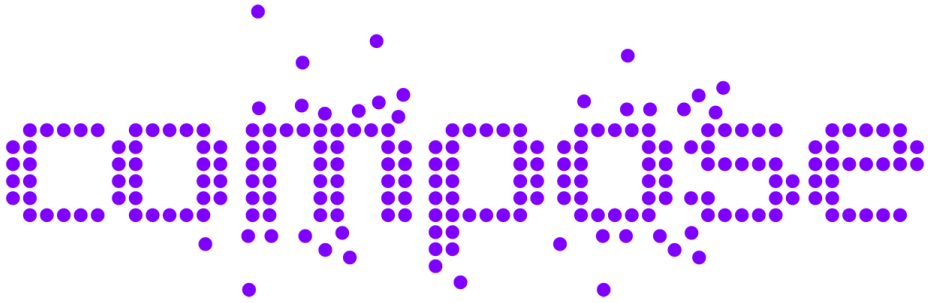

Collaborative Open Market to Place Objects at your Service

D32.1.2

Prototype of the execution environment for service execution

Project Acronym	COMPOSE	
Project Title	Collaborative Open Market to Place Objects at your Service	
Project Number	317862	
Work Package	WP32 Services Deployment	
Lead Beneficiary	Retevision	
Editor	Rafael Giménez	BDigital
Editor	Elena Villa	Retevision
Reviewer	Yoav Tock	IBM
Reviewer	Charalampos Doukas	CREATE-NET
Dissemination Level	PU	
Contractual Delivery Date	30/04/2014	
Actual Delivery Date		
Version	V1.0	

Abstract

The present prototype represents the first version of the COMPOSE services execution environment. This is a central piece of the COMPOSE architecture, since all COMPOSE entities will pass through this component to be finally executed. The presented COMPOSE life cycle management capabilities enable developers to create their own COMPOSE applications and manage the lifecycle of those applications autonomously. This report describes the components comprising the solution and the set of interfaces provided by the prototype.

This document aims to accompany the initial prototype of the COMPOSE execution environment for service execution platform rather than provide a detailed design document of the demonstrated system.

Document History

Version	Date	Comments
V0.1	10/04/2014	Initial version
V0.2	29/04/2014	First version ready for review
V0.3	29/04/2014	Incorporated comments from IBM review
V0.4	29/04/2014	Updated after internal RETE review

Table of Contents

List of Figures

List of Tables

Acronyms

Table 1: Acronyms table

Acronym	Meaning
COMPOSE	Collaborative Open Market to Place Objects at your Service
API	Application Programming Interface
REST	Representational State Transfer

1 Introduction

The COMPOSE execution environment supports all the different COMPOSE entities, such as COMPOSE services and applications. As is shown in the Figure 1, the COMPOSE life cycle management component enables developers to create their own COMPOSE services and manage the lifecycle of those services autonomously.

