SCIDIP-ES
SCIENCE DATA INFRASTRUCTURE FOR PRESERVATION - EARTH SCIENCE

D21.3 Orchestration Service Installation, Deployment and User Manual

Work package	WP21	Services/Toolkits Development and Adaptation
Task	T21.1 Orchestration Service Development	
Author (s)	Pasquale Andriani, Francesco Servidio	ENG
Authorized by		
Reviewer	Shirley Crompton	STFC
Doc Id		
Dissemination Level	~~CONFIDENTIAL/~~PUBLIC	
Issue	1.0	
Date	28/02/2014	

Abstract:

This document represents the Deployment and User Manual for the Orchestration Service developed in the frame of SCIDIP-ES project updated for the month 30 release. This document contains all useful information on how to install, configure and use the Orchestration Service.

Document Log

Date	Author	Changes	Version	Status
12/02/2014	Pasquale Andriani, Francesco Servidio	Created base document from D21.4. Added Blueprint Thinkerpop and OrientDB graph db as replacement of the relationational database persistence layer.	0.1	Draft
28/02/2014	Pasquale Andriani	Full revision according to Shirley Crompton review. Improved design section. Added API documentation.	0.8	Draft
28/02/2014	Pasquale Andriani, Shirley Crompton	Final editing	1.0	FINAL

TABLE OF CONTENTS

1 Introduction

1.1 Purpose and Scope

This document provides an overview of the Orchestration Service focusing in particular to its installation, configuration and usage.

1.2 Who should read this document

System administrators and users who wish to install, deploy, understand and operate the Orchestration Service.

1.3 System Context

The Orchestration Service is developed as part of the SCIDIP-ES e-infrastructure. Its role is to act as a collector of information and provides a single entry point for exchanging intelligence about changes, made by human or software agents, which might impact the long-term use and/or access of preserved data. The Orchestration Service could then identify the implication of these changes through the Gap Identification Service and deliver the output to the interested audience so that they may take the most appropriate action/s to mitigate any risk arising from the change event.

1.4 Release Notes

M30 release of Orchestration Service is a reengineered version based on results produced by the CASPAR[1] project as well as additional new features specified in D12.1 and WP32 relating to scalability and robustness issues. In this release, the main enhancement is to refactor the persistence layer by replacing the embedded relational database (H2) with a more robust graph database. The new solution is based on:

- OrientDB[2] - as a NoSQL Graph Database instance
- Blueprints Tinkerpop[3]- used as an abstraction layer to interact with the graph database. JDBC was used to interact with the superseded H2 relational database

1.5 Download information

Software code is available on Sourceforge at:

- **http://sourceforge.net/projects/digitalpreserve/**

Orchestration Service subversion repository is available at:

- **svn://svn.code.sf.net/p/digitalpreserve/code/SCIDIP-ES/software/services/orchestration /**

To directly download the recent releases, please visit:

[1] CASPAR Project: http://www.casparpreserves.eu/
[2] OrientDB: http://www.orientechnologies.com/orientdb/
[3] Blueprints Tinkerpop: https://github.com/tinkerpop/blueprints/wiki

- http://nexus.scidip-es.eu/content/repositories/releases/eu/scidipes/services/orchestration/

1.6 Licence Information and Terms of Use

2 Software Design

The Orchestration Service is designed to support data curators and other users to notify events/activities which may bear certain preservation risk to their data holdings so that they may take corrective actions according to established preservation plans.

The Orchestration Manager deals with the following entities:

- the notification **message,**
- the **topics** of notifications (descriptions of events/activities/objects) used by the ES-community in a hierarchical structure (taxonomy),
- the **team/person/software agent** to be informed about changes (Subscriber role),
- the **team/person/software agent** entitled to share information about changes (Publisher role).

Figure 1 shows the Public APIs which are described in Section 5.1.

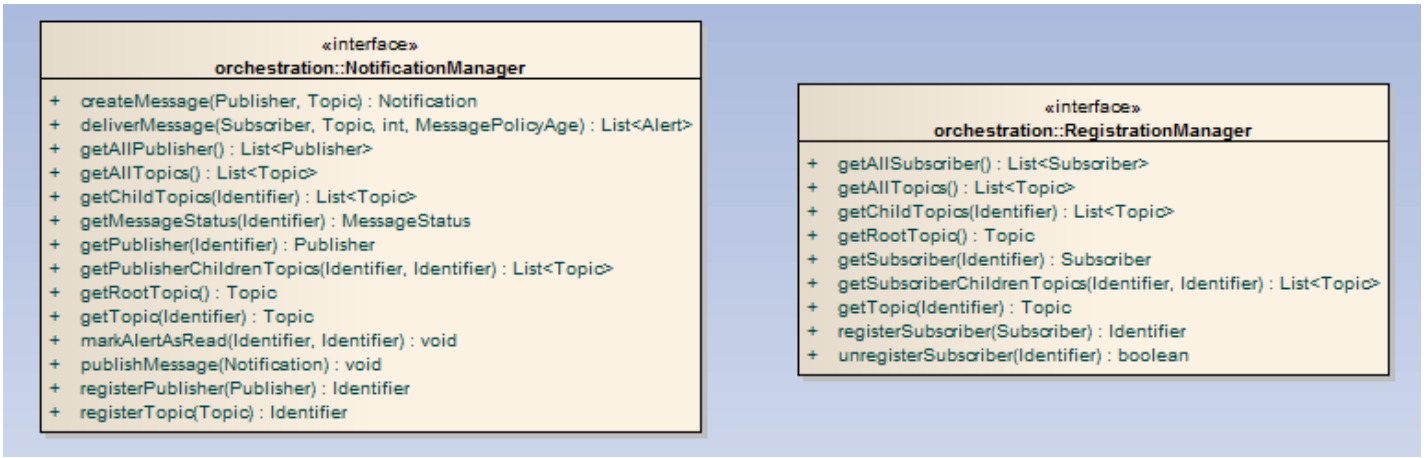

Figure 1 - Orchestration Service Interfaces

An extract of the domain model, showing the relationship between the entities presented is shown in Figure 2.

