(FP7 614100)

D7.3.1 Initial Design and Implementation of the Configuration and Composition Manager

2014-08-01 – Version 1.0

Published by the IMPReSS Consortium

Dissemination Level: Public

Project co-funded by the European Commission within the 7th Framework Programme and the Conselho Nacional de Desenvolvimento Científico e Tecnológico
Objective ICT-2013.10.2 EU-Brazil research and development Cooperation

Document control page

Document file: D7.3.1 Initial Design and Implementation of the Configuration and Composition Manager_v1.0.docx
Document version: 1.0
Document owner: Enrico Ferrera (ISMB)

Work package: WP7 – IDE Framework for Model-driven development
Task: T7.3 Unified configuration management API
Deliverable type: P

Document status: ☒ approved by the document owner for internal review
☒ approved for submission to the EC

Document history:

Version	Author(s)	Date	Summary of changes made
0.1	Enrico Ferrera	2014-07-20	Initial structure of the document. Contributions in all sections.
0.2	Davide Conzon	2014-07-22	
0.3	Enrico Ferrera	2014-07-25	Document review and minor modifications
0.4	Davide Conzon	2014-07-27	Document review and minor modifications
1.0	Enrico Ferrera	2014-08-01	Deliverable ready to be submitted

Internal review history:

Reviewed by	Date	Summary of comments
Ferry Pramudianto	2014-07-31	Accepted with minor comments

Index:

1. Executive summary

This document describes the role and architecture of the Configuration and Composition Framework (CCF). Moreover, APIs for interaction with CCF are described.

The deliverable is organized as follow: in Section 2 the need and concept of provisioning is introduced and defined. Section 3 describes the CCF architecture, specifying which stakeholder this framework is intended to apply to. In Section 4, API operations are described. Finally, in section 5, are shown some example of usage of the CCF.

Deliverable D7.3.1 actually consists in the first prototypical implementation of the CCF, for this reason the document has to be considered a compendium of the software and consequently the number of pages have been kept restricted.

2. Introduction

During the last years, much research has been carried out around the concept of Internet of Things (IoT). The holistic interaction among entities such as objects, systems, services and people, as prescribed by the IoT paradigm, provides the infrastructure for development of complex platforms enabling the pursuit of smarter environments and society. A plethora of research projects focus on new and advanced platforms providing more and more smart features for many different purposes, e.g. sustainable buildings more advanced power grids, more efficient logistics, e-health, etc.

IMPReSS project aims at realize a systems development platform (SDP) which enables rapid and cost effective development of systems involving the Internet of Things and Services and at the same time facilitates the interplay with users and external systems. The IMPRESS development platform is going to be useful and usable for the realization of any kind of application that intend to embrace a smarter society. Such kind of platform has to provide components, which are designed to be more general-purpose possible. In spite of the generic finality, existing IoT platforms are often designed to reach specific objectives and they lack of an easy and customizable way for the provisioning of the whole platform (the process of preparing and equipping a system to allow it to provide, new, services to its users).

The provisioning process consists in configuration and composition of the different components of the platform, in order to initialize them and establish how they have to work together, to set the correct workflow of the platform. This is an important part to improve the ease of use of IoT platforms, but often the developers not address it, during the design and implementation of these solutions.

The following parts of the document provide an overview of the current design and implementation of the Configuration and Composition Framework. Further additions and refinements or modifications to this component are expected in subsequent phases of the project.

3. Configuration and Composition Framework Architecture

3.1 Context

Figure 1, depicted in Deliverable D2.2.1 SDP Initial Architecture Report, shows the IMPReSS platform architecture from a functional point of view.

