.
Cloud readiness check	This functionality enables to evaluate the readiness of the business process for being deployed in a cloud environment	Yes (1st release)	Complete lifecycle.
Verification	This functionality applies formal verification methods over the business process in order to verify behavioural properties, like the absence of deadlock or livelocks and the correctness of a path.	Yes (1st release)	Complete lifecycle.
Simulation	This functionality gives the possibility to statistically evaluate a Business Process or an Executable Workflow in terms of costs, time and unexpected behaviours	Yes (1st release)	Complete lifecycle.

Semantic Lifting	This functionality enables the semantic lifting of business processes and workflows (templates). It includes a questionnaire that enables to: • show the current question to be answered • search for the term that is already in the ontology with realtime results. • guide the user in service selection through the set of questions by following the logic that is implemented in the web service	Yes (2nd release)	Complete lifecycle.

Table 29 - 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/common-understanding-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 at: https://www.cloudsocket.eu/BPAASDesigner

Credentials: available on demand.

Most of the source code of the BPaaS Design Environment is not available apart from the case of the integrated research prototype which is available in the following links:

- Questionnaire User Interface: https://github.com/BPaaSModelling/Semantic-Annotation-Questionnaire-WebApp
- Questionnaire Web Service: https://github.com/BPaaSModelling/Questionnaire-WebService

5.3.2.2 *Executable Workflow Modeler*

This component is responsible for the modeling of the executable workflow. Please have a look at Section 4.3.3.1 for a detailed description of this component. 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. It is identified in the BPaaS Design Environment at Figure 76 baring the colour of yellow.

5.3.3 BPaaS Allocation Environment

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

5.3.3.1 Allocation Environment Architecture

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

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

Figure 77 shows the detailed internal architecture of the Allocation Environment.

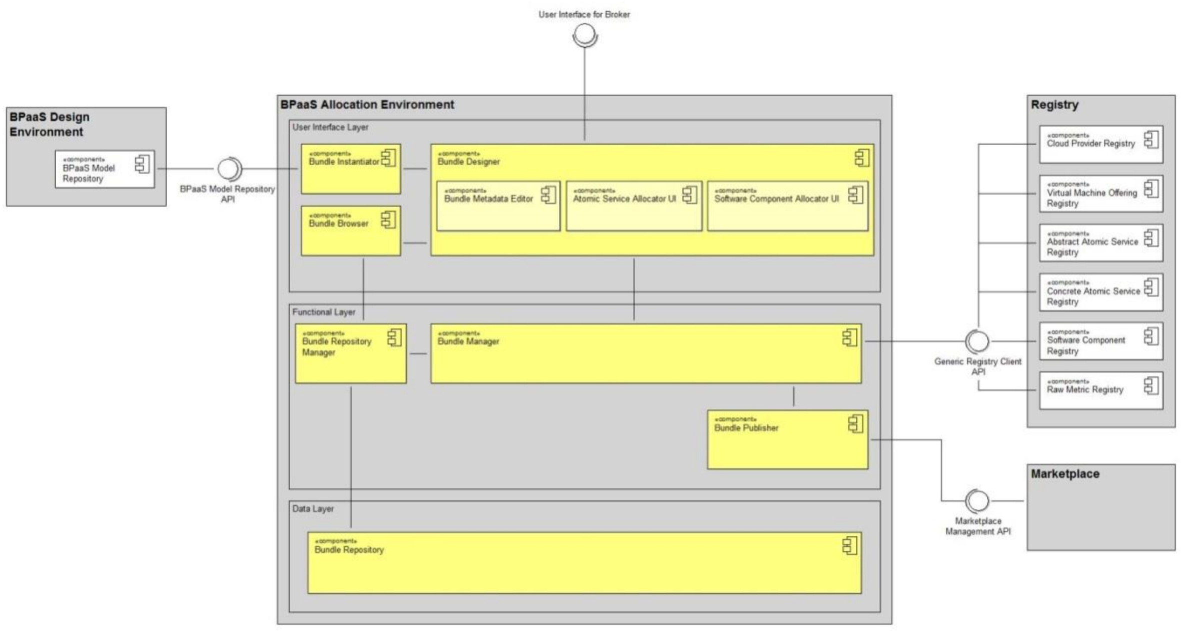

User Interface for Broker
BPaaS Design Environment
«component» BPaaS Model Repository
BPaaS Model Repository API
BPaaS Allocation Environment
User Interface Layer
«component» Bundle Instantiator
«component» Bundle Browser
«component» Bundle Designer
«component» Bundle Metadata Editor
«component» Atomic 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 77 - 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 [37].

