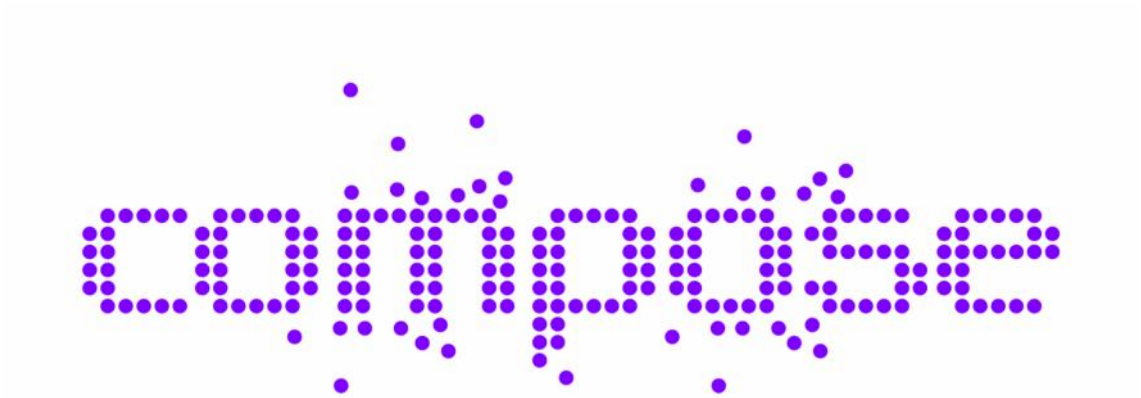

Collaborative Open Market to Place Objects at your Service

D6.2.1 Developer SDK – First Version
D6.2.2 Developer IDE – First Version
D6.3.1 Cross-platform GUI for end-user – Fist Version

Project Acronym	COMPOSE	
Project Title	Collaborative Open Market to Place Objects at your Service	
Project Number	317862	
Work Package	WP6 Open Marketplace	
Lead Beneficiary	3	
Editor	Robert Kleinfeld, Lukasz Radziwonowicz Alexander Futász	Fraunhofer FOKUS
Reviewer	Iacopo Carreras	u-hopper
Reviewer	Elena Villa Estebaranz	Abertis
Reviewer	Benjamin Mandler	IBM Research

Reviewer	Luca Panziera	The Open University
Dissemination Level	Prototype	
Contractual Delivery Date	30/04/2014	
Actual Delivery Date	30/04/2014	
Version	V0.6	

Abstract

This deliverable summarizes D6.2.1, D6.2.2 and D6.3.1 in one document and describes work that was done in the first implementation phase of WP6. Developer portal as Graphical User Interface (GUI) for COMPOSE stakeholders was produced including tutorials, walkthrough and demo videos. The first version of the component for smart object registration and management was developed and integrated in the developer portal. In addition a browser-based graphical editor for composing service objects was created based on WP2 specifications. For developers important features such as the COMPOSE API and SDK were developed in a first version. All prototypes are documented in this deliverable and underpinned with demonstration videos.

Document History

Version	Date	Comments
V0.1	24.03.14	Table of Contents
V0.2	01.04.14	Section for Developer portal
V0.3	10.04.14	Abstract for Smart Object Composer
V0.4	14.04.14	Defined features for Smart Object Manager
V0.5	24.04.14	Outline for SDKs and APIs
V0.6	29.04.14	Changes related to review comments
V1.0	30.04.14	Final

Table of Contents

List of Figures

Acronyms

Acronym	Meaning
COMPOSE	Collaborative Open Market to Place Objects at your Service
CMS	Content Management System
GUI	Graphical User Interface
IoT	Internet of Things

2 Introduction

The main challenge of WP6 is to provide COMPOSE stakeholders an attractive and easy-to-use front-end that is built with state-of-the-art web technologies.

This deliverable concludes the work that was carried out in the 2nd phase of WP6 after the requirement analysis and specification of D6.1.1. Based on the DoW we have to deliver three deliverables D6.2.1 Developer SDK, D6.2.2 Developer IDE and D6.3.1 Cross-platform GUI for end-users. This document summarizes all three activities in one deliverable. This approach is due to the fact that all three prototypes are interrelated and integrated in one dashboard.

The *dashboard* is a web-based front-end for the COMPOSE platform including components for information purposes and interactions with COMPOSE internal components. The frond-end wraps the functionality of the internal components in a user-friendly interface, so that interaction with the individual components like the data provider are effortless. WP6 developed this dashboard with a focus on three parts: *developer portal* (D6.3.1 Cross-platform GUI for end-users), *web tools* (D6.2.2 Developer IDE) and *API & SDK* (D6.2.1 Developer SDK).

The developer portal is a Content Management System (CMS) built using Wordpress 3.9 providing information for stakeholders such as tutorials, supported hardware, API and SDK documentations as well as a general overview about the capabilities and features of the COMPOSE platform. Stakeholders for the portal include developers, service and data providers and application end-users.

As web tools we summarize the account manager for user registration and API token management. The smart object manager for (i) registration and management of devices, sensors and actuators, (ii) visualizing data streams, (iii) deployment including API access and finally (iv) data stream access control via API token management.

