D6.7

Symbiotic SOA- EDA Platform for Real-Digital World Interconnection First

Document Owner:	TXT
Contributors:	FINC, IBM
Dissemination:	Public
Contributing to:	WP6
Date:	20.05.2016
Revision:	V1.0

Contents

List of Figures

Executive Summary

This is the document accompanying the release of the D6.7 prototype.

This software release closes the first iteration of the work in WP6 providing the architecture comprising all the WP6 elements complementing it with one additional components to show the events generated. This deliverable is important to show the SOA – EDA integration realised by EDA events management and storage triggering services in the software service environment.

The overall integrated architecture is composed by five main components; things (e.g.: sensors, actuators, smart objects, etc.) in the real world are interfaced and handled by the IOT broker component, supported by a complex event processor able to provide reasoning capabilities like pattern recognition, threshold monitoring, temporal correlation, etc. The IOT broker is powered with an Attribute Based Encryption (ABE to provide privacy). Events are stored into the events database that is the major integration points. On the right in the picture, the digital world viewer is visible and composed by two elements: the former is the event viewer that allow the interaction with events and services managed by users (mainly pull services) while the latter SEMed provides the interoperability with non-human services storing, for example, data in different kind of databases (mainly push approach).

Most of the components have been provided, as standalone components, in previous deliverables (D6.1, D6.3 and D6.5). A completely new tool for the digital world manager is the EventCloud component. The purpose of the tool is twofold: first of all is to allow people at runtime to access and interacts with events and connect several other services in a visual way. Second is to allow people to program themselves the interfaces and the interactions among them at modelling time. The tool is implemented by a data mashup interface based on social gadgets. The openness of the architecture allows technicians to add any kind of gadget/widget to interact with database, services and, of course, events defining the GUI, the behaviour of the piece of software as well as the Input/Output interfaces. End-users (usually not technical) are free to define, for its own account, which gadget to view, where to view them and which interaction are activated between gadgets.

About the real world manager is completed in this phase by the modeller of the graphical editor for the creation of the access policy.

The software is presented in this document using an example freely taken from the PIACENZA use case. The software as WP6 architecture is going to be instantiated into pilots in WP9 and run in WP10; first target of the application is PIACENZA pilot as visible by the example (realistic but not real) taken from that domain.

1 Introduction

This chapter will introduce the software release including the purpose of the software, the positioning of this deliverable in the context of the project and the applicable documents.

1.1 Introduction to the software release

This software release closes the first iteration of the work in WP6 providing the architecture comprising all the WP6 elements complementing it with one additional components to show the events generated. This deliverable is important to show the SOA – EDA integration realised by EDA events management and storage triggering services in the software service environment.

Here below the description of WP6.4 DOA

> In order to give real-time insights to ongoing factory circumstances and situations, an interconnection between Real and Digital World needs to be integrated in the overall SOA-EDA platform. The aim is to detect which services are to be activated based on real-time events. The complex event definitions for the use case scenarios will be articulated as part of, and in collaboration with, the design of the scenarios.

1.2 Positioning of the deliverable in PSYMBIOSYS

This deliverables integrates and complements what has been described in three deliverables in one single platform. Inputs are from D6.1 Privacy, Security and Data Protection in product-service design First (FINC - M12), D6.3 Real World Event Driven Architecture and Events Processing First (IBM – M12) and D6.5 Digital World Service Oriented Architecture and Semantic Mediators for Data Access First (BIBA – M12). The output of this software release will be integrated into the overall Platform in WP9.2 and instantiated into end-users scenarios in WP10.

1.3 Applicable Documents

- PSYMBIOSYS DOA describing WP and Deliverable goals
- PSYMBIOSYS D6.1, D6.3 and D6.5 providing components built on M12 and integrated into the current architecture
- PSYMBIOSYS D9.1 providing the overall WP6 architecture

2 Overview of software Components Integrated

The architecture goal is to bring all the elements implemented in WP6 together in a comprehensive and useful architecture to solve the SOA-EDA interconnection.

In the picture here below the overall architecture is depicted. On the left the Real World Manager is visible; it interacts with the things (e.g.: sensors, actuators, smart objects, etc.) in the real world and handle them. Major components are the IOT broker and the complex event process; in particular the IOT broker is powered with an Attribute Based Encryption (ABE). Events are stored into the events database that is the major integration points. On the right in the picture, the digital world viewer is visible and composed by two elements: the former is the event viewer that allow the interaction with events and services managed by users (mainly pull services) while the latter SEMed provides the interoperability with non-human services storing, for example, data in different kind of databases (mainly push approach).