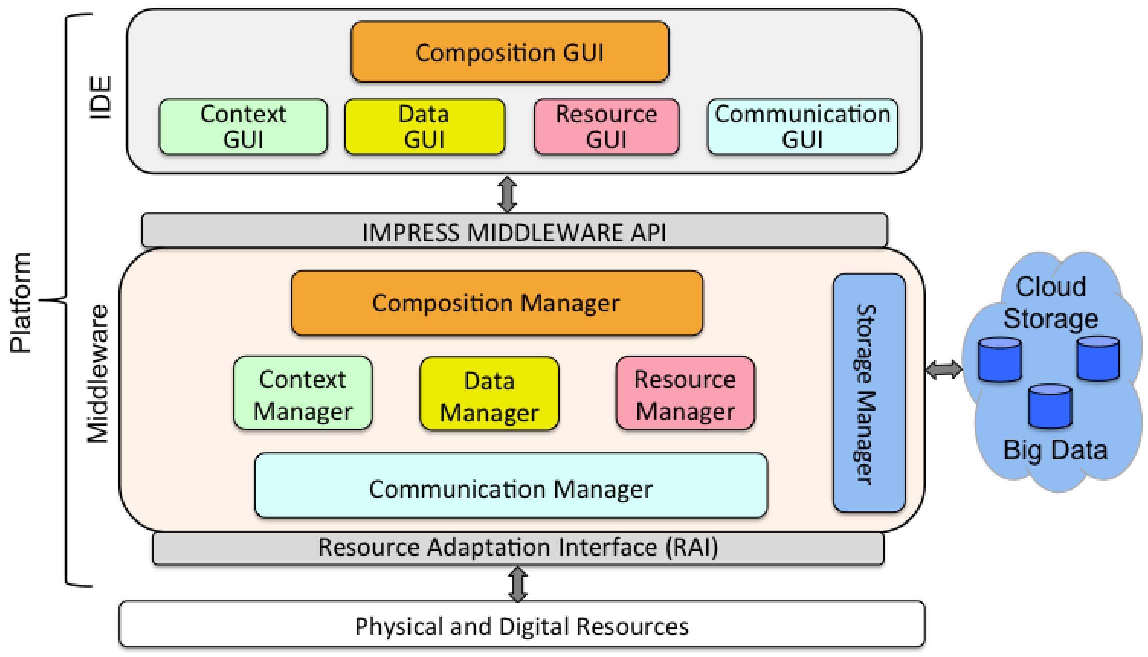

Figure 1: Functional architecture

The orange modules are responsible for the provisioning of the platform. They have two main goals: the first one is to allow the composition of the different modules in order to realize applications, using them; the second one is to allow configuring the entire platform and the single modules. The Composition GUI relies on the Composition Manager, which is responsible to perform the provisioning tasks requested by the stakeholder, through the composition GUI.
D2.2.1 SDP Initial Architecture Report introduces the different stakeholders of the IMPReSS platform, which for recap, are reported here in the following:

- **Partner**: The IMPReSS Partner who contributes to the development of the IMPReSS System Development Platform (SDP). Partners considered here are the European ones - FIT, CNET, IN-JET, ISMB, VTT – and the Brazilian ones - UFPE, UFAM, TAO, CHESF, ENG, UFABC. IMPReSS Partners have a natural broader view of the internal components of the architecture, because they need to put them to work together by orchestrating components and dataflows.
- **Developer:** The Application Developer who uses the IMPReSS SDP to develop IMPReSS-enabled Applications. Target applications are energy efficiency systems addressing the reduction of energy usage and CO2 footprint, within the context of the Internet of Things (IoT).
- **Integrator**: The Solution Integrator who installs, configures, deploys application, and connects them to other external services and hardware components. Different people or organizations may play the role of integrators. Integrators must have special interfaces (GUIs actually, in different flavors, such as Web-based and smartphone/tablet apps) with the

system, so that they are easily able to configure the system to operate under different circumstances in different environments.

- **Recipient**: The Final Recipient, who is affected by the solution, such as university professors, students and staff, employees of a company (with different skills and positions), audience of a theater, or even house home owners. These people can interact with the solution by means of different interfaces (web-based, apps) for configuring certain parameters and receiving real time information.