[4] Apache Licence, Version 2.0 - http://www.apache.org/licenses/LICENSE-2.0

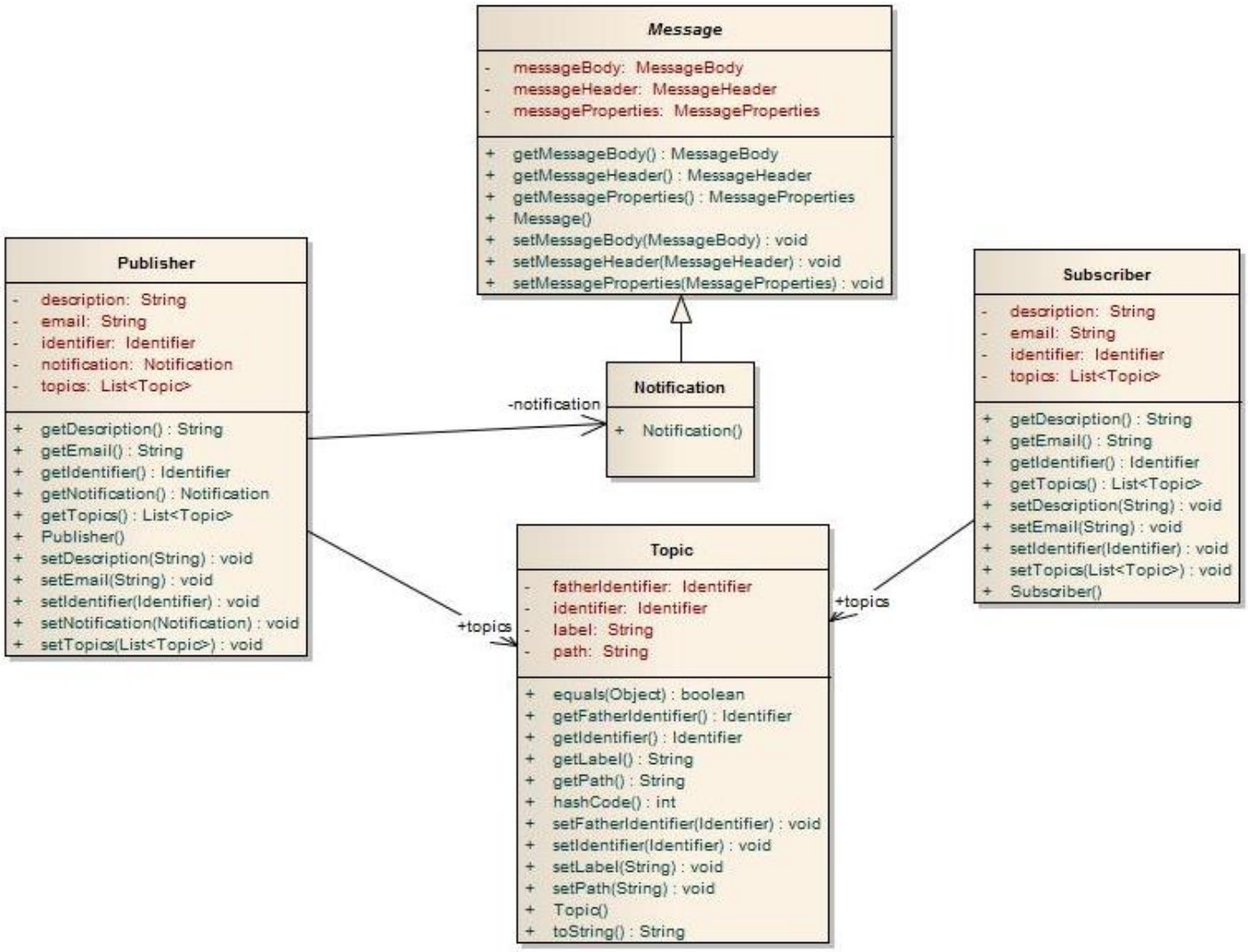

Figure 2 – Orchestration manager domain model

3 Installation Guide

This section explains how to install the Orchestration Service which involve deploying the three components depicted in the component diagram in Figure 3.