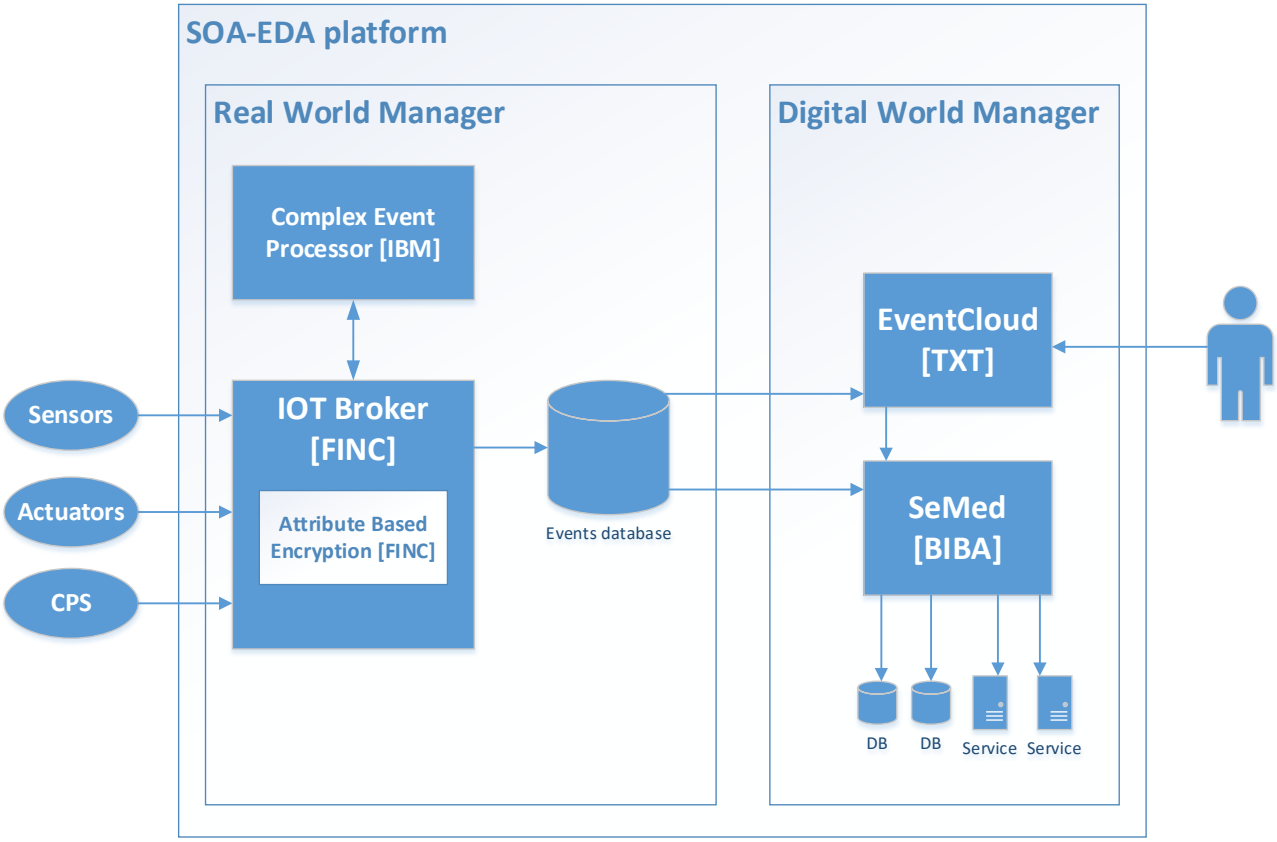

Figure 1: WP6 SOA-EDA platform architecture

3 Digital World Manager – EventCloud

This chapter focuses on the description of the "event viewer" component released. The section starts summarising the overall information about the software released (description, overall data, functionalities and architecture), after that technical information are reported about architectural stack, technical manual for installation and licensing (including third parties components). Finally, the user manual and the conclusions and future steps closes the chapter.

3.1 Software Description

3.1.1 Overall Data

Item	Value
Component Name	EventViewer
Software version	v1.0
Reference workpackage	WP6.4
Responsible Partner	TXT
Contact person	Paolo Civardi (paolo.civardi@txtgroup.com)
Source control	http://demos.txt.it:8096/wp6/platform/
Short Description	This software allows to create

3.1.2 Purpose of the tool

The purpose of the tool is twofold: first of all is to allow people at runtime to access and interacts with events and connect several other services in a visual way. Second is to allow people to program themselves the interfaces and the interactions among them at modelling time. The tool is implemented by a data mashup interface based on social gadgets. The openness of the architecture allows technicians to add any kind of gadget/widget to interact with database, services and, of course, events defining the GUI, the behaviour of the piece of software as well as the Input/Output interfaces. End-users (usually not technical) are free to define, for its own account, which gadget to view, where to view them and which interaction are activated between gadgets.

