SEVENTH FRAMEWORK PROGRAMME

Networked Media

Specific Targeted Research Project

SMART

(FP7-287583)

Search engine for MultimediA environment generated contenT

D4.3 Integrated Edge Server

Due date of deliverable: 31-01-2013

Actual submission date: 15-03-2013

Start date of project: 01-11-2011 Duration: 36 months

Summary of the document

Code:	**D4.3 Integrated Edge Server**
Last modification: **State:**	07/03/2013 Final
Participant Partner(s): **Author(s):** **Fragment:**	AIT, Imperial, S3Log N. Katsarakis, A. Pnevmatikakis, R. Riveret and M. Tarquini No
Audience:	☒ public ☐ restricted ☐ internal
Abstract:	This report accompanies the prototype part of the D4.3 deliverable, the Integrated Edge Server. It summarises the components already discussed in D4.1 [3] and provides a manual for installing them, together with their 3rd party dependencies. It then proceeds presenting what is present in a freshly installed edge node in terms of metadata for connectivity trials with the rest of SMART, as well as how the edge node manager can create new feeds of metadata. The way the material is presented is by providing short summaries and links to the extensive public documentation of these components.
Keywords:	• Edge node • APIs
References:	See end of document

Table of Contents

1 Executive Summary

1.1 Scope

This deliverable is meant to present a first version of the open source implementation of the SMART edge servers. Since the architecture, the components of edge nodes and a guide to setting up your first edge node are already presented in the Smart Trac Web site[1], this short deliverable provides thus many pointers to this on-line documentation instead of reproducing it here. This deliverable will be completed in a second version due in month 30.

1.2 Audience

This deliverable is mainly addressed towards anyone interested in a quick overview of edge nodes with relevant pointers to the on-line documentation. The target audience includes in particular SMART project members and the open source community.

1.3 Summary

A smart edge node is server where data from local sensors, linked data and other information from social networks are collected and fused into high-level information. In this short deliverable, an overview of the architecture of the existing edge node, its components and a set up guide is given: in order to avoid a reproduction of the on-line documentation, we give pointers to this documentation for any further explanation.

1.4 Structure

The document is structured as follows:

- Section 2 provides an overview of the latest version of the layered architecture of the edge node.
- Section 3 presents the layers and their components by providing links to their public documentation.
- Section 4 is the core of this deliverable. On the one hand it serves as a quick guide on how to set up an edge node, referring to the details in the public documentation. On the other hand it explains how a newly installed edge node looks like, in terms of demo metadata feeds and tools for creating new ones.
- Section 5 concludes the deliverable.

[1] http://opensoftware.smartfp7.eu/projects/smart

2 Introduction

A Smart edge node is server where data from local sensors, linked data and other information from social networks are collected and fused into high-level information or events. An edge node is meant to cover a particular location such a neighbourhood or part of a city centre.

The overall architecture of an edge node has already been given in the deliverable D4.1, “Smart Distributed Knowledge Base and Open Linked Data Mechanisms” [3] and in the Web site of Smart, http://www.smartfp7.eu/. As a quick recap and an account of what is present in the first software release (see D6.2 [6]), the block diagram of the edge node and its three layers is given in Figure 1.