The stakeholder mostly involved with the provisioning issues are the Developer, who combines different modules and composes the specific logic flow, in order to realize the final application, and the Integrator, who sets the parameters of the platform modules to make the system effective. To fit these two different needs of the Developer and Integrator, the CCF allows dealing with the two aspects of provisioning:

- **Composition**: i.e. interconnect different available components of the platform (e.g. service proxies, data filtering and aggregation modules, decision support systems, etc.). In other word, the composition aims to realize the application, defining, for each relevant platform component available, from which other components it has to take the inputs and to give its outputs. This stage defines the workflow of the application. This feature is used by the platform Installers for defining connections among the different entities in order to implement specific application logic. In fact, through the composition is possible to realize the actual application to be executed.
- **Configuration**: this stage provides to each platform component involved in the realization of the application (i.e. the ones interconnected through the composition stage) the values for the correct behaviour of the applications. For instance, suppose we have interconnected, through the composition stage, the output of a temperature sensor to a module that raises an alert whenever the temperature exceeds a threshold. In this case, the configuration stage is responsible for set parameters such as the sensing rate of the sensor temperature and the threshold temperature at which the second module has to rise the alert. The CCF shows to the platform Manager all the available services and entities, allowing to configure the parameters of the entities of the overall platform.

These two different aspects of the provisioning issue has been considered for the design of the CCF in order to satisfy the functional requirements from the two stakeholders.

3.2 Architecture description

The role of the CCF is to provide a unique and general way of preforming the provisioning of the platform.

The architecture of the CCF is shown in Figure 2. This architecture is inspired by the SNMP one (SNMP , 2014) and aims at performing the configuration and composition of hardware and software resources, for the platform provisioning. The architecture of CCF consists of two levels, global and local levels, and is mainly composed by two components:

- A Configuration and Composition Manager (CCM) at a global level.
- A Configuration and Composition Agent (CCA) at local level.

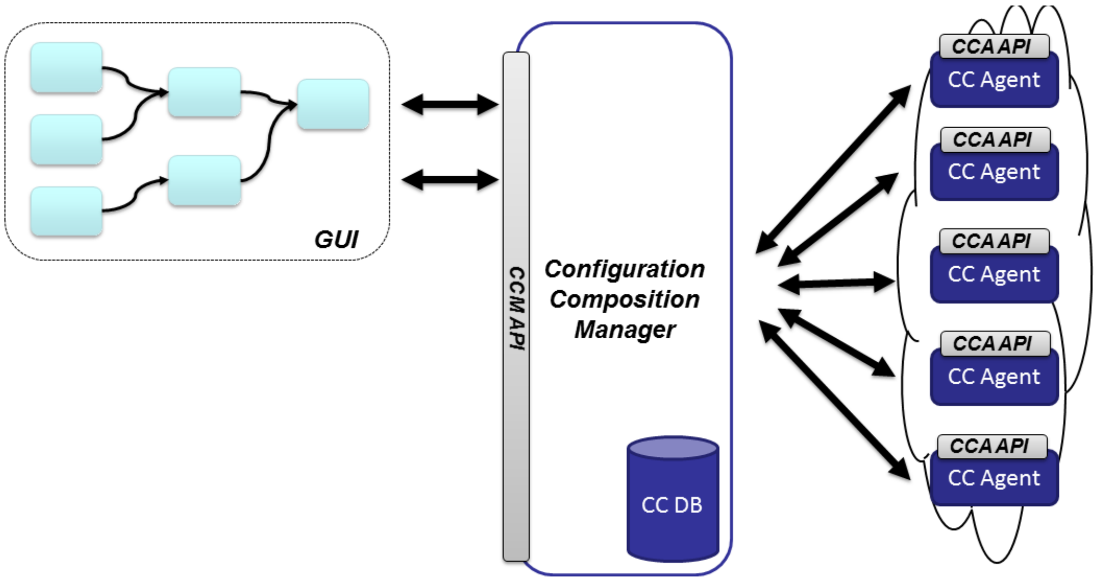

Figure 2: Configuration and Composition Framework architecture

The Configuration and Composition Manager is the module in charge of managing the configuration and composition processes of the other modules into the platform; it works as an interface between the GUI and the various components of the platform. The functionalities of CCM consist in the following:

- Notifies the tools on the status of the components available in the middleware.
- Retrieves the configuration from the CCA, when required through a REST interface.
- Updates the configuration of the components through the CCA via REST.