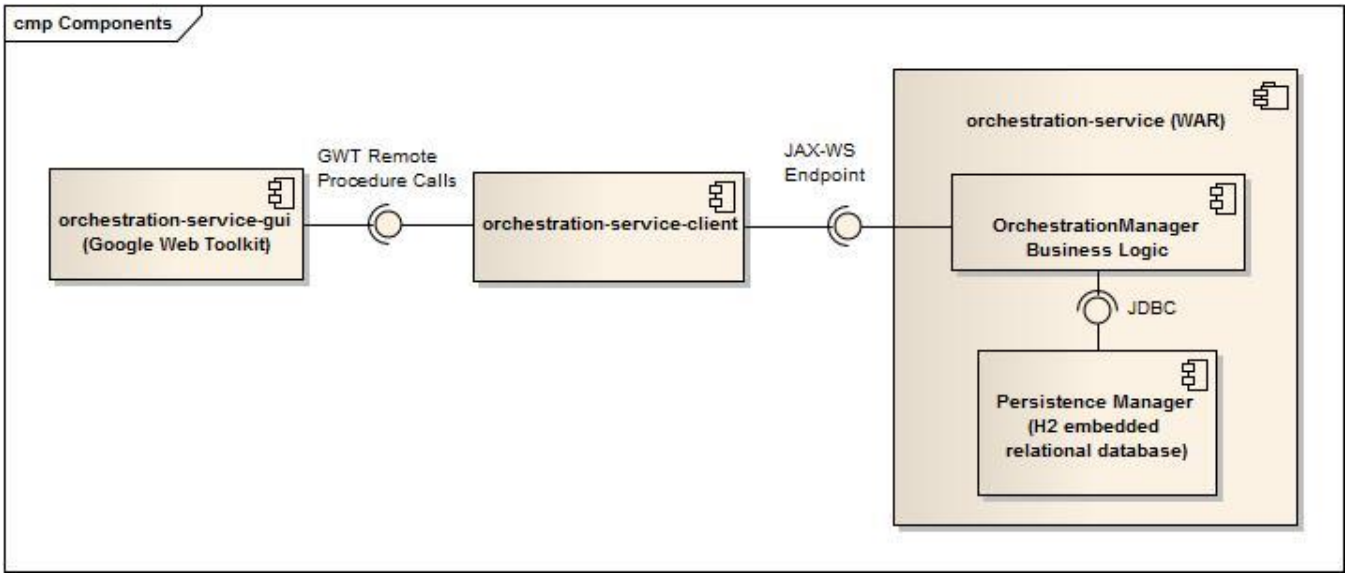

Figure 3 – Orchestration Service component model

3.1 Overview

Recent releases of the artefacts are available on the SCIDIP-ES Nexus repository (see Section 1.5) and each release contains:

- **orchestration-service.war** is the service to be deployed and, at the time of writing, embeds also the persistence layer
- **orchestration-service-client-*.jar** is a Java client library that have to be used to invoke the orchestration-service
- **orchestration-service-gui.war** is a GUI web application frontend that, by embedding the orchestration-service-client-*.jar, interacts directly with any deployed orchestration-service.war, i.e. the web service.

Orchestration Service uses either "H2" embedded database for the persistence of data, or a GraphDB, which improves performance when high volume of data is stored. OrientDB is the default database adopted by Orchestration Service. This may easily be replaced with a different one (e.g. Neo4J) as Orchestration Service uses Tinkerpop Blueprints as an abstraction layer to decouple interactions with the physical database. Please refer to instructions below on how to deploy a GraphDB.

3.2 Prerequisites

3.2.1 Software prerequisites

Software prerequisites respect SCIDIP-ES guidelines and include:

- a graphDB, e.g. OrientDB 1.5 Graph Edition[5], which is the default
- Java SE Development Kit 6[6] (for installing and running Tomcat)
- Tomcat version 7[7]

[5] OrientDB 1.5 Graph Edition - https://github.com/orientechnologies/orientdb/releases/tag/1.5
[6] Java SE Development Kit 6 - http://www.oracle.com/technetwork/java/javase/downloads/jdk6downloads-1902814.html

In the next section it will be assumed that the Tomcat installation location, $TOMCAT_HOME, is known.

3.3 Orchestration Service Installation

Assuming that Tomcat is installed and running:

1. **Deploying the Orchestration Web Service**: copy **orchestration-service.war** to $TOMCAT_HOME/webapps

   Depending on your Tomcat configuration, two URLs are available to test if the orchestration-service is running:

   http://localhost:8080/orchestration-service/NotificationManager?wsdl

   http://localhost:8080/orchestration-service/RegistrationManager?wsdl

2. **Deploying the Orchestration client web application**: copy **orchestration-service-gui.war** to $TOMCAT_HOME/webapps. Set the properties *host* and *context* for accessing the deployed orchestration-service by editing $TOMCAT_HOME/webapps/orchestration-service-gui/WEB-INF/classes/eu/scidipes/orchestration/gui/orchestrationgui.properties. For example, if orchestration-service is available and accessible on http://localhost:8080/orchestration-service, set

   host=*localhost:8080*
   context=*orchestration-service*

3. **Configuring the persistent store**: Check the web.xml file under the folder (TOMCAT_HOME/webapps/orchestration-service/WEB-INF) and make sure that the "context parameter" for "db.type" has "param-value" set to "Blueprints" instead of "H2":

```
<context-param>
     <param-name>db.type</param-name>
     <param-value>Blueprints</param-value>
</context-param>
```

4. **Installing and Configuring OrientDB 1.5 Graph edition** (if not already installed): After installing OrientDB, it is necessary to perform the following configurations:

   a. **Add and enable OrientDB user**: Add user "orient" in **orientdb-server-config.xml** (under the config folder of OrientDB installation). The installer is free to use any user name and password. The followings are the default credentials used by the Orchestration-service component:

```
<users>
     <user resources="*" password="[GENERATED]" name="root"/>
     <user resources="connect,server.listDatabases,server.dblist"
password="guest" name="guest"/>
```

[7] Tomcat version 7 - http://tomcat.apache.org/tomcat-7.0-doc/

```
    <user resources="*" password="s3cret" name="orient"/>
</users>
```

b. **Configuring graph properties**: Revise the graph.properties under the folder (TOMCAT_HOME/webapps/orchestration-service/WEB-INF/classes) to configure user credentials and path to the OrientDB installation as follows:

```
driver = orient
# driver = neo4j
user = orient
password = s3cret
host = localhost
#Property "path" is for orientdb
# path = /opt/orientdb-graphed-1.5.0
path = C:/orientdb-graphed-1.5.0
# Property "directory" is for neo4j
# directory = /opt/neo4j-community-1.9.5/
```

5. **(OPTIONAL for using Neo4j) Configuring Neo4j Graph Database**: Orchestration Service supports Neo4j graph database. To use it, uncomment the row “driver = neo4j” and relevant “directory” property. In this case, the driver and path properties for orientdb should be commented out.