The smart object composer as graphical Integrated Development Environment (IDE) for (i) connecting data streams of service objects[1], (ii) raw data streams from service not related to COMPOSE (e.g. social network streams), (iii) data stream manipulations, (iv) conditions statements and (v) deployment. All web tools are built on top of the Node.js platform including Node-RED[2] as browser-based flow editor.

The COMPOSE API & SDK contain client libraries for popular embedded hardware platforms (Arduino and openPicus Flyport) that ease the communication and data collection for developers. They allow to write applications directly without using the manager and composer tools just described. This way developers have more control and flexibility to integrate COMPOSE according to specific needs. In addition, a MobileSDK is currently under development for enabling the communication with COMPOSE for mobile app developers. The MobileSDK is being provided for both native environments (iOS and Android) as well as for cross-platform frameworks (Titanium Appcelerator). Swagger provides the COMPOSE API. Swagger is a specification and complete framework for describing, producing, consuming, and

[1] Service objects is the terminology we use for concrete smart objects, i.e. objects in the COMPOSE platform which represent a data source or device and can be used as a service by COMPOSE apps.

[2] Node-RED is a Web-based graphical editor useful for creating and executing flow graphs of input and output elements that are connected through transformation elements. See: http://nodered.org/

visualizing RESTful web services. Swagger was applied for the entire documentation of the COMPOSE API.

We registered the domain www.gluethings.com as the Graphical User Interface (GUI) of the described dashboard. glue.things is also an experiment for potential exploitation strategies, giving COMPOSE an exposed spot on the Web, positioning itself against other Internet of Things platforms. User feedbacks and user experience tests will flow back in the COMPOSE platform for shaping the unique selling point of COMPOSE. The following figure presents the COMPOSE ecosystem including the important dashboard components for the end-users. As mentioned before the dashboard includes the smart object manager and composer.

The marketplace part will be addressed at a later point in the project. The goal of the marketplace is it to create an ecosystem for developers to publish and share their data for other developers as well as distribute their apps to potential customers and end-users.

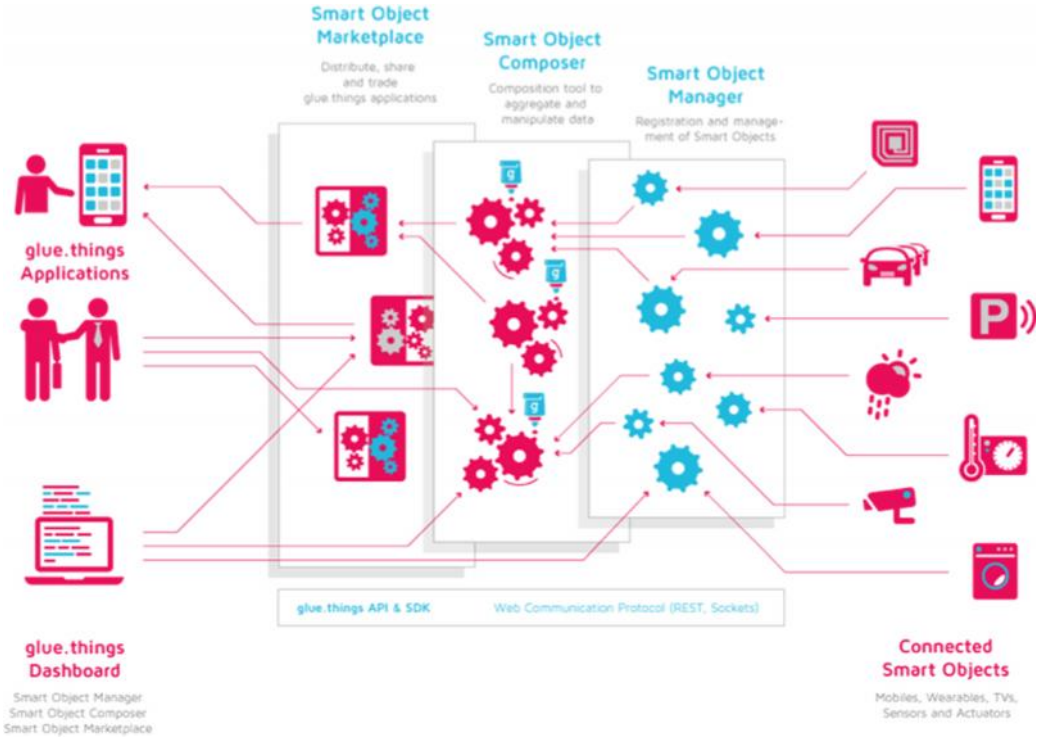

Figure 1: Schema of Dashboard Components

This deliverable contains three sections describing the components of the dashboard. We give a brief overview about the capabilities and features of each component. Furthermore we try to visualize the produced work with screenshots. glue.things will be extended step by step depending on the next release cycles of COMPOSE components. These release cycles are currently under discussion with the other WPs. With regard to upcoming hackathons and codefests we will test the dashboard in this environment.