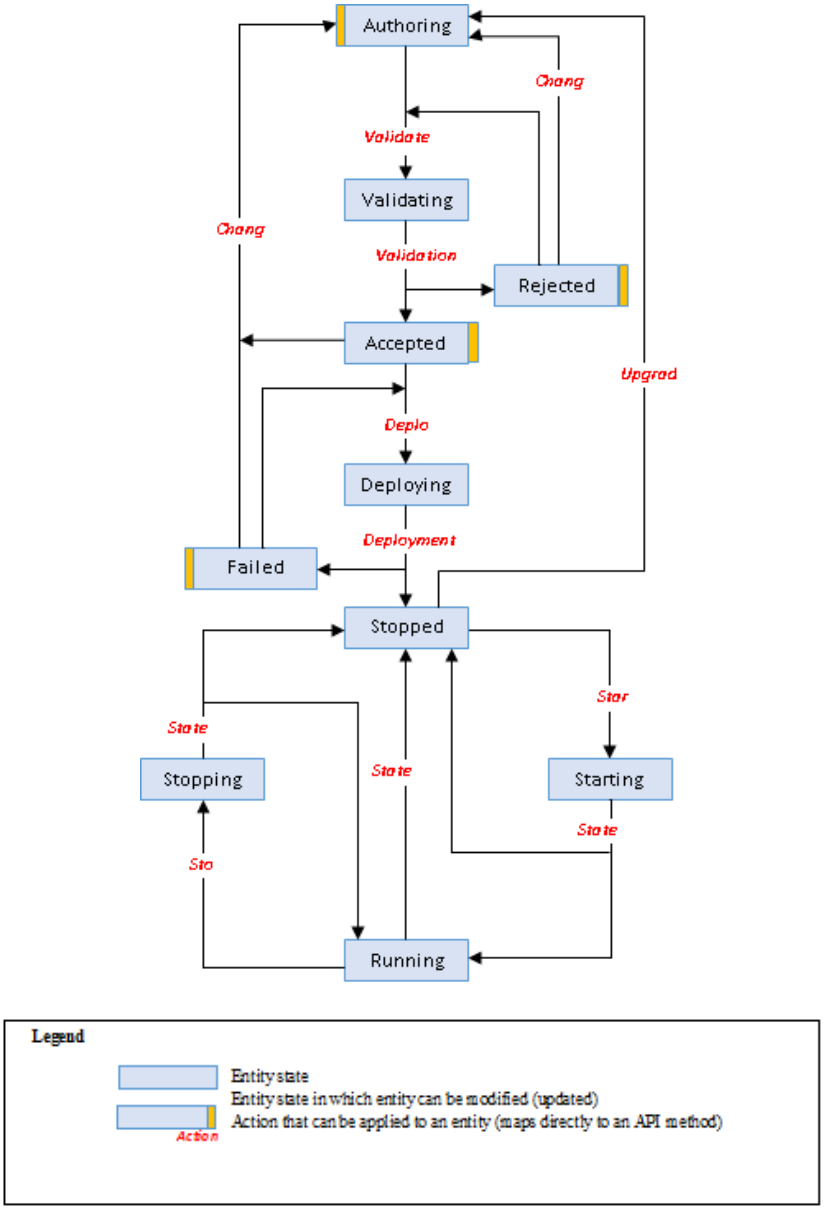

Figure 1: Life cycle

Figure 2 shows the final architecture proposal for the COMPOSE Service execution component. The top and bottom would be other components of the COMPOSE platform, such as the security component or runtime component (Cloud Foundry).