You can now use Orchestration-service at http://localhost:8080/orchestration-service-gui from your web browser.

3.4 Uninstallation

Orchestration Service can be uninstalled by undeploying the two war packages **orchestration-service.war** and **orchestration-service-gui.war:** delete the following objects under $TOMCAT_HOME/webapps:

- **orchestration-service.war** file
- **orchestration-service** folder
- **orchestration-service-gui.war** file
- **orchestration-service-gui** folder

If OrientDB has been used for graphDB persistence, also uninstall OrientDB.

4 Using SCIDIP-ES Orchestration Service

This section is divided into two subsections:

- Section 4.1 Getting started and initial configuration – this illustrates how to set up the Orchestration Service in line with user requirements

- Section 4.2 Day-to-day usage - how to use the Orchestration Manager on a day-by-day basis to perform common tasks.

The Orchestration Service is assumed to be accessible @ http://localhost:8080/orchestration-service-gui/ .

The home page of the Orchestration Service shows the different menu choices for accessing the three different views, as shown in Figure 4:

- **Browse Alert** – allows a Subscriber to browse the notification messages according to its own subscribed Topics
- **Registration** – allows the registration of new Topics into the topic hierarchy and the registration of new Publishers and Subscribers with their chosen Topics subscription
- **Publisher** – allows the publication of new notification messages

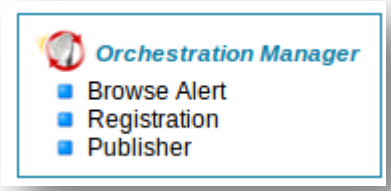

Figure 4 – Orchestration Service main menu

4.1 Getting started and initial configuration

4.1.1 Creating the Topic hierarchy

A Topic is used to model a change event which can be about:

- new version, bug solution, new documentation, new format, format obsolescence
- a software used for accessing, searching, processing data
- a dataset archived in the OAIS environment
- the target user communities, etc.

To add a Topic (e.g. Designated Community, see Figure 5), you have to:

1. **select the parent topic** on the Topic hierarchy - i.e. Topics/Change
2. **write the name** of the topic who want to add into the input field Topic Section – i.e. Designated Community
3. **click the button Add Topic**

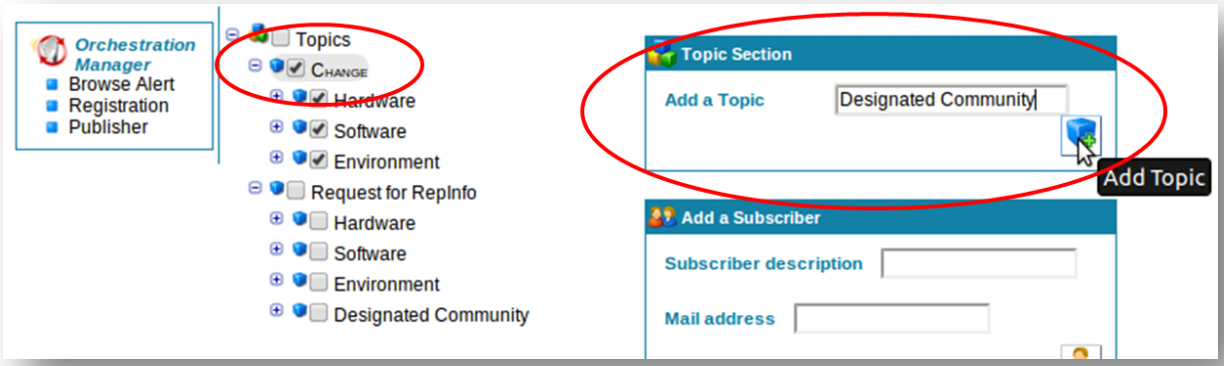

Figure 5 – Add a Topic

The result is shown in Figure 6, where the new Topic is available on the Topic hierarchy and a confirmation message appears if the operation was successful.

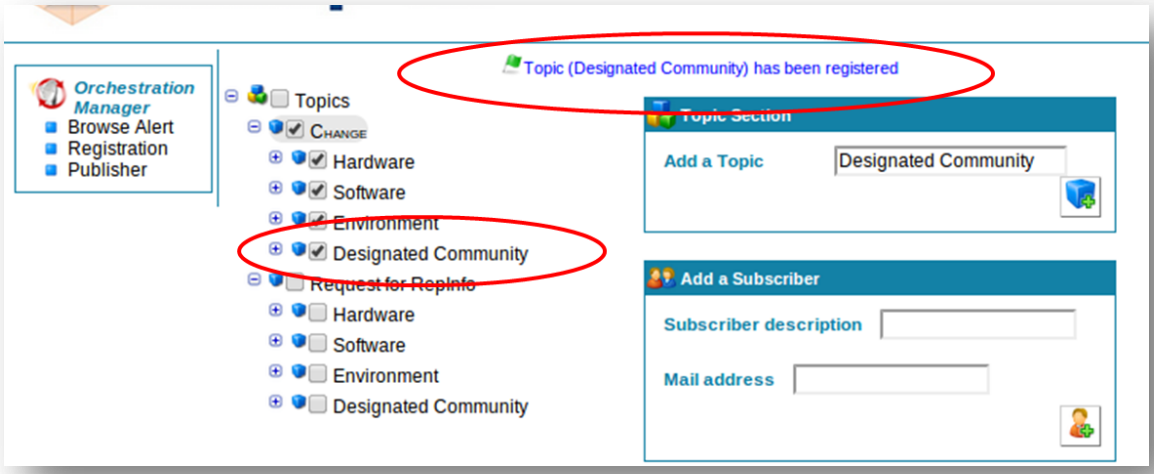

Figure 6 – Add Topic result

4.1.2 Creating a Subscriber

To add a Subscriber (e.g. *HW/SW Pasquale*, see Figure 7), you have to:

1. **select the topics** on the Topic hierarchy - i.e. Topics/Change/Hardware and Topics/Change/Software
2. **fill the Add a Subscriber box** with a name of the subscriber and his email address
3. **click the button Add Subscriber**

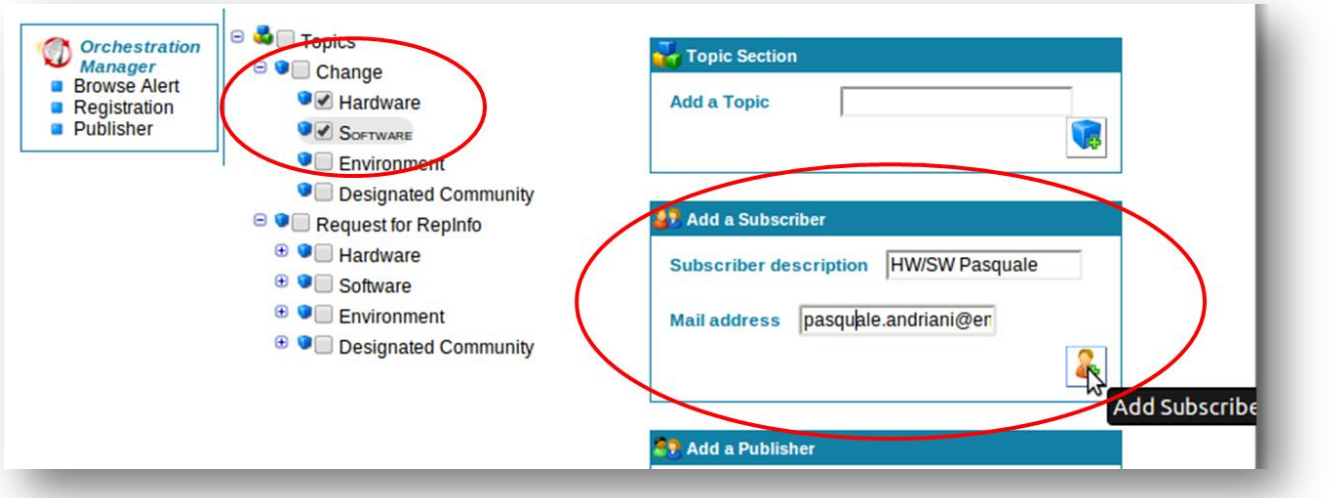

Figure 7 – Add a Subscriber

The result is shown in Figure 8, a confirmation message appears if the operation was successful.

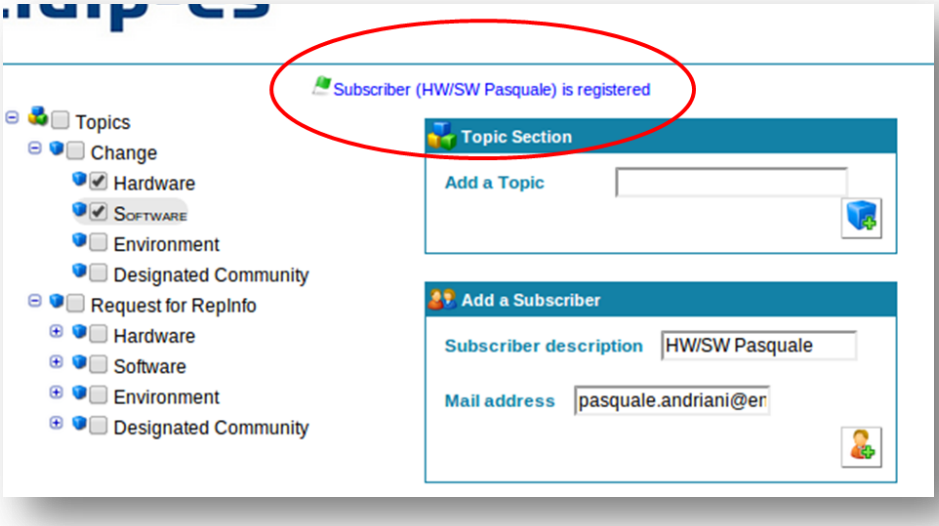

Figure 8 – Add Subscriber result

You can also open the **Browse Alert** view (Figure 9) and by clicking the Subscriber's name (*HW/SW Pasquale)* added above to display the topics subscribed to.

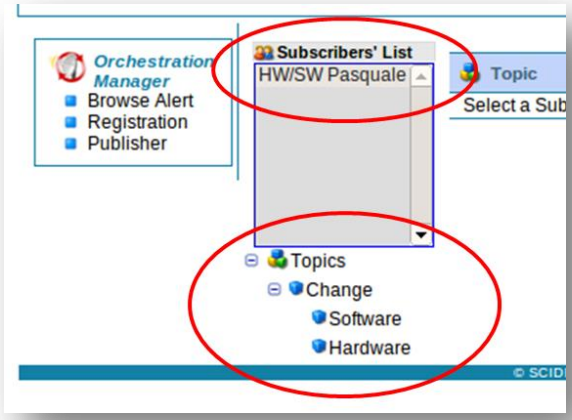

Figure 9 – Subscriber & Topic hierarchy

4.1.3 Creating a Publisher

To add a Publisher (e.g. *HW/SW Publisher*, see Figure 10):

1. **select the topics** on the Topic hierarchy - i.e. Topics/Change/Hardware, Topics/Change/Software, Topics/Request for RepInfo/Hardware and Topics/Request for RepInfo/Software
2. **fill the Add a Publisher box** with a name of the publisher and his email address
3. **click the button Add Publisher**