3 Cross-platform GUI for end-users

The developer portal was developed with Wordpress 3.9 [3] and a customizable theme. It is accessible via www.gluethings.com. Features and benefits are described as well as useful information how the platform works. The following figure shows the home screen of glue.things. We tried to work out a position statement for the platform: connect, build, and distribute, 3 steps to simplify the Internet of Everything.

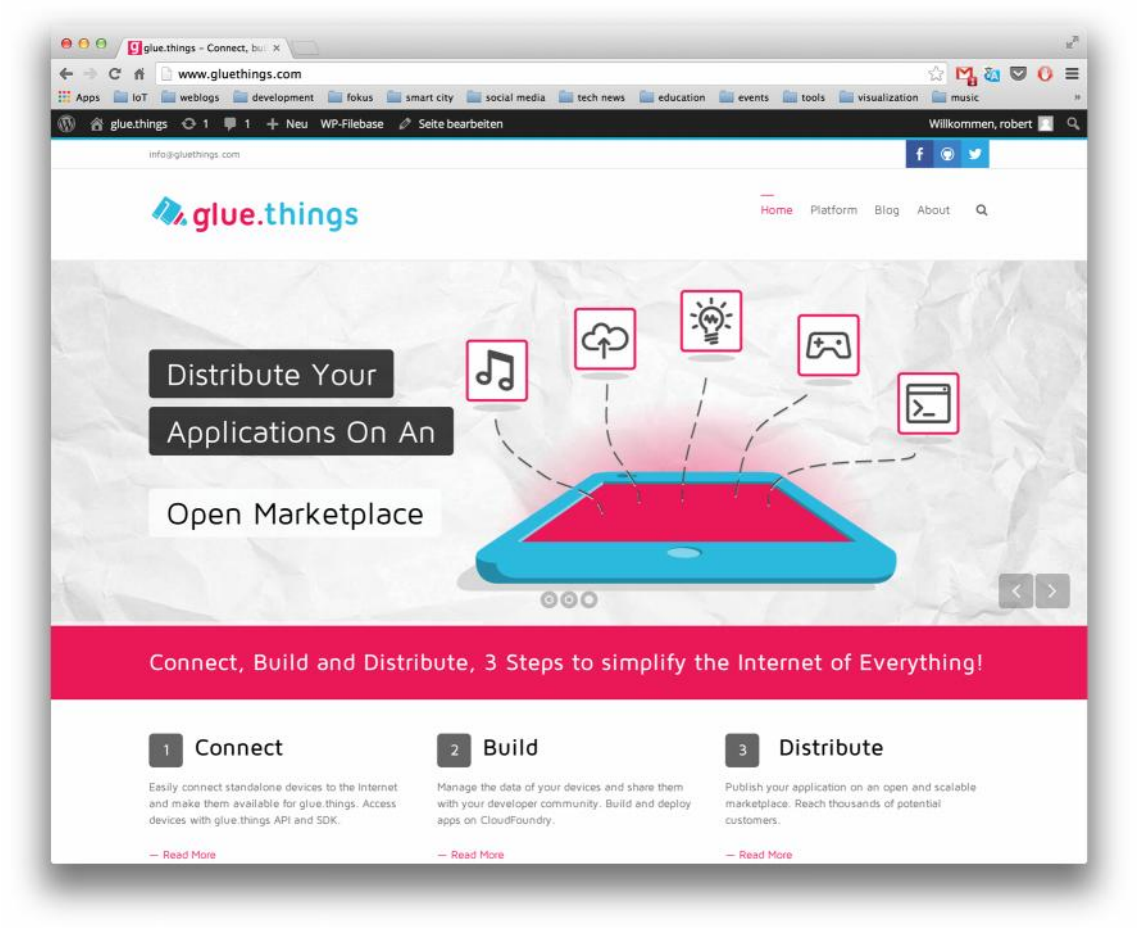

Figure 2: Home Screen and Value Proposition Statement

In addition glue.things provides a “Get Started” tutorial that explains how to use smart object manager and composer and how to quickly expose devices to the COMPOSE platform. The portal also provides extensive technical information about the architecture and concept of the platform. The developer portal further provides integration to various social media communities. Especially all code and client libraries will be published step by step on GitHub[4]. The following screenshot presents these integrations. As a project that provides the base components as Open Source we also outlined our applied and extended technologies.