CCM is responsible for the management of the composition stage. In order to do this, the CCF leverages on the publish/subscribe paradigm (publish/subscribe, 2014), which allows the complete decoupling of the various component. The details about how CCF uses this paradigm will be provided in the section 5.1.

A CCA is associated with each component of the platform. It exposes get and set methods for the configuration parameters of a specific component to the CCM. The CCA operates actually the configuration commands coordinated by CCM. The association of an agent to each module makes the system more expandable and scalable from the point of view of configuration issues. One CCA is assigned for each component of the platform. CCA is responsible for:

- Register the component in the Configurations and Composition Manager.
- Handling the configuration parameters of the component.

Handling the interconnection of the components with each other, adding and removing input sources.

4. Configuration and Composition Framework API

In this chapter are reported the operations that can be managed by the REST API exposed by the CCM and CCA.

4.1 Targets concerning creating visibility of the project

Operation	Parameters	Returns	Description
register()	Component identification name	-	Registers the component in the manager when it is started.
notify()	Component identification name Component status (available, unavailable)	-	Used to notify the configuration and composition tools when a component availability change (it becomes available or unavailable).
getConfigurationForm()	Component identification name	The parameters that can be configured in the component.	This operation is used to get the list of the parameters that can be configured in the component (with the current values).
setConfiguration()	Component identification name ConfigurationForm	-	Updates the configuration of the component, setting the values passed as parameter.
addInput()	Component identification name Input identification	-	Adds to the component a new input source.
removeInput()	Component identification name Input identification	-	Removes from the component an input source.

4.2 Composition and Configuration Agent API

Operation	Parameters	Returns	Description
getConfigurationForm()	-	The parameters that can be configured in the component.	This operation is used to get the list of the parameters that can be configured in the component (with the current values).
setConfiguration()	configurationForm	-	Updates the configuration of the component, setting the values passed

			as parameter.
addInput()	Input identification	-	Adds to the component a new input source.
removeInput()	Input identification	-	Removes from the component an input source.

5. Examples

In this chapter are described some examples of the two different stages of the provisioning task: the configuration and the composition.

5.1 Composition operations

A Graphical GUI, designed and implemented within task T7.4, is able to show a graphical representation of all available components of the platform. The availability of components are managed through the registration of a component within CCM and the notification of the component to the GUI. The available components are those that may contribute to the realization of the final application and can be connected among them, through graphical links. This GUI can be called Composition tool.

Whenever is defined a link between two components of the platform, the addInput operation of CCM is called. CCM will contact the correct CCA in order to ask it to subscribe itself to events/outputs published by another components on a publish/subscribe node. The publish/subscribe node can be created and managed by the CCM but external publish/subscribe services can also be leveraged.