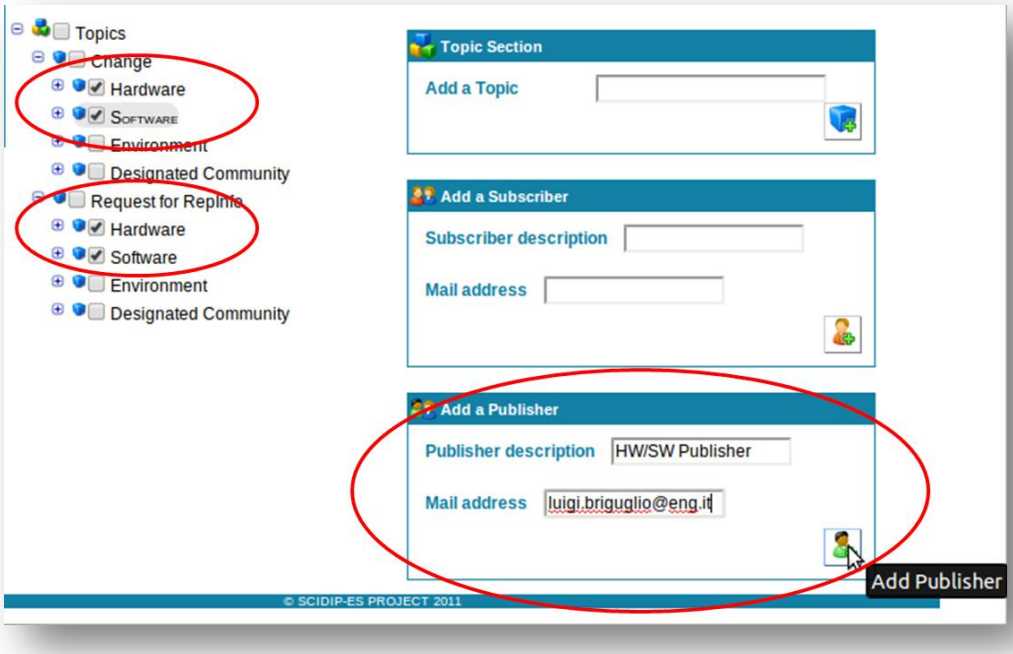

Figure 10 – Add a Publisher

The result is shown in Figure 11, a confirmation message appears if the operation was successful.

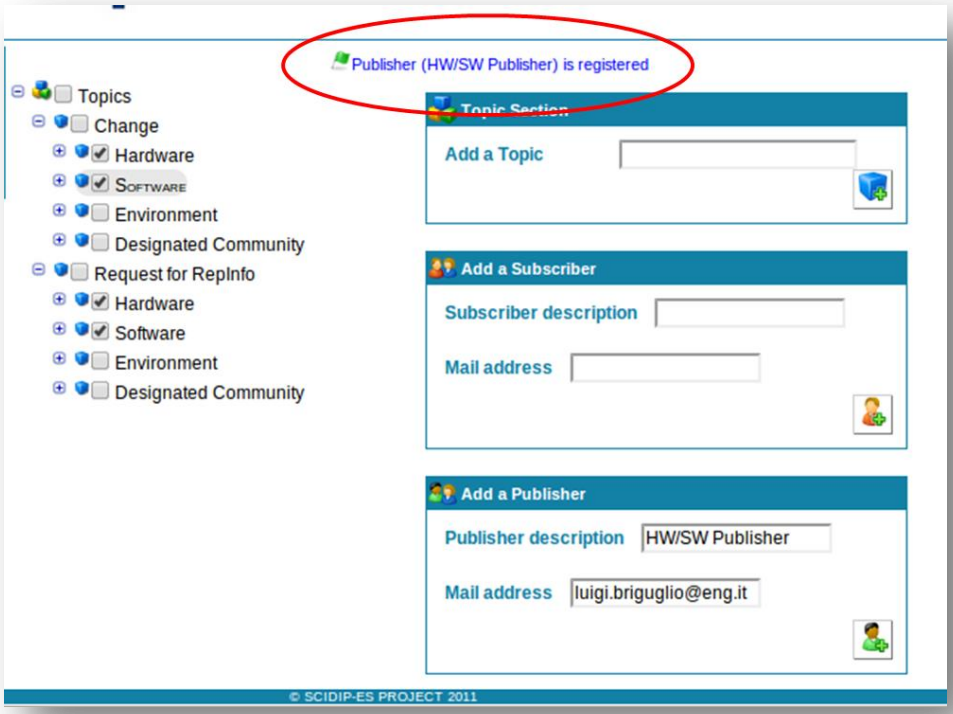

Figure 11 – Add Publisher result

You can also open the **Publisher** view (Figure 12) and by clicking the Publisher's name that has just been added to load his partial topic hierarchy.

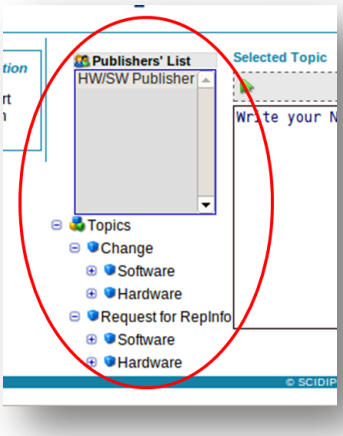

Figure 12 - Publisher & Topic hierarchy

4.2 Day-to-day usage

This section documents the two main operations that can be performed with the Orchestration Service when using it on a day-to-day basis:

- A Publisher **creating a notification** on a Topic
- A Subscriber **checking available notifications** by filtering the subscribed topics

4.2.1 Creating a notification

You can create a notification by opening the Publisher view and:

1. click on the Publisher's name,
2. browse the topic hierarchy and select the topic on where you want to publish the notification
3. write the content of the notification (since M18 release free text is allowed)
4. click on the Send button

An example of the above steps is shown in Figure 13.

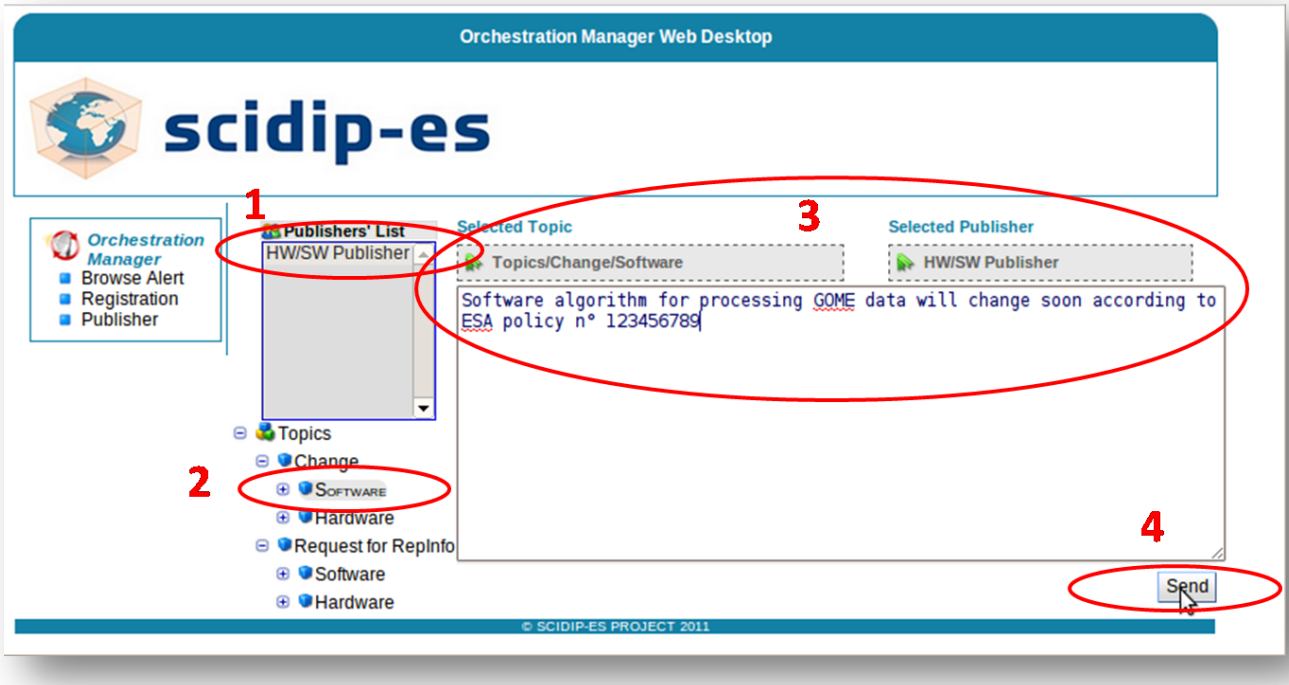

Figure 13 – Send Notification

4.2.2 Checking and browsing the available notifications

As a subscriber, you can browse notifications sent to you according to your subscribed topics by opening the Browse Alert view and:

1. click on the Subscriber's name,

2. browse the topic hierarchy and select the topic about which you would check the available notification
3. if available, a list of notification messages will be displayed. You can expand each message to view its details and relative content.

An example is shown in Figure 14.

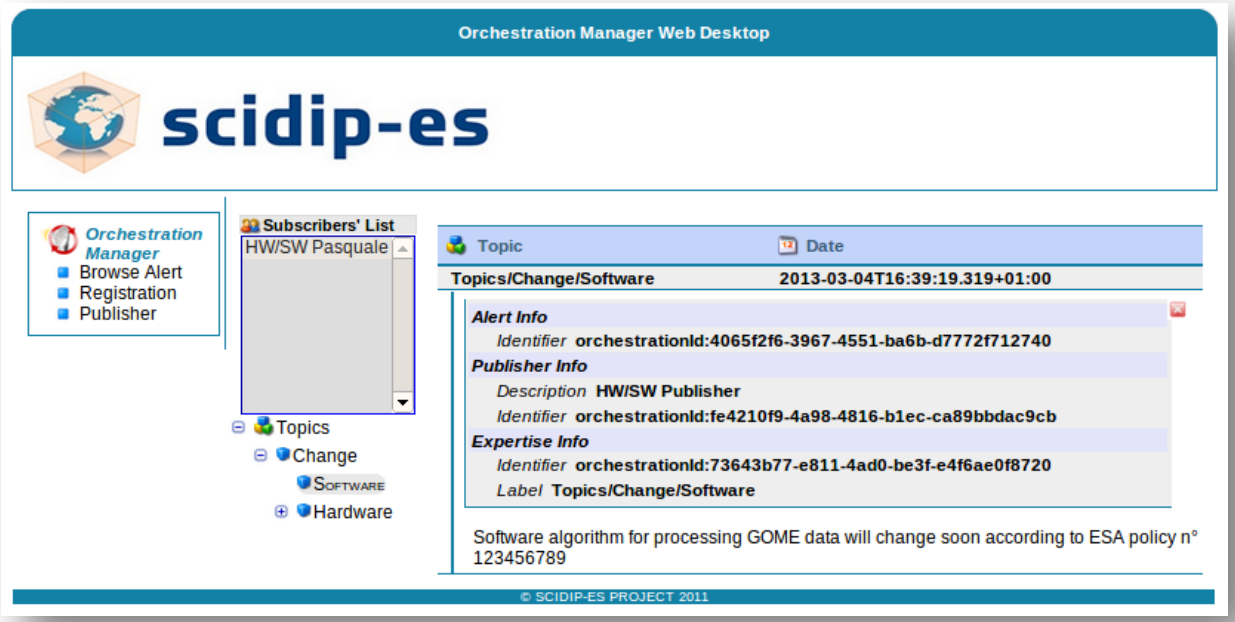

Figure 14 – Browse Alert view

5 Reference Manual

5.1 Public APIs

The Orchestration Manager provides its functionality through two interfaces, namely NotificationManager and RegistrationManager. The operations detailed in the tables hereafter.

5.1.1 Notification Manager APIs

Method Summary	
Notification	`createMessage(Publisher publisher, Topic topic)` Orchestration Manager prepares a Notification message.
List<Alert>	`deliverMessage(Subscriber subscriber, Topic topic, int maxBunch, MessagePolicyAge policyAge)` A registered subscriber (i.e. users/systems) obtains a bunch of alerts

	according to his expressed topics and policyAge.
List<Publisher>	`getAllPublisher()` POM retrieves all registered publishers.
List<Topic>	`getAllTopics()` Obtain all registered topics.
List<Topic>	`getChildTopics(Identifier topicIdentifier)` Orchestration Manager resolves child topics starting from a specified topic identifier.
MessageStatus	`getMessageStatus(Identifier msgId)` Evaluate a summary about the status of a published Notification.
Publisher	`getPublisher(Identifier pubId)` Obtain a publisher instance by its unique identifier.
List<Topic>	`getPublisherChildrenTopics(Identifier pubId, Identifier topicId)` Obtain a list of children topics of topicId for a registered Publisher.
Topic	`getRootTopic()` POM retrieves root topic
Topic	`getTopic(Identifier topicId)` Obtain a registered topic by its unique identifier.
Void	`markAlertAsRead(Identifier alertId, Identifier subId)` A Subscriber can confirm - by a "mark as read" - its own alerts.
Void	`publishMessage(Notification notification)` A registered publisher publishes a notification message on Orchestartion Manager.
Identifier	`registerPublisher(Publisher publisher)` Register a publisher with its topic (and its relative children).
Identifier	`registerTopic(Topic topic)` Register a topic by defininig its label and its father Topic Identifier.

Table 1 – Notification Manager APIs

5.1.2 Registration Manager APIs

Method Summary	
List<Subscriber>	`getAllSubscriber()` Orchestration Manager retrieves all registered subscribers.
List<Topic>	`getAllTopics()` Obtain a list of registered topics.
List<Topic>	`getChildTopics(Identifier topicIdentifier)` Orchestration Manager resolves child topics starting from a specified topic identifier.
Topic	`getRootTopic()` Orchestration Manager retrieves root topic
Subscriber	`getSubscriber(Identifier subId)` Obtain a subscriber instance by its unique identifier.
List<Topic>	`getSubscriberChildrenTopics(Identifier subscriberId, Identifier topicId)` Obtain a list of children topics of topicId for a registered Publisher.
Topic	`getTopic(Identifier topicId)` Obtain a registered topic instance by its unique identifier.
Identifier	`registerSubscriber(Subscriber subscriber)` Register a subscriber to enable reception of alert messages for its specified topics.
Boolean	`unregisterSubscriber(Identifier subId)` Unregister a subscriber from Orchestration Manager, specifying its unique identifier.

Table 2 – Registration Manager APIs

5.2 APIs usage example

The public APIs can be used by including the classpath **orchestration-service-client.jar.** Extracts of Java code are appended below showing how to use the Orchestration Service.

```
import java.net.MalformedURLException;
import java.net.URL;

import eu.scidipes.orchestration.client.NotificationManager;
import eu.scidipes.orchestration.client.RegistrationManager;
import eu.scidipes.orchestration.client.utils.ServiceFactory;

public class OrchestrationManagerExample {
```

```
    private static final String notificationManagerURL =
"http://localhost:8080/orchestration-service/NotificationManager?wsdl";
    private static final String registrationManagerURL =
"http://localhost:8080/orchestration-service/RegistrationManager?wsdl";

    /**
     * @param args
     * @throws MalformedURLException
     */
    public static void main(String[] args) throws MalformedURLException {
        NotificationManager notificationManager = ServiceFactory
                .buildNotificationManager(new URL(notificationManagerURL));
        RegistrationManager registrationManager = ServiceFactory
                .buildRegistrationManager(new URL(registrationManagerURL));

        /*
         * use objects notificationManager and registrationManager for invoking
         * Orchestration Service APIs
         *
         * Topic rootTopic =
         * notificationManager.getRootTopic();
         * ....
         */
    }

}
```

Annex A. Figures and Tables

A.1. List of Figures

A.2. List of Tables

Annex B. Terminology

ACRONYM	DESCRIPTION
AIP	Archival Information Package
ARK	Archival Resource Key
CDMI	Cloud Management Interface
DOI	Digital Object Identifier
ES	Earth Science
GIS	Gap Identification Service
KB	Knowledge Base
OS	Orchestration Service
OWL	Web Ontology Language
PI	Persistent Identifier
PNM	Preservation Network Model
PURL	Persistent Uniform Resource Locator
RDF	Resource Description Framework
RepInfo	Representation Information
SNIA	Storage Networking Industry Association
SWKM	Semantic Web Knowledge Middleware
VM	Virtual Machine
WP	Work Package
XAM	eXtensible Access Method
XML	eXtensible Mark-up Language