[3] http://wordpress.com/

[4] https://github.com/

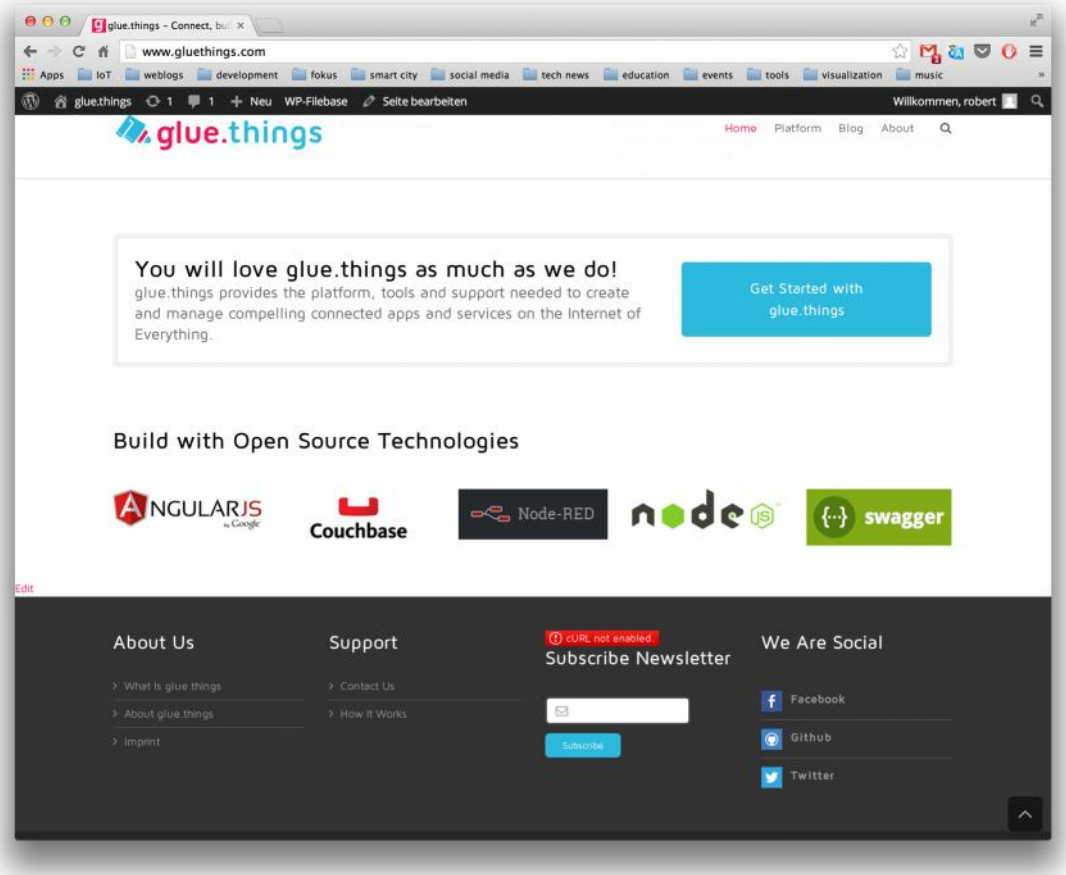

Figure 3: Get Started Tutorial and Social Media Integration

Currently this is the first prototype of the developer portal. We will include in the 2nd phase of WP6 the core components of COMPOSE. For this reason we already defined a set of user interface features that will be exposed via the developer portal. The following sections provide a brief description of these features:

Smart Object Management: Easily connect your devices such as TVs, wearables, smartphones, tablets, sensors and actuators with glue.things. Through service object virtualization, connected devices are centrally managed and accessible from anywhere on the Internet. Remotely maintain, control and interact with your devices regardless of their location. Check accessibility, change configuration or enable and disable the status of your devices. Monitor the input of your devices, set offsets, select threshold limits for data channels and modify the data visualization. Access control is supported security tokens, explained below.

User Management: glue.things provides different role models for providers and developers. Depending on the user type, glue.things supports individual views, access policies and privacy enforcements. Register your account as developer and glue.things provides you access to all management features and the glue.things API and SDK. With your developer account, easily connect your devices with glue.things and start creating innovative apps for the Internet of Everything.

Token Management: glue.things provides token management for devices and their data. API access is controlled by API tokens, which grant and control the access rights concerning is allowed to access and change. Define them using glue.things fine grained policy and visibility management.

Developer Tools: Access and manage your devices with the glue.things API and SDK. glue.things supports various software and hardware combinations needed to create apps for the Internet of Everything. The SDK provides libraries for a variety of programming languages and platforms. The glue.things API leverages the RESTful implementation of the data component from WP2 and supports the commonplace JSON and XML data formats.

Data Management: Monitor and visualize the input data of your devices in a consistent interface. Add new data channels or change their description and type. Retrieve real-time updates of your data directly in the glue.things dashboard. glue.things token management provides you the ability for defining policies and views of your data. Aggregate, manipulate, and mashup any available data stream with the Smart Object Composer. The Smart Object Composer is a powerful composition tool built on top of NODE-Red. Combine many data channels into one, then sort, filter and manipulate it with basic operations. Finally grab your composition and deploy it as an app.

Dashboard: The glue.things dashboard offers an intuitive, Web-based interface for improving the method of app development for the Internet of Everything. Connect, manage and monitor your devices with the Smart Object Manager. Use our powerful composition tool the Smart Object Composer to aggregate, manipulate and mashup data input from your connected devices. Take advantage of the Smart Object Marketplace, publish and share your data for developers and finally distribute your apps to potential customers. The glue.things dashboard supports your development work through documentations and tutorials.