5.3.3.2 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 concrete Atomic SaaS Services, realising the functionality of BPaaS workflow (service) tasks, and IaaS/PaaS services supporting the operation of internal SaaS components of the BPaaS.

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 BPaaS Execution Environment (to deploy, execute, monitor and adapt a BPaaS) and by the Marketplace (to allow the BPaaS Customer to effectively search the BPaaS bundles of interest) for a certain BPaaS. Such information includes:

- The Business Process Model (in BPMN 2.0).
- The Workflow Model (in BPMN 2.0).
- Atomic Service allocation (part of CAMEL).
- Software Component allocation (part of CAMEL).
- KPI model (completed from the BPaaS Design Package and now part of CAMEL).
- Service Level Agreement (SLA).
- Marketing Metadata.
- Adaptation Rules (part of CAMEL).

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 30 - 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 Figure 77 above, as the Allocation Environment is composed only by the Allocation Tool.

The Allocation Tool provides a web application and follows a multi-tier architecture in which 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 BPaaS Design Package of the BPaaS Design Environment. The tasks performed by the Bundle Instantiator are:
    - Browse the workflow models mapping to BPaaS Design Packages already created in the BPaaS 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) all 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 the allocation for both 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 CloudSocket 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 Bundle Repository and to delete a bundle from the Bundle Repository.

  The Bundle Repository Manager supports multi-tenancy, where the tenant is the CloudSocket Broker. Indeed, this component has the responsibility to let each CloudSocket Broker to access only its own bundles as well as to allow only legitimate state transitions and actions to be executed over the bundles of that CloudSocket 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 BPaaS 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 (e.g. Software Component Registry, Cloud Provider Registry and so on). 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 that 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 BPaaS 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 BPaaS 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 BPaaS Bundle already published in the Marketplace.
  Tasks performed by the Bundle Publisher: this server-side component is responsible to publish and update a BPaaS Bundle into the Marketplace.

Functionalities

The Table 31 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 own existing bundles	Yes (1st release)	Complete lifecycle.
Create/Edit BPaaS Bundle	This functionality allows a broker to create new bundles or edit existing ones	Yes (1st release)	Complete lifecycle
Multi-Tenancy Support	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	Yes	Complete lifecycle.

	BPaaS bundle, like title, description, price, categories and tags	(1st release)	
Allocate Software components	This functionality allows the browsing and selection of virtual machines and their assignment to each software component (internal service) mapping to a service task in the BPaaS workflow	Yes (1st release)	Complete lifecycle
Allocate Atomic services	This functionality allows the browsing and selection of concrete service endpoints and their assignment 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	Yes (2st release)	Complete lifecycle.
Integration with Repository Manager	The environment is currently integrated with software component, cloud provider and atomic service registries of the Repository Manager	Yes (2st release)	Complete lifecycle.
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 machines (if the CAMEL will support this feature)	Partially (2nd release)	Integrated only horizontal scaling. Other further adaptation rules can be easily integrated.
Adaptation rules for Atomic services	This functionality will allow the definition of adaptation rules for an atomic service in terms of switching the endpoint of the services	No (2nd release)	CAMEL support currently missing for this feature.
Creation of Metric Conditions	This functionality will allow defining KPIs based on raw or composite metrics (which have been already defined for other KPIs). The KPIs will be used to define SLO conditions and adaptation rules	Yes (2nd release)	Complete lifecycle.
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.	Yes (2nd release)	Complete lifecycle.
Publish/update BPaaS bundle in the Marketplace	This functionality allows the publication or updating of the BPaaS bundles on the Marketplace	Yes (1st release)	Complete lifecycle.

Integration with YMENS IdM	This functionality allows to log in into the Allocation Tool using the YMENS IdM credential	Yes (2nd release)	Complete lifecycle.
Enabling the Guest Account	This functionality allows to use the Allocation Tool with guest account for demonstration purpose only. The guest user cannot publish in the Marketplace and can see only the bundles created from guest account.	Yes (2nd release)	Complete lifecycle.

Table 31 - Functionalities of the Allocation Tool

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/common-understanding-wiki/-/wiki/Main/Allocation+Environment+Components

Instances

There is available an instance of the BPaaS Allocation Environment deployed on the Fhoster infrastructure https://hs21.fhoster.com/cloudsocket/Allocation_demo to integrate the final prototype with other dependent environments: BPaaS Design Environment and Marketplace. The same instance provides also the access to guest users by disabling the feature to publish bundles into the Marketplace:

https://hs21.fhoster.com/cloudsocket/Allocation_demo/ApplicationResource.jsp?application=Allocation&locale=en

Credentials: available on demand using the YMENS IdM.