3.1.3 Summary of Functionalities

Modelling time:

- Add a gadget on the marketplace
  - Events
  - services
- Create a view
- Add a gadget to a view
- Resize gadget
- Position gadget
- View connections
- Modify (graphically) the gadgets interactions

Runtime

- Run the view events on gadget
- Run the service view on gadget
- Click on gadget pushing another gadget update

3.2 Technical Information

3.2.1 Internal Architecture

The architecture is based on the software "Application Mashup – Wirecloud" offering an end-user centred web application mashup platform. The software has been originally developed by the Universidad Politécnica de Madrid (UPM) and finalised in the latest version under the FI-PPP framework in the FIWARE project released in the "Applications/Services and Data Delivery" chapter. The original download is available in the FIWARE catalog at the following address: http://catalogue.fiware.org/enablers/application-mashup-wirecloud. On top of this application, a set of gadgets has been developed in order to look at events coming from the real world.

3.2.2 Technological stack

Item	Value
Nature	Web platform
Programming Language	HTML, Java, Javascript
Development Tools	Eclipse
Additional Libraries	N/A
Application Server	Apache 2.2, Apache Tomcat 8 on Linux OS
Databases	MySQL, Postgres 9

3.2.3 Technical Manual

Requirements

In order to get Wirecloud up and running, the following software is needed:

- Application Server (Tomcat 8 or greater)
- A Database Manager(Postgres 9 or greater)
- Python 2.6 or 2.7. Python 3 and other versions are currentyly not supported
- For others requirements see the following link: http://conwet.fi.upm.es/wirecloud/requirements

Installation

All the required software is available for Linux operating systems, an Ubuntu 64bit. It is highly advisable to have at least 2GB of free RAM.

Images of system with a pre-installed instance of Wirecloud is available in Oracle VirtualBox. The image can be downloaded here:

ftp://desk.txt.it/PSYMBIOSYS/psy_wirecloud.7z

Use the following credentials to download:

Username: coin

Password: coin_txt

Once the image has been download and the virtual machine has been started, execute these steps from the terminal in order to access wirecloud platform:

1. Create the project database in Postgres (only once), the database backup is available in following path:
    - */home/ psymbiosys /Downloads/psy_wirecloud.backup*
2. Move backend service in the folder *webapps* of Tomcat (only once)
    - **webapps** path: */opt/tomcat/webapps*
    - **backend service** path: */home/ psymbiosys /Downloads/PsymbiosysRestService.zip*
3. Change user in SuperUser
    - *sudo su*
        - password: *psymbiosys*
4. Go to wirecloud folder
    - *cd /opt/wirecloud_psy/*
5. Start wirecloud instance
    - *python manage.py runserver 0.0.0.0:8080 –insecure*

3.2.4 Licensing

Software associated to the Wirecloud product is provided as open source under Affero General Public License version 3 (AGPL v3) with a classpath-like exception (allowing widgets and operators running onto Wirecloud to be licensed under any license without any restriction). Please check the specific terms and conditions linked to this open source license at https://github.com/Wirecloud/wirecloud/blob/develop/LICENSE.txt.

3.3 User Manual

How to access to the platform

To access wirecloud platform, open a browser(Firefox, Chrome) and type http://0.0.0.0:8080 and use the username and password you provided when populating the database to sign in on the platform

Username: administrator

Password: admin

Click on “My Resources” to upload the widgets.

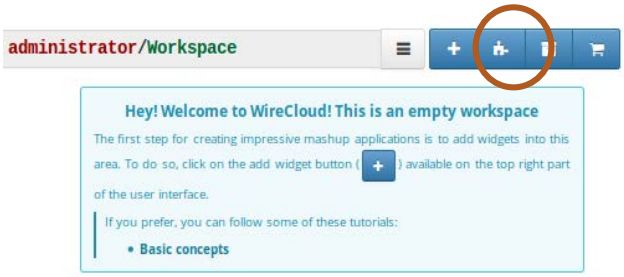

Figure 1: user Workspace

use the following files:

/home/psymbiosys/Downloads/LoomLog.wgt

/home/ psymbiosys /Downloads/LoomsItem.wgt

/home/ psymbiosys /Downloads/LoomsItemProperties.wgt

To upload widget click on “upload” button

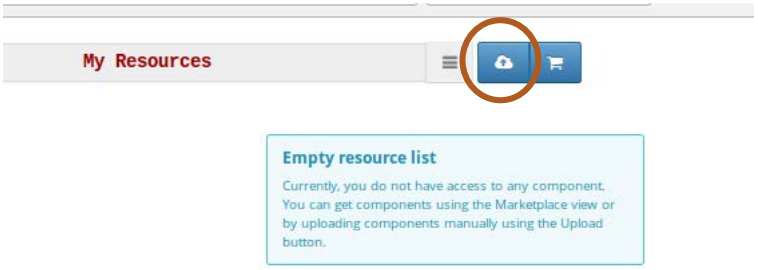

Figure 2:upload widget

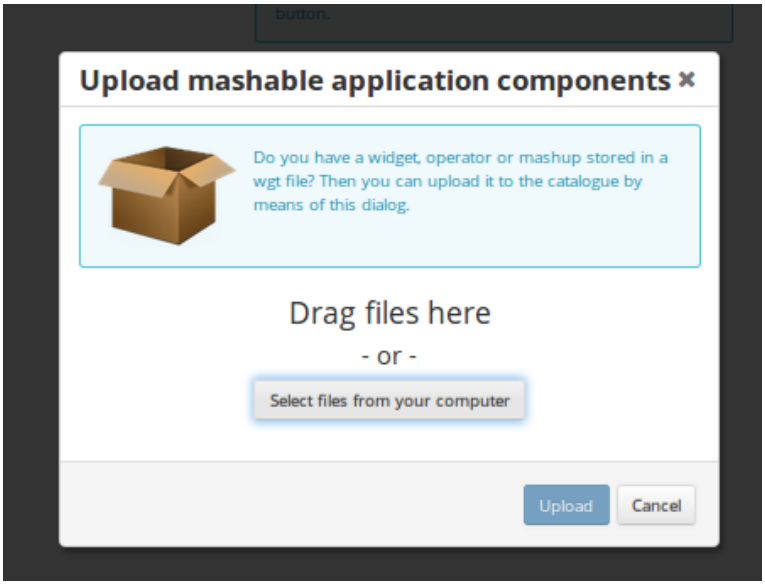

Figure 3: upload form

Once installed, you should be able to see all widgets in the “My Resouces” view:

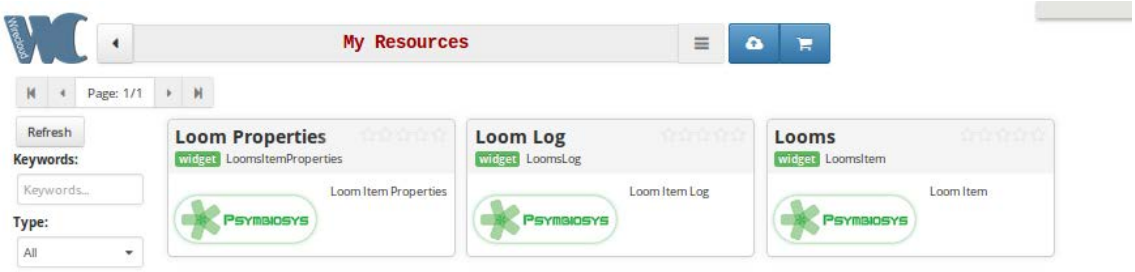

Figure 4: My Resources

Open the mashable application component details clicking on it and then click on *Publish*

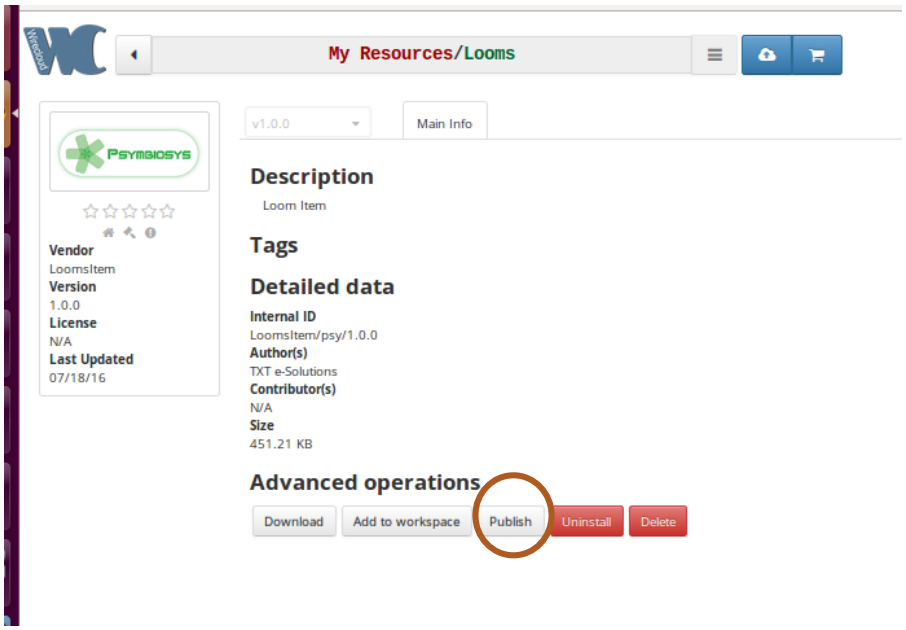

Figure 5: Widget details

Back to Workspace view and click on “Add widget” button to add on the local Workspace