Deployment: Build, run, and scale your apps for the Internet of Everything. glue.things builds on CloudFoundry capabilities exposed by the COMPOSE platform to deploy apps, where scalability is managed by CloudFoundry. Currently deployment supports apps written in Node.js, with support for other languages in the future. glue.things gives you powerful tools to build and manage you app. Deploy you app quickly, easily and in just one click directly from the glue.things dashboard. The dashboard provides an overview of all your apps, recently activity, and collaborators, giving you a cohesive interface to manage all of your apps.

Distribution: Distribute and share the input data of your devices with the developer community. glue.things provides an open and scalable marketplace not only for sharing data but also for distributing your apps. Developers have access to a complete and integrated process from app development, deployment to distribution.

As cross-platform GUI we decided to choose a responsive web design of the developer portal. With this approach the entire GUI will be accessible via all kinds of mobile devices such as smartphones and tablets. The following figure shows the developer portal on desktop and mobile devices.

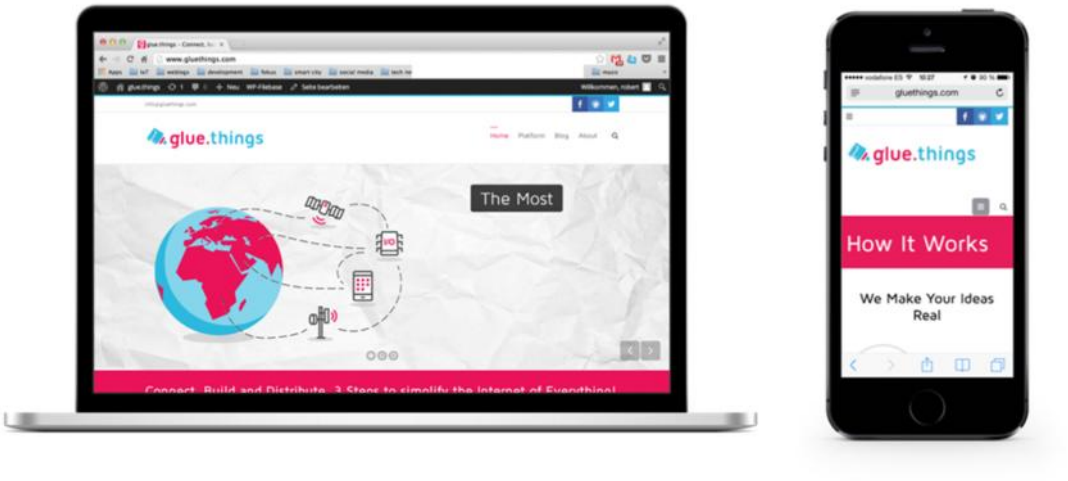

Figure 4: Responsive Web-design for Mobile and Desktop

As Wordpress is a very flexible and customizable CMS we are able to easily include the COMPOSE components in the developer portal. In the next step we will provide more tutorials, videos and walkthroughs. Currently we integrated a blog for developers about the latest news and updates around the Internet of Things world and COMPOSE findings. The blog will be extended through a community portal and a Frequently Asked Questions (FAQ) section. With this approach we hope to reach a broad mass of developers and users. With the submission deadline of this deliverable we will official launch the developer portal at www.gluethings.com.

4 Developer IDE

With the submission of this deliverable we launch the first initial components of the web tools: smart object manager and smart object composer. Both are integrated in the developer portal and accessible via a restricted area for demonstrations. During May 2014 the smart object manager will be officially released. Therefore developers can register their devices on the platform and have access to the provided API & SDK. The following sections will explain the capabilities of these web tools in detail.

4.1 Smart Object Manager

The smart object manager is responsible for the registration of devices. The developer comes to the platform creates a user account and gets an API token for development. The smart object manager provides three easy steps for connecting devices with the platform. WP6 used the servIoTicy[5] implementation of WP2 for initiation and registration of service objects. As shown in the following figure developers can easily add devices to the platform via choosing device templates.

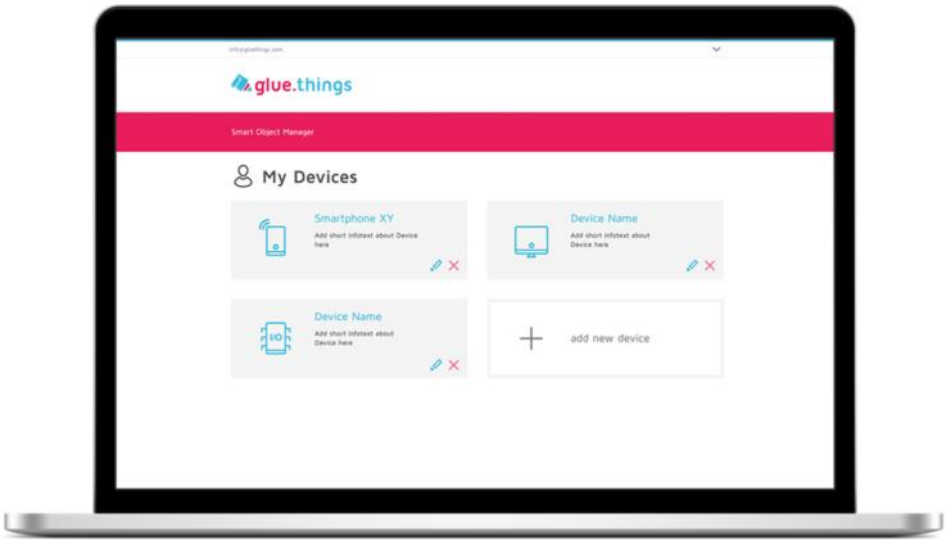

Figure 5: Smart Object Manager

For the first developer tutorial we included templates for connecting mobile devices to the platform and pushing location data to the platform. In the first step the developer creates a new device in the smart object manager. In the second step he connects his device to the platform. The final step includes configuration and management steps for the desire data streams from the device that should be exposed. The following figure shows a successful connected smartphone that pushes location data to the platform.