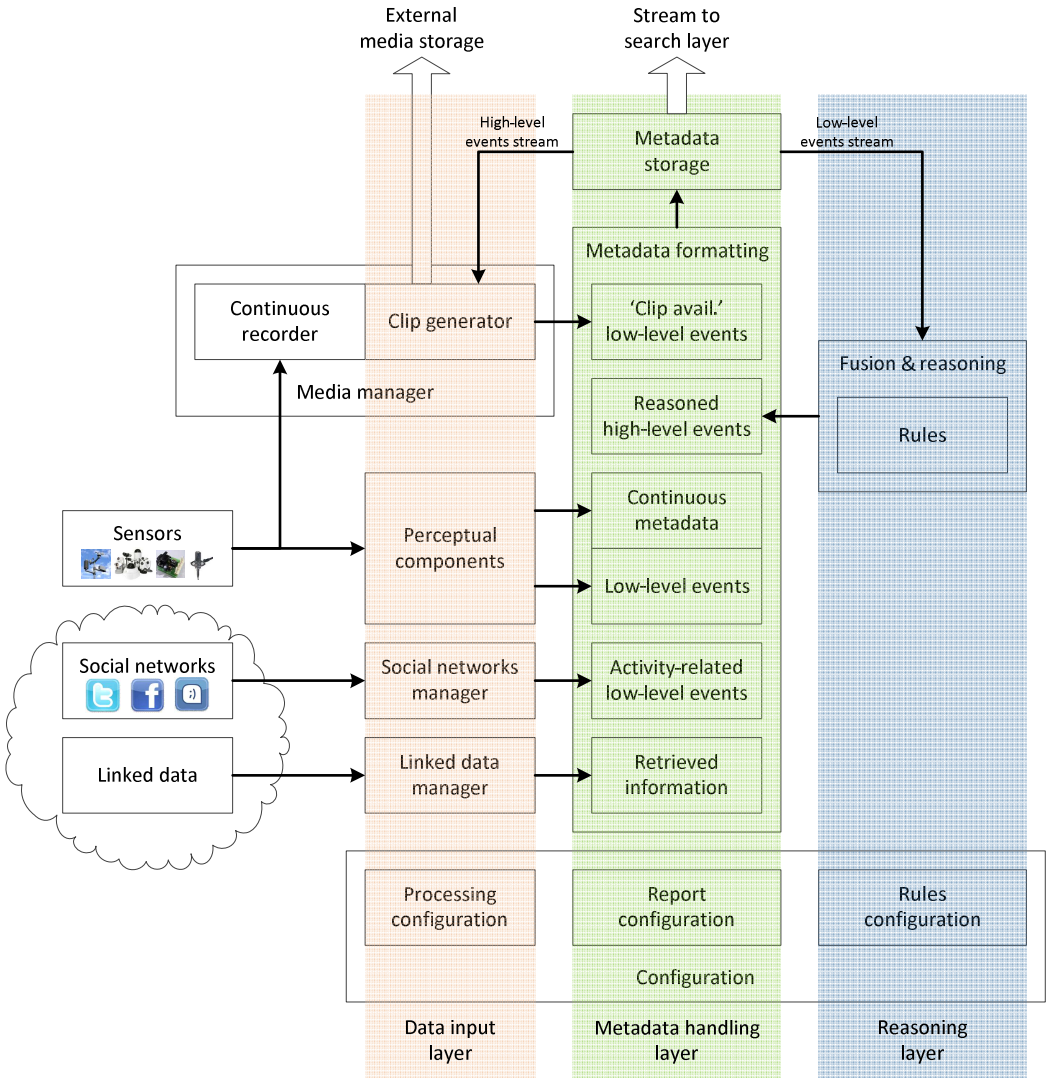

Figure 1. Edge node architecture.

The **data input layer** is meant to provide the mechanisms to abstract sensors, social networks and the linked data cloud as sources of data:

http://opensoftware.smartfp7.eu/projects/smart/wiki/Description#dataLayer

Its components process the signals from the physical sources (cameras, microphones, weather stations, chemical sensors, etc.) or virtual ones (linked data cloud or social networks) in order to extract metadata. The social network manager and a set of public perceptual components are included in the

first release. The implementation of the media manager is not yet fully tested, not is it integrated with the rest of SMART, hence it is not included in the first release.

The **metadata handling layer** handles the metadata streams from the data input layer and makes them available both to the reasoning layer and to the SMART search engine. This is achieved by storing every feed in a CouchDB database. The first release contains the full functionality of this layer, but updates and improvements are still under way regarding the common interface for metadata manipulation. We do not expect any changes to the direct interface. For more information refer to D4.1 [3] or http://opensoftware.smartfp7.eu/projects/smart/wiki/Usage#EdgeNodeInterfaces.

The **reasoning layer** encapsulates the Intelligent Fusion Manager (IFM). The IFM has two main goals: (i) infer high-level information from the collected data using some rule-based patterns and (ii) learn those patterns. Although the reasoning engine is in place and documented, it is not yet integrated with the rest of the system and thus is not included in the first release.

In the next chapter, we give some pointers to the Web documentation for the components of each of these layers.

3 Edge node components

The layered architecture of the edge node has already been described in D4.1 [3] and in our software documentation in Trac (http://opensoftware.smartfp7.eu/projects/smart/wiki/Description). Since this documentation is always the most up-to-date source of information, in this document we give a quick overview of the layers together with some pointers to the on-line public documentation.

3.1 Data input layer

The data input layer of the edge node includes the mechanisms to abstract sensors, social networks and the linked data cloud as sources of data. Its components process the signals from the physical sources (cameras, microphones, weather stations, chemical sensors, etc.) or virtual ones (linked data cloud or social networks) in order to extract metadata useful for the SMART users. The layer includes the following modules:

- The media data manager (MDM) produces the media clips and images to be used together with the metadata in answering queries. The complete documentation can be found in Trac, at http://opensoftware.smartfp7.eu/projects/smart/wiki/EdgeMediaMgr
- The perceptual components operate on the sensor data streams. They are detailed in Trac at http://opensoftware.smartfp7.eu/projects/smart/wiki/PerceptualComponents.
- The social network manager (SNM) provides services for requesting data from social networks like Twitter and Facebook, and also for filtering social network streams to provide social metadata. See http://opensoftware.smartfp7.eu/projects/smart/wiki/SocialNetworkManager for the detailed documentation in Trac.
- The linked data manager collects the linked data of interest to the edge node.

3.2 Metadata handling layer

Metadata handling for feed manipulation is the subject of D4.1 [3]. Lots of information can also be found in our software documentation in Trac, like its high-level description:

http://opensoftware.smartfp7.eu/projects/smart/wiki/Description#handlingLayer

and the description of its interface and API:

http://opensoftware.smartfp7.eu/projects/smart/wiki/Usage#EdgeNodeInterfaces

3.3 Reasoning layer

The reasoning layer consists of a so-called Intelligent Fusion Manager (IFM) that is meant to (i) infer high-level information from low-level information using some rule-based patterns and (ii) to learn those patterns. So, external and internal linked data from the knowledge base (KB) are fused to infer high-level information. The resulting high-level pieces of information are then parsed to ensure their proper format before being asserted to the knowledge base. We have identified the following essential requirements with regard to reasoning:

1. The recognition patterns must be rule-based. This shall ease the definition of patterns by human operators.
2. The reasoning must be able to deal with uncertainty. Indeed, sensors' data may come up with some incertitude or some confidence, and thus the reasoning engine has to deal with this uncertainty.
3. The reasoning must be able to deal with inconsistencies. For example, two sensors may produce inconsistent data.
4. The reasoning must be able to deal with incomplete set of data. If a sensor is broken or if the transmission of some data is blocked then the reasoning engine should be able to move on.
5. The reasoning must be defeasible in the sense that new pieces of data may invalidate conclusions previously derived.

To match these goals and requirements, we have chosen to build the IFM around a so-called Markov

Logic Network and its implementation Alchemy. The complete documentation about this IFM can be found in Trac, at http://opensoftware.smartfp7.eu/projects/smart/wiki/Description#reasoningLayer.

4 Edge Node Installation

The latest version of the guide to install SMART components and their 3rd party prerequisites can be found in Trac at http://opensoftware.smartfp7.eu/projects/smart/wiki/Installation. This documentation is the product of installation iterations by different SMART partners and the consolidation of their input.

Note that in order to download either of our binary or source code packages, you must first fill in our survey found at http://www.smartfp7.eu/content/download-survey, as detailed in D6.2 [6].

4.1 Installation of components

First, some 3rd party prerequisites are installed, following the guidelines in:

http://opensoftware.smartfp7.eu/projects/smart/wiki/Edge3rdParty

Then, the edge node common interface and tools are installed either using the binary version:

http://opensoftware.smartfp7.eu/projects/smart/wiki/EdgeInstallWar

or the source code:

http://opensoftware.smartfp7.eu/projects/smart/wiki/Development#SMARTEdgeNode

The same procedure is followed for the social network manager, using either the binary version:

http://opensoftware.smartfp7.eu/projects/smart/wiki/SNMinstallBin

or the source code:

http://opensoftware.smartfp7.eu/projects/smart/wiki/Development#SocialNetworkManager

Finally, the public perceptual components are installed:

http://opensoftware.smartfp7.eu/projects/smart/wiki/Development#SampleClients

Note that these components are written in C/C++ and that binaries are only provided for Windows installations. For other operating systems the source code should be used. Again this code is organised in Visual Studio projects.

A summary of the installation issues raised for the different test platforms is given here.

- **Windows**: No issues were raised during the installation of the third party components;
- **Linux**: No issues were raised during the installation of the third party component;
- **OSX**: Many more dependencies were discovered during the installation of CouchDB. At first, the installation of the XCode development environment is required. The installation of XCode cause the installation of the C compiler and all libraries needed. Before completing the installation, the procedure requires the installation of many other tools and dependencies: Homebrew, Erlang, Autoconf, Automake, Libtool. Finally CouchDB can be installed.

The installation of other SMART components complete without relevant issues.

4.2 Edge node initialisation information

The SMART edge node gathers sensors data, linked data and data from social networks that are of relevance to it. This can work because the linked data manager and the social network manager utilize some descriptors that are manually provided at the edge node setup phase. These descriptors are:

- GPS coordinates marking a rectangle of interest. Anything with location description that falls within this rectangle is relevant for the edge node.
- Geographic identifiers, i.e. names of the region. These are preferably local names, but it is up to the manager of the edge node to decide how wide the identifiers should be. E.g. for an edge node in a city, the geographic identifiers should be street names, name of a square or a neighbourhood, not the name of the city itself or the country.

- Names of places of interest in the vicinity of the edge node. Places related to education, entertainment, religion, shopping, etc. can be named here. Preferably activity outside these places should be captured by the edge node sensors, to allow for the edge node to fuse information from the sensors and the cloud. Examples include museums and shopping centres.
- Any known social network aliases of the places of interest, like Twitter names and hash-tags.

For example, consider the edge node at the Santander City Hall. The area is shown in Figure . The area of interest can span around the GPS coordinates +43° 27' 42.76", -3° 48' 36.27". Geographic identifiers related to the edge node are the streets Calle Jesús de Monasterio, Calle Amos de Escalante, Calle Miguel Artigas, Calle los Escalantes and Calle Isabel II and the square Plaza Ayuntamiento.

Figure 2. Google maps view of the city hall square in Santander.

Places of interest in the area include a museum (Museo de Arte Moderno y Contemporáneo de Santander y Cantabria), the city hall (Ayuntamiento de Santander), a restaurant (Heladería Regma), a café (Cafe-bar Ortiz), a travel agency (Viajes Ecuador) and a toy store (Jugueteria San Carlos).

Regarding instant messaging (e.g. Twitter) as information source, the edge node will send requests at initialization for periodic updates containing the geo-localised tweets near the GPS coordinates of interest and the hash-tags related to the area.

4.3 Edge node initialisation feeds

To facilitate edge node setup and early testing, every edge node installation comes with two feeds up and running:

- The demo feed contains some demo data. These data come from crowd analysis in front of the Santander City Hall. They are a result of visual processing and contain both continuous metadata and low-level events. They last for about 10 minutes, but can be arranged in a continuous loop for testing purposes.
- The @smartfp7 twitter feed. The data from this feed, unlike the static demo data, get updated as new tweets are made by the project.

After setup, the edge node manager can test the existing feeds. The list should contain the demo feed and the @smartfp7 twitter feed. Metadata retrieval from both feeds can then be tested.

Note that these feeds exist even if the edge node manager does not provide any initialisation information. After initial testing, real feeds should be setup and the initialisation ones should be deleted.

4.4 Built-in perceptual components

Our guide for installing and running the perceptual components shipped with the edge node installation can be found in Trac at http://opensoftware.smartfp7.eu/projects/smart/wiki/PerceptualComponents.

Currently two components are provided, the simple_camera that serves as an example on how to interface a camera and send metadata, and the simple_crowd_analysis that provides static crowd analysis metadata and is a public version of our full algorithm. Note that this is a growing list of components, as more will be made available by the SMART project or the community. Also note that it is up to the manager of the edge node to provide more perceptual components.

4.5 Working with the feeds

Feed manipulation includes the following tasks (ordered by level of complexity):

- Discover which data feeds are available in the edge node.
- Retrieve metadata from an existing feed.
- Append new metadata to an existing feed.
- Create a new feed.

These tasks can be achieved using two different interfaces, as discussed in [3], and presented in Trac: http://opensoftware.smartfp7.eu/projects/smart/wiki/Usage.

Feed creation requires the creation of one XML description, as detailed in Section 4 of [4], which may be quite demanding. To encourage the public to start building their own edge nodes, we are providing an automated tool where the edge node manager fills in a form and the XML feed description is automatically generated. The edge node manager can then either submit the generated XML to create the feed, or can review and manually modify it prior to submission. The tool is accessible from:

<edgeNodeURL>/SMARTEdgeNode/createFeed.html

A screenshot of the current version of the tool is shown in Figure 3. The folder icons allow the user to expand the detailed description. The links in the field names lead to additional information about the individual field.

4.6 Registering the Edge Node with SMART

There are two ways of making the Edge Node known to the Terrier search engine, the manual and the crowdsourcing, both discussed in Section 6.5 of [5].

When having multiple edge nodes providing information for answering the user queries, the search engine needs to know what to use. Our metadata streams inform the search engine that the crowd density is high or that people currently mostly move at a particular direction, but this is not enough to answer the query “crowded streets in Santander”. To answer this, Terrier needs to know two things:

- Where are the metadata collected from?

  This information is obtained from the edge node initialisation, made known to Terrier during the edge node registration phase. So Terrier knows that this edge node can be queried for information regarding particular general or finer geographic regions.

- What do the metadata refer to?

  Crowd density is measured by an edge node in Santander, but is it density in a mall, in a university amphitheatre or in the streets? Descriptors defining this are included in the feed description XML. The <Exposure> field (see section 4 of [4]) informs whether the stream comes from observing indoors or outdoors environment. The <URITags> field accepts keywords refining the disposition. Exactly where outdoors are the metadata collected from? The edge node is in Santander town hall, and there are feeds for crowd density from the square and from the street next to it. We are currently looking into standard ontological descriptors for these keywords. Should we not find them, we will define our own.