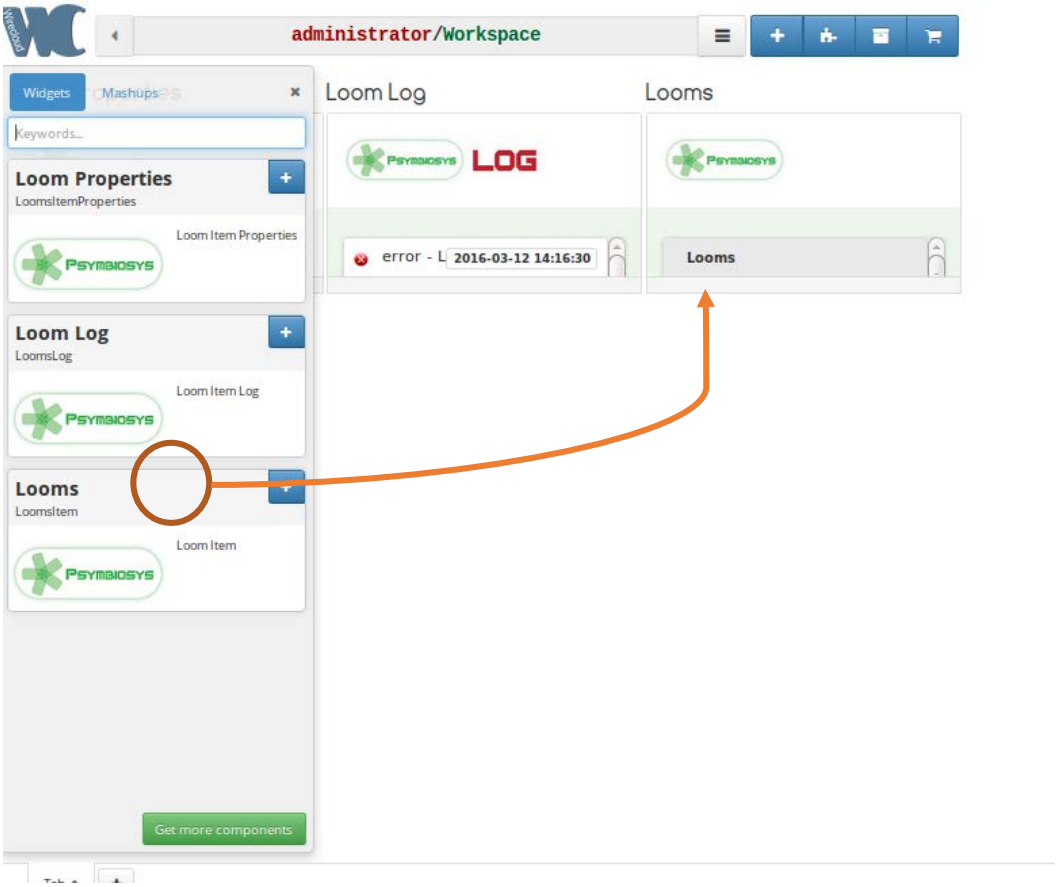

Figure 6: add widget in user workspace

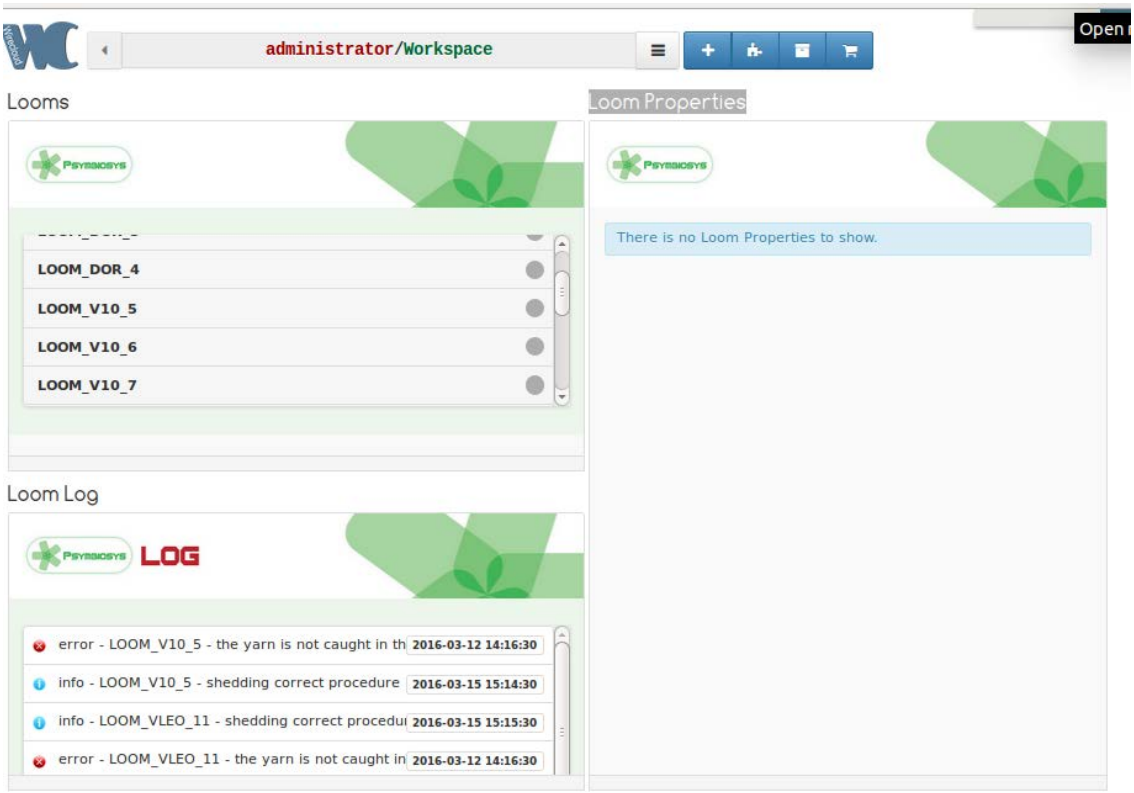

Figure 7:Workspace view with widgets - no wired

Wiring widgets

Once you have chosen the widgets, you can wire them to enable interactions between widgets. Widgets in wirecloud, are capable of sending and/or receiving events and data through well-identifield ports called endpoints. When you connect two compatible endpoints, the second