[5] http://www.serviotic y.com/

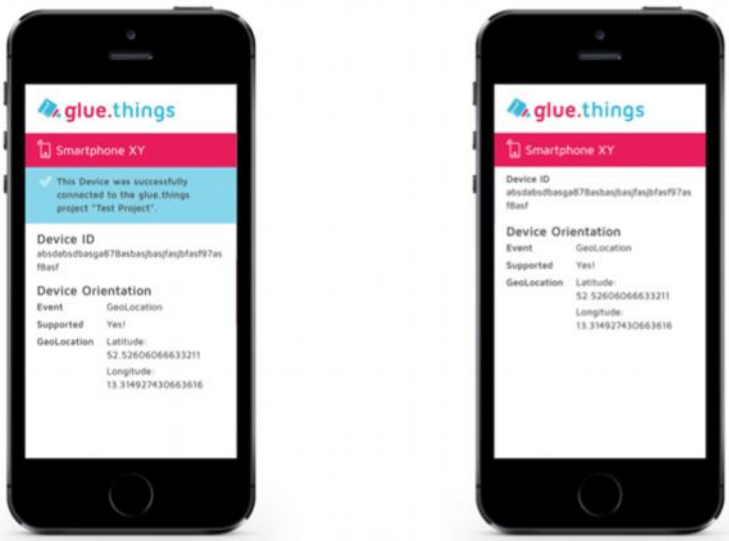

Figure 6: Connected Smartphone

The smart object manager provides also a data management dashboard for manipulating the data streams. The following figure shows the data management dashboard.

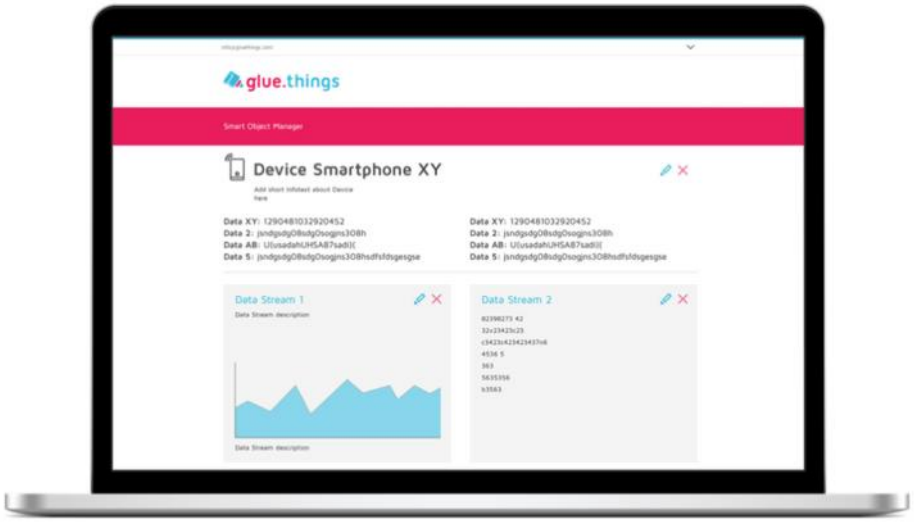

Figure 7: Dashboard for Data Management

After successful testing and bug-fixing we will officially launch the smart object manager in the middle of May, supporting various device data templates that come preconfigured for a number of desired data streams (e.g. a template for GPS location data). Currently the smart object manager is accessible via a restricted area on the developer portal. For the first initial step we included templates for device location and orientation. Based on the current developments of the pilot partners we include these examples also in the smart object manager.

For demonstration purposes a walkthrough video of the smart object manager is available as part of this deliverable.

The source code is available at the following URL: https://github.com/compose-eu/COMPOSE-UI

4.2 Smart Object Composer

The smart object composer was built on top of the browser-based flow editor Node-RED[6]. The composer is a graphical tool, to easily create workflows, that integrate the various services and service objects delivered by the COMPOSE platform.