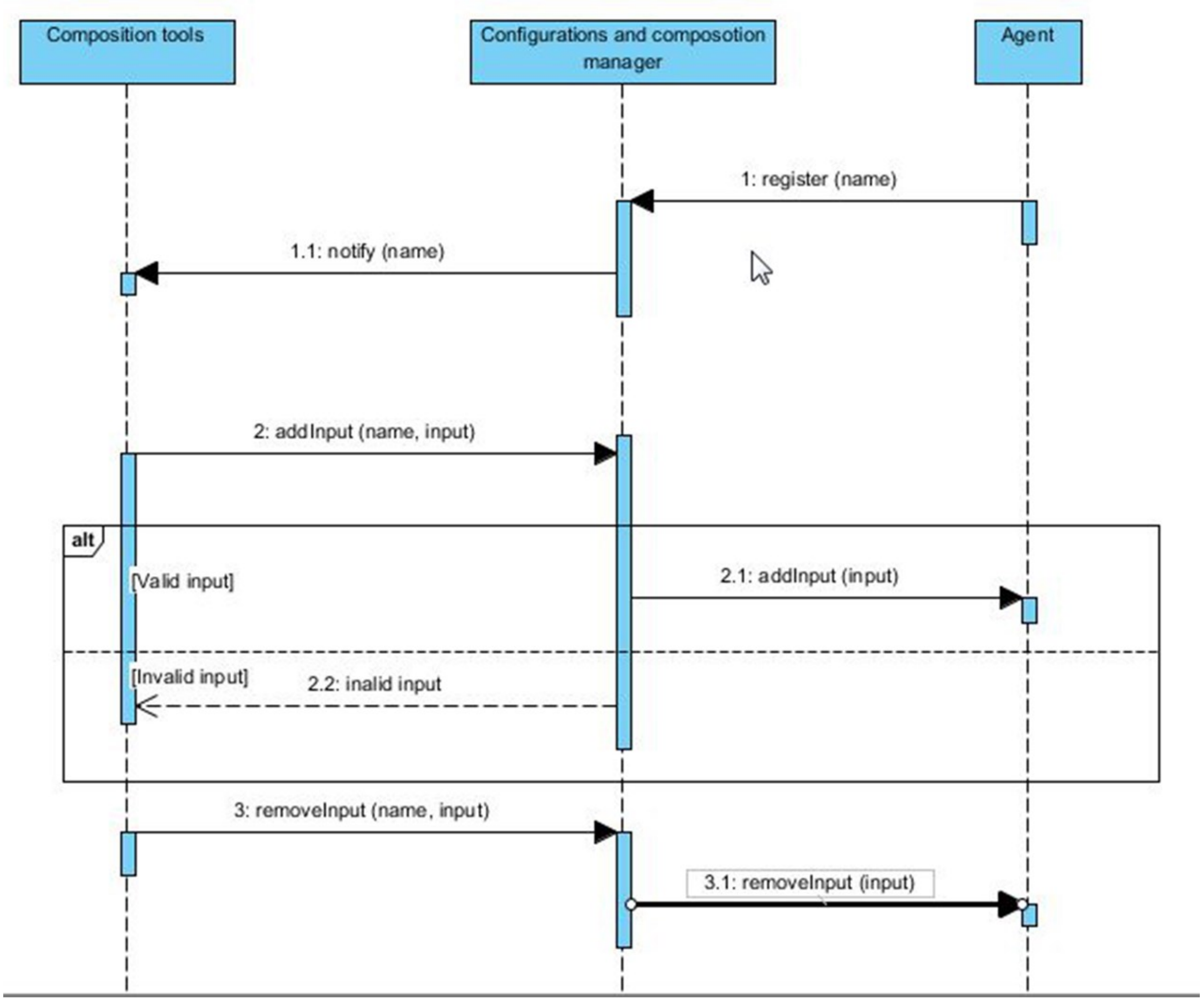

Figure 3: Sequence scheme for composition

5.2 Configuration operations

The same GUI used for the composition can be also used for the configuration of the modules. In this case, when the user selects the module, its configuration form is retrieved by the CCM querying the agent. The configuration form contains all the parameters that can be configured on the component. The form will be presented to the user, which can insert the values in it. Finally, the values inserted are sent to the agent (through the CCM), which sets them in the component