one(i.e. the input or target endpoint) prepares to receive data coming from the first one(i.e. the output or source endpoint).

Drag and drop the components (operators/widgets) from the sidebar for being able to connect them

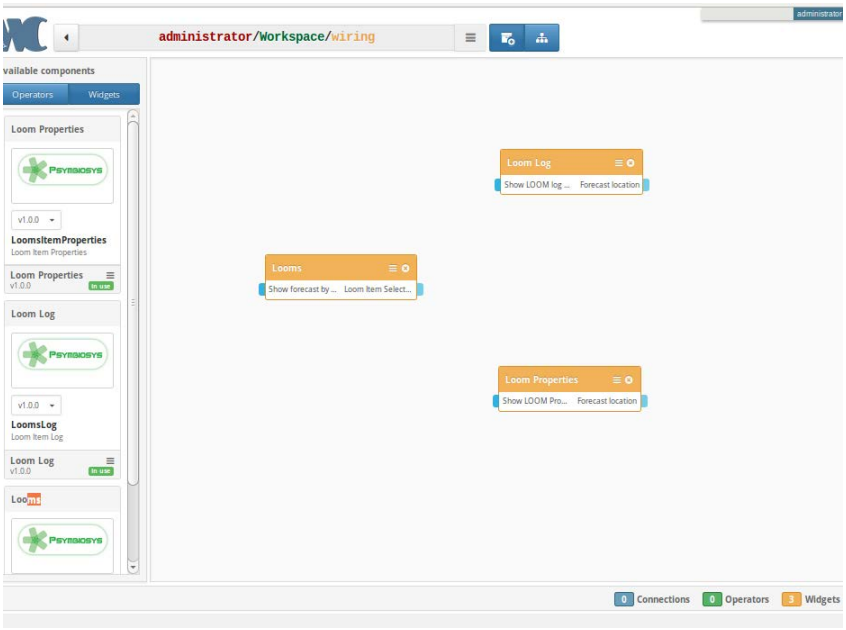

Figure 8: wiring editor

To connect two widgets, using drag&drop from output endpoint to input endpoint, you will see that endpoint get highlighted, this means that the endpoints are compatible