5.3.4 BPaaS Marketplace

For CloudSocket Brokers, the Marketplace's role is the main place where BPaaS bundles are offered to the BPaaS Customers. When a BPaaS bundle is bought through the marketplace, an automatic provision process is started. The general description of the BPaaS Marketplace is detailed in the Section **Fehler! Verweisquelle konnte nicht gefunden werden.**, and only the Marketplace component will be described from the point of view of the Broker functionalities.

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 enables brokers to:
    - View Dashboard information through the Portal.
    - Create company for their customers.
    - Create users associated with registered companies.
    - View a list of external tools that can be accessed from 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 as mentioned above.

Comprises:

- Shop – main module that manage shopping experience, including checkout and payment
- Portal – module that manage non-shop related interactions of the users (account details, create company, create user, access external tools, view reports and dashboards)
- Service API – provides data for Web UI and core workflow functionality (Allocation Environment – product endpoint, Execution Environment – provision endpoint)
- Identity provider – ensures M2M and U2M authentication and authorization flows

Depends on:

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

Check Section 4.3.1-BPaaS Marketplace for Architectural information and component responsible.

Functionalities

The Table 32 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, to pay the checkout items using PayPal and Bank Order, to search through items by tags, status and price	Yes (2nd 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, Payment API)	Yes (1st release)	Complete lifecycle
Portal	The module enables enables brokers to acces a list of external tools, create company and users, give users access to different reports and statistics related to orders.	Yes (2nd release)	Complete lifecycle
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 Complete U2M with Execution Environment

Table 32 - Functionalities of the Marketplace

The manuals and instances are described in the Section 4.3.1.

6 CONCLUSIONS

The Final CloudSocket prototype has been developed based on the final architecture D4.5 [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-specific demonstration is explained and then the respective relevant environments are detailed. Besides, the documentation has been produced for each environment in the wiki [19], which is aligned with the content of this deliverable, allowing the readers to decide the level of detail (small - current deliverable, high - wiki) that they want to obtain depending on their needs.

The prototype includes many of the research results in WP3, such as the introduction of cloud orchestration based on the CAMEL standard, the BPaaS monitoring & adaptation approaches and the semantic lifting of the business process model. Such integration has enabled the CloudSocket platform to exhibit extra, added-value features that make it even more attractive for being exploited by potential CloudSocket brokers.

Despite the project task are finished, the different prototype components are alive and requires, in the future, support for brokers, like bug fixing and minor adaptation of current functionalities, which as consortium we planned to provide.

7 REFERENCES

[1] D4.5 - CloudSocket_D4.5_Final-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/demonstration-end-user-perspective-2
[31] PaaS orchestration and adaptation: https://www.youtube.com/watch?v=aGtQ210wih8
[32] D3.3 Allocation-Execution-Environment-Blueprints https://www.cloudsocket.eu/deliverables
[33] D3.4 Allocation-Execution-Environment-Prototypes https://www.cloudsocket.eu/deliverables
[34] D4.9 CloudSocket Integration Report
[35] Domaschka, J., et al. "Cloudiator: a cross-cloud, multi-tenant deployment and runtime engine." 9th Symposium and Summer School on Service-Oriented Computing. 2015.
[36] D3.5 BPaaSMonitoring and Evaluation Blueprints https://www.cloudsocket.eu/deliverables
[37] D4.1 - CloudSocket_D4.1_First-CloudSocket-Architecture: https://www.cloudsocket.eu/deliverables
[38] D3.6 - BPaaS Monitoring and Evaluation Prototypes https://www.cloudsocket.eu/deliverables
[39] D5.4 - CloudSocket BPaaS Execution Environment at user site https://www.cloudsocket.eu/deliverables
[40] D4.2/D4.3/D4.4 First BPaaS Prototype at user site https://www.cloudsocket.eu/deliverables

[41] Sending Invoice Handbook
https://site.cloudsocket.eu/documents/251273/0/Sending+Invoice+handbook/7d07326b-2331-4737-9507-bf29375f68d5

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
- M2M: Machine to machine
- OSGI: Open Service Gateway Initiative
- QoE: Quality of Experience
- PaaS: Platform as a Service.
- R2RML: RDB to RDF Mapping Language
- REST: Representational State Transfer
- SaaS: Software as a Service
- SLA: Service Level Agreement
- SLO: Service Level Objective
- SPARUL: SPARQL/Update
- SPARQL: SPARQL Protocol and RDF Query Language
- TSDB: Time-Series Database (TSDB).
- U2M: User to machine
- UI: User interface
- VM: Virtual Machine
- WSAG: WS-Agreement