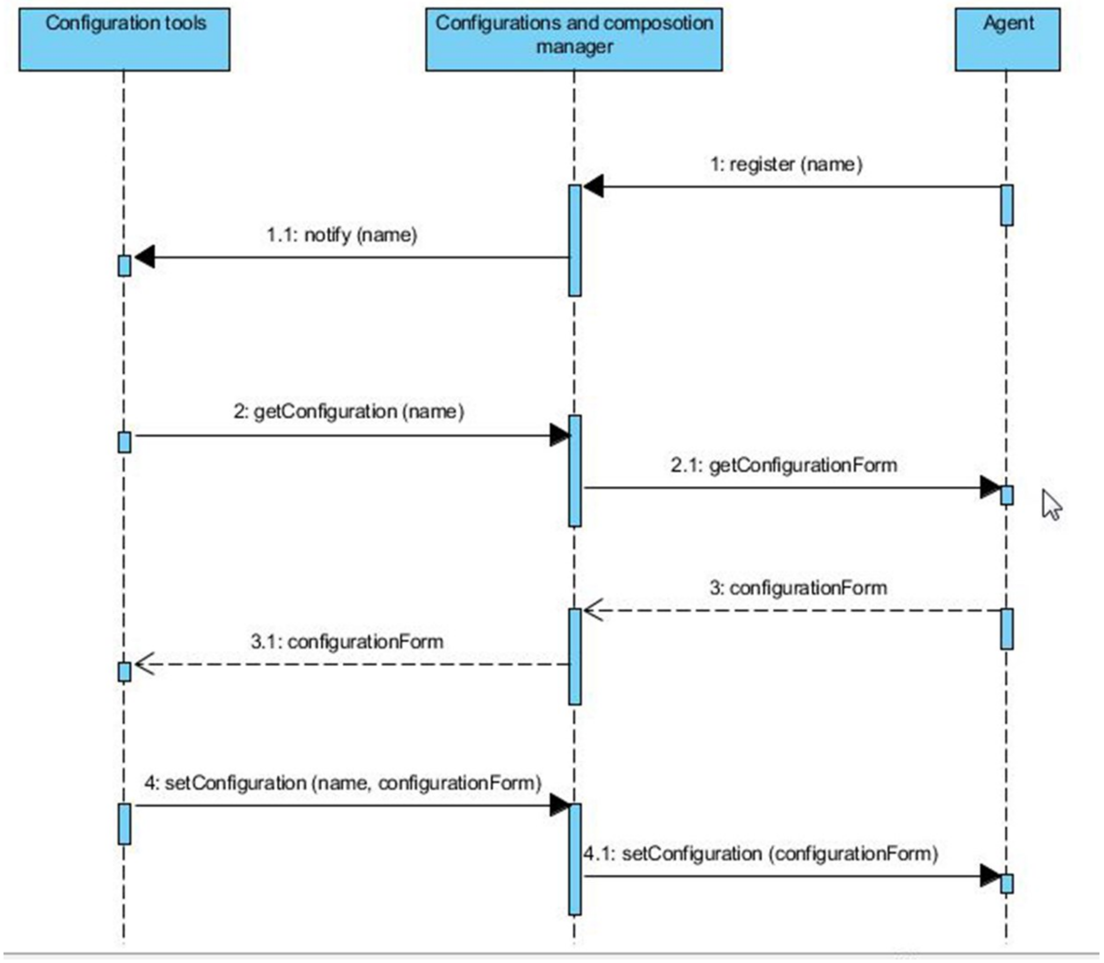
Configuration tools
Configurations and composotion manager
Agent
1: register (name)
1.1: notify (name)
2: getConfiguration (name)
2.1: getConfigurationForm
3: configurationForm
3.1: configurationForm
4: setConfiguration (name, configurationForm)
4.1: setConfiguration (configurationForm)

Figure 4: Sequence scheme for configuration

6. Summary & Conclusion

In this deliverable, the current Composition and Configuration Framework (CCF) architecture and features have been outlined. It manages the provisioning of the platform performing two different stages: the composition of the application and its configuration.

This document presented a first design version of CCF, for next release the APIs are going to be refined and/or extended. The Configuration and composition Agent API are intended to be part of the interfaces exposed by the Service Proxy.

7. Bibliography

publish/subscribe. (2014). Tratto da http://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern

SNMP . (2014). Tratto da http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol

List of Figures and Tables