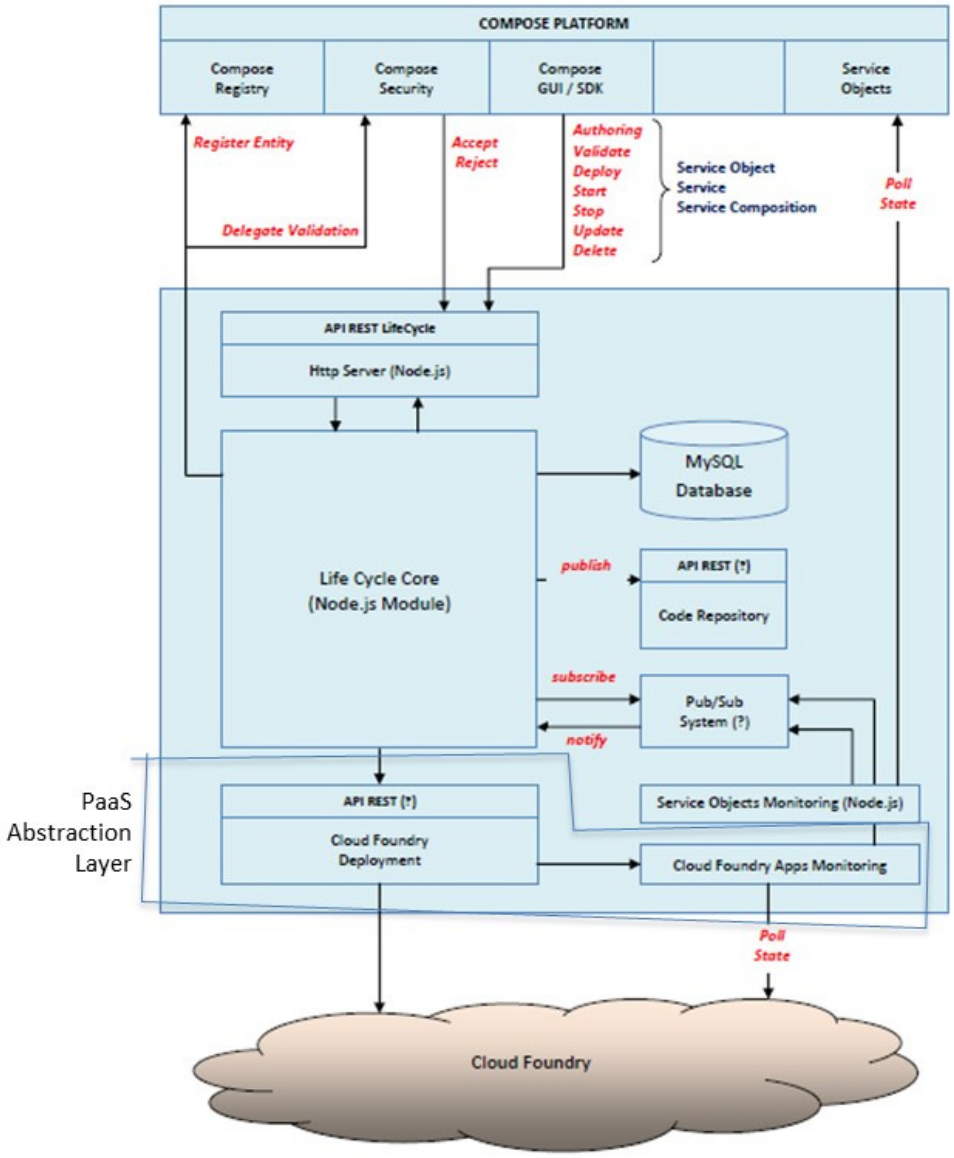

Figure 2: Final architecture of service execution

2 Prototype overview - short description

The first prototype of the service execution environment aims to support the life cycle of a COMPOSE Service, ranging from Authoring to Deployment states. The current implementation

provides a proof of concept for two different scenarios: one based on a very simple COMPOSE service which simply returns “Hello World” messages to external REST API calls, and one based on a COMPOSE Service Object. We distinguish those types because of the deployment logic, but that distinction will be hidden from the developer perspective. In fact, developers will create COMPOSE Services and the service execution component will have the intelligence to know type of service is being deployed (COMPOSE Service Object, COMPOSE Service or COMPOSE Composed Service).

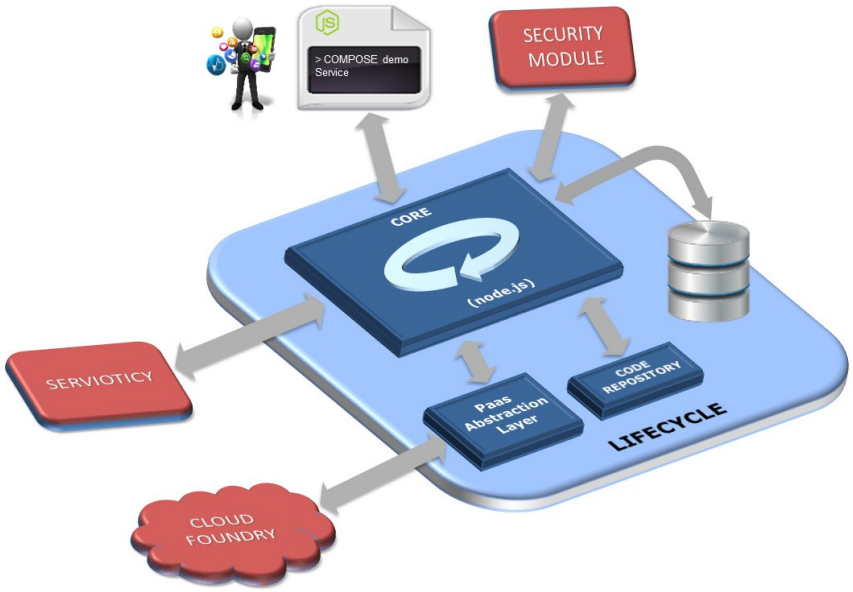

Figure 3: Prototype of the execution environment for service execution

2.1 Architectural responsibilities and interactions

The COMPOSE life cycle management component is an essential part of the COMPOSE platform. It controls the validation, deployment and registration of COMPOSE entities providing an API to upper level controllers responsible for supporting the core functional flows.

It relies on the COMPOSE security module in order to authenticate and authorize users and in order to validate services that developers create. It implements the functionality that actually deploys those services, sending them to the right place to be executed and it takes care of their registration in the COMPOSE registry.

2.2 High level design

2.2.1 Main functionalities introduced

The main functionalities that will be introduced by this prototype are:

- Registering (inserting) a COMPOSE entity into the life cycle workflow.
- Validating a COMPOSE entity.
- Deploying a COMPOSE entity.

These are the essential steps that must be applied on COMPOSE entities.

2.2.2 Main Design choices

The implementation of the life cycle component relies on the following components:

- **A REST API controller supporting the workflow operations.**

  A Node.js application. This module implements the end points of the REST API, used to move the COMPOSE entities through the workflow.

- **A MySQL database to store services and their related data.**

  This is a simple relational database to store service related data

- **A code repository to store the files associated with services.**

  This is the place where the actual code (files associated with services) is stored. The current implementation provides a temporary solution, which stores files directly in the file system.

- **A Monitoring system to take care of the state of deployed services (running, stopped, crashed ...).**

  This module will be responsible for monitoring COMPOSE services (running as Cloud Foundry applications), Service Objects (running in Servioticy) and Service compositions (running as Cloud Foundry applications via node-red runtime and node.js). It will be also a node.js module.

- **A pub/sub system to send/get notifications about state changes of deployed services.**

  All notifications of state changes of any deployed COMPOSE service will be sent to this pub/sub system.

3 What is being demonstrated

The purpose of this prototype is to show the steps that must be followed in order to get a COMPOSE Service up and running and ready to be used by external users.

The prototype is composed by two different scenarios:

- The first scenario shows how to create, deploy and validate a Service Object to be deployed on the COMPOSE platform.
- The second scenario shows how to create, deploy and validate a COMPOSE Service in the COMPOSE platform.

The COMPOSE Service Object will be a temperature sensor from the SmartBrain Platform and the COMPOSE Service will be a simple service created to test the proper operation of the workflow and verify that it is possible to perform the actions defined in D32.1.1, as start or stop a service on the COMPOSE platform, and the underlying Cloud Foundry infrastructure.

Finally, a video is being created in order to demonstrate the prototype, where a COMPOSE Service object is a file with the manifest description and on the other hand, a COMPOSE Service is a set of files in Node.js. Within the workflow it is shown the manner in which these two services go through different states and how they can perform different actions, such as stopping the COMPOSE Service.

4 External API - for use by COMPOSE components

Other components of the COMPOSE platform must use the API exposed by this component in order to get COMPOSE services validated, deployed and registered. All information related API is available in http://compose-lcm.ng.bluemix.net/.

5 Future directions

- At the present, the prototype allows the execution of simple services or service objects. In the coming months, it is planned the integration of the Service Composition deployment.
- WP32 contributors are exploring different code repository solutions to choose the one that best suits the needs of COMPOSE platform and the next step is the creation of the COMPOSE code repository.

- At the moment, service execution is using an internal registry and the component will be integrated with the COMPOSE registry.