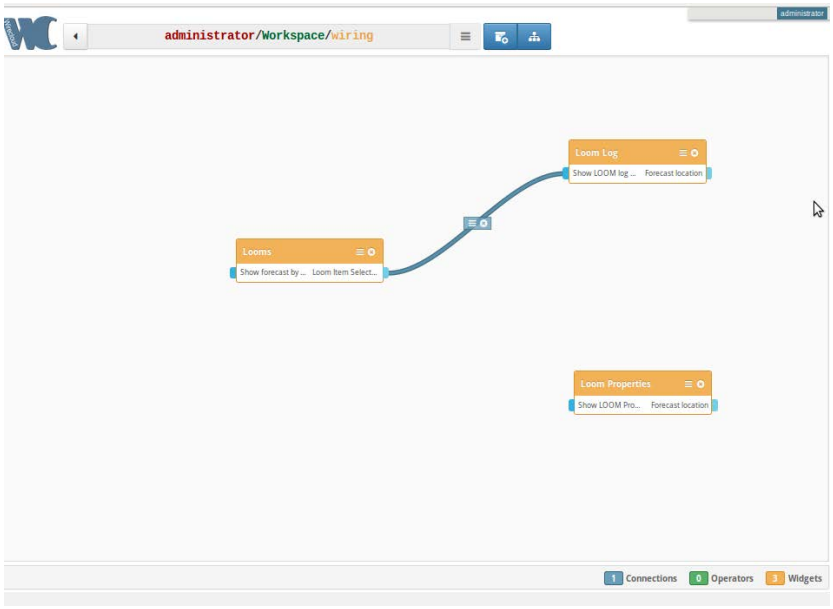

Figure 9: widgets wired

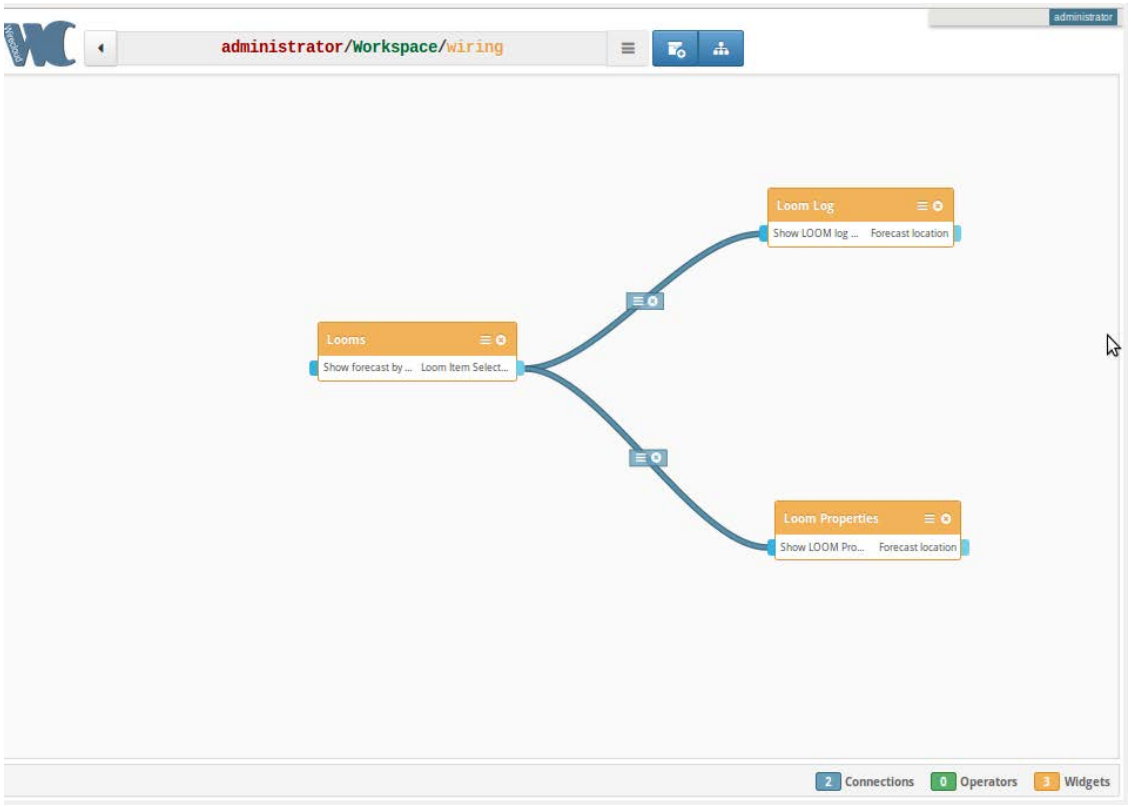

Figure 10: widgets wired

If you return to the workspace view, you will see that “loom properties” updated each time you click on loom item