The smart object composer combines textual and graphical models as appropriate. An example of the former is a syntax coloured scripting language editor. An example of the latter is Yahoo Pipes[7]. While this might be a simple wizard like process to construct causality chains, like IFTTT[8] others, the smart object composer provides a more complex tool, to create composite services. It is best compare to the Yahoo Pipes editor, though providing more standardized building blocks and staying more extensible. The smart object composer provides building blocks that enable the creation of workflows that rely on clearly defined control flow constructs. The smart object composer prohibits the linking of building blocks with incompatible inputs and outputs and thus already provides a basic validation capability. The following figure shows the Node-RED inspired COMPOSE smart objects composition editor with various service objects and composite service objects:

[6] http://nodered.org/

[7] http://pipes.yahoo.com/pipes/

[8] IFTTT https://ifttt.com/ is a service that enables customers to connect channels (e.g., Facebook, Evernote, Weather, Dropbox, etc.) with personally created or publicly shared profiles known as "recipes".

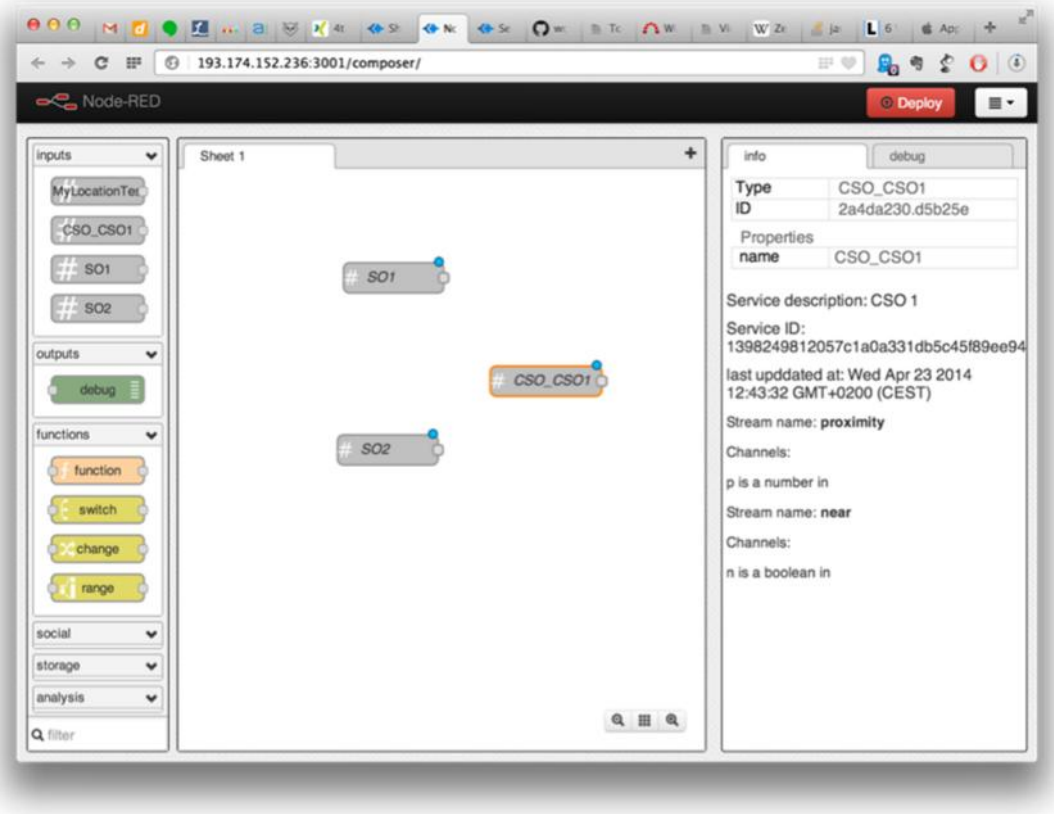

Figure 8: Smart Object Composer based on Node-RED

Composite services created with the smart object composer can be enriched with metadata that can be used by objects built on top. All required metadata will be pre-set with smart defaults, where possible, to further ease the development process. Metadata includes description, keywords and access policies for the created composite service object. Start- and endpoints of a composite service object can either be external Web services or simple HTML forms and widgets that will be created by simple wizards, which will additionally be provided as building blocks. In our case we implemented for the first iteration the automatically instantiation of service object. That means, once registered devices in the smart object manager will also appear in the smart object composer. If validation passes, all necessary files for the created composite service object will be packaged and posted to the COMPOSE platform, where it will be registered in the according service object registry. Front-end files, that consist of simple HTML5 stack files, will be transferred to a hosting space within the COMPOSE platform, that is dedicated to the creating developer. Therefore a COMPOSE internal application is essentially another form of a composite service but enhanced as simple front-end widget.

For demonstration purposes a walkthrough video of the smart object composer is available as part of this deliverable.

The source code is available at the following URL: https://github.com/compose-eu/COMPOSE-UI

5 Developer SDK

We extended the COMPOSE Service Object API provided by WP2 for integrating in the developer portal. By using Swagger the COMPOSE API can be released in various programming languages. For the first prototype we will released the COMPOSE SDK as JavaScript and Java library that wraps the COMPOSE API. See the WP2 deliverable for an in-depth explanation of this API.

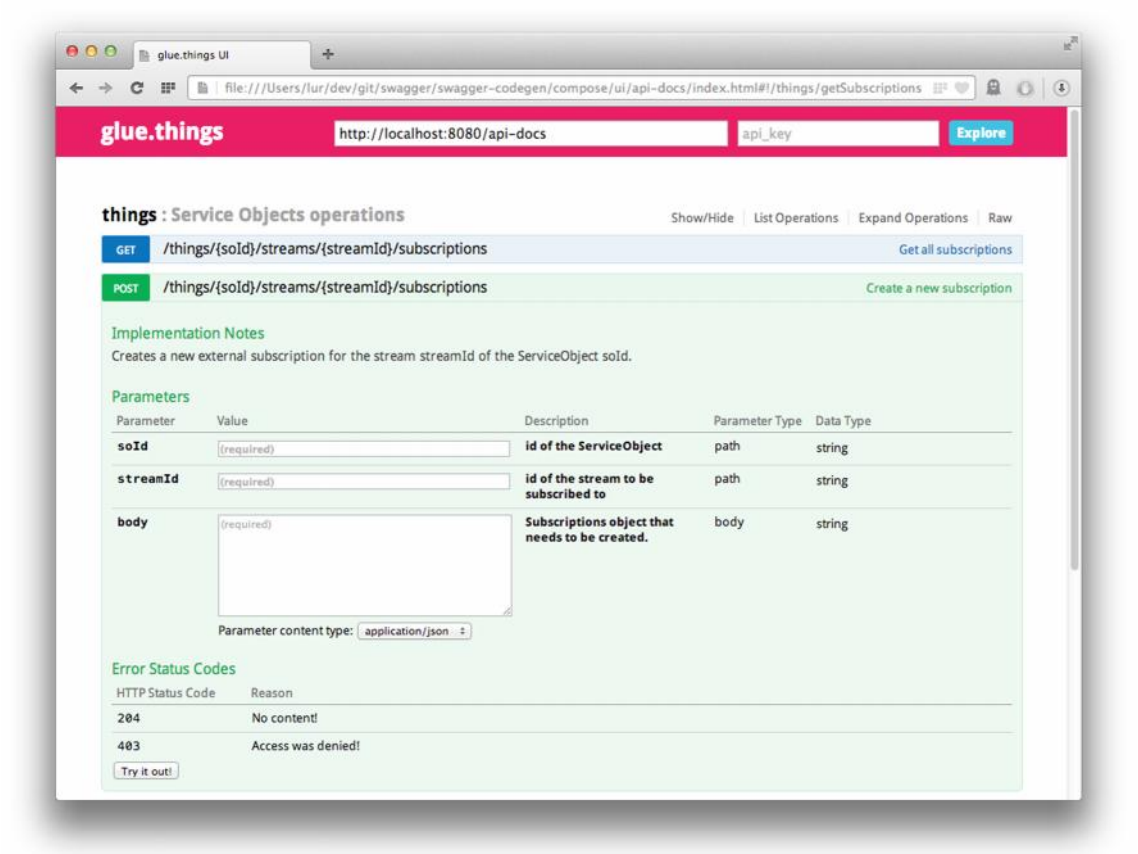

Figure 9: Developer SDK and API test page

The COMPOSE SDK can be accessed via the following URL: https://github.com/compose-eu/COMPOSE-SDK

Client libraries for popular embedded hardware platforms (Arduino and openPicus Flyport) have been developed that ease the communication and data collection on COMPOSE. In addition, a MobileSDK is currently under development for enabling the communication with COMPOSE for mobile app developers. The MobileSDK is being provided for both native environments (iOS and Android) as well as for cross-platform frameworks (Titanium Appcelerator). The MobileSDK can be accessed via the following URL: https://github.com/compose-eu/MobileSDK

6 Future Directions

This paragraph outlines the next steps for the components in Work Package 6. The immediate next step is the further integration with the work of the other work packages.

6.1 Integration with Work Package 5 (Security)

Currently Token Management is provided by WP2, but work is already being done to integrate the token access control work from WP5. This will result in a few changes to the API provided by WP2 that deal with authorization and how requests need to be formatted. We expect to adapt the Smart Object Manager and Compose components to have a fully integrated platform.

6.2 Integration with Work Package 3.2 (Deployment)

Deployment is responsible for taking the flow graphs created in the Smart Object Composer in order to run them in the cloud infrastructure that is being developed in Work Package 4. The next step here is to integrate with the deployment component and to provide them with access to flow graphs in the composer.

6.3 Next iteration for COMPOSE SDK

So far we have provided language support for JavaScript and Java. We plan to evaluate and tweak the provided client libraries for these languages based on feedback of Work Package 7 (pilots). We will also evaluate what demand there is for providing client libraries in further languages, e.g. Ruby or Python.

7 Summary

This deliverable presented the first initial prototypes of the entire developer front-end. We produced a common GUI for developers, providers and end-users. This will be used as further access point for all exploitable COMPOSE components. The produced prototypes can be accessed via the following URLs:

- Developer portal: www.gluethings.com
- Smart Object Manager: https://github.com/compose-eu/COMPOSE-UI/tree/master/manager
- Smart Object Composer: https://github.com/compose-eu/COMPOSE-UI/tree/master/composer
- COMPOSE SDK with API: https://github.com/compose-eu/COMPOSE-SDK