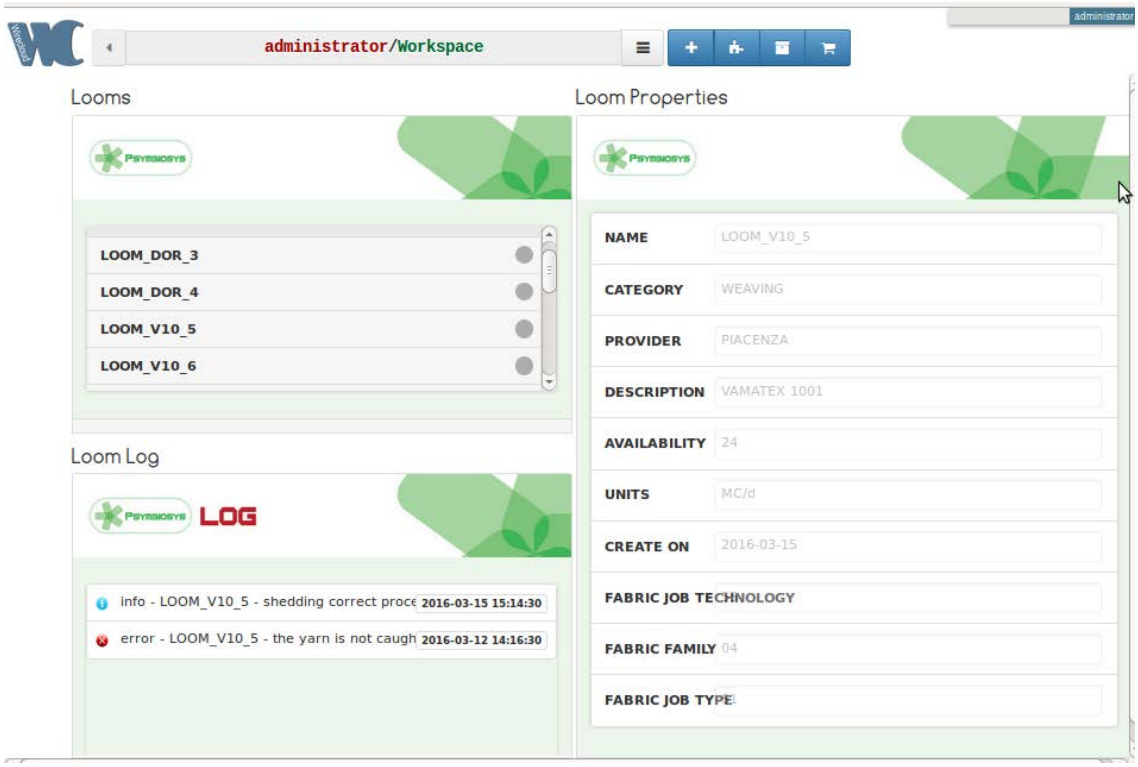

Figure 11:Workspace view with widgets - wired

3.4 Conclusions and Future plans

The implemented application complements the WP6 SOA-EDA platform injecting a component that allows human users to look at EDA events in the same interface of SOA services and to create, even without any programming skills, communication among different components. The software is presented in this document using an example freely taken from the PIACENZA use case. The software as WP6 architecture is going to be instantiated into pilots in WP9 and run in WP10; first target of the application is PIACENZA pilot as visible by the example (realistic but not real) taken from that domain.

4 Real World Manager - Privacy component

In order to complete the SOA-EDA platform is necessary to give to the user the possibility to interact will all the components also on the editing phase. The editing about the real world manager is completed in this phase by the modeller of the graphical editor for the creation of the access policy.

D6.1 details the ABE Service which is a web application able to encrypt/decrypt documents using the CP-ABE encryption technique. In summary, the CP-ABE (Ciphertext Policy Attribute Based Encryption) makes possible, on the one hand, to encrypt data using a public key plus an access policy specified by the data owner, and, on the other hand, to decrypt the encrypted information using keys that are specific of each involved subject and that are generated according to characteristics of these subjects (e.g., based on the user's profiles). The decryption process will succeed only if the subject's keys meets some of the characteristics envisaged by the data's access policy. Therefore, the data owner can specify an access policy without having to identify in advance each subject authorized to have access to the data, but only specify what are the conditions the requesting subject has to meet to gain access to the information, and, at the same time, each authorized subject has its own decryption key that is generated, for example, from the subject's profile.

As compared to the version described in D6.1, FINCONS has released a new version of the service that adds the following functionalities:

- A graphical editor for the creation of the access policy. The user can grab specific graphical elements and combine them to create an access policy (see Figure 12). The created policy can be saved assigning it a specific name;
- The security mechanisms between the user's browser and the CP-ABE service, for authentication and authorization, is now based on the new IETF standard *JSON Web Token* specification[1].

Figure 12: CP-ABE policy graphical elements

[1] IETF, "RFC7519 - JSON Web Token (JWT)", May 2015

Finally, the backend LDAP component, which in the previous version was the Apache DS server, has been replaced with the OpenLDAP server[2] that is more reliable and provides LDAP DIT backup/restore features.

The new version of the component is available on the PSY Portal under this URL: http://demos.txt.it:8096/intranet/wp6/d6-1-priv-sec-dataprot-fincons/.

[2] http://www.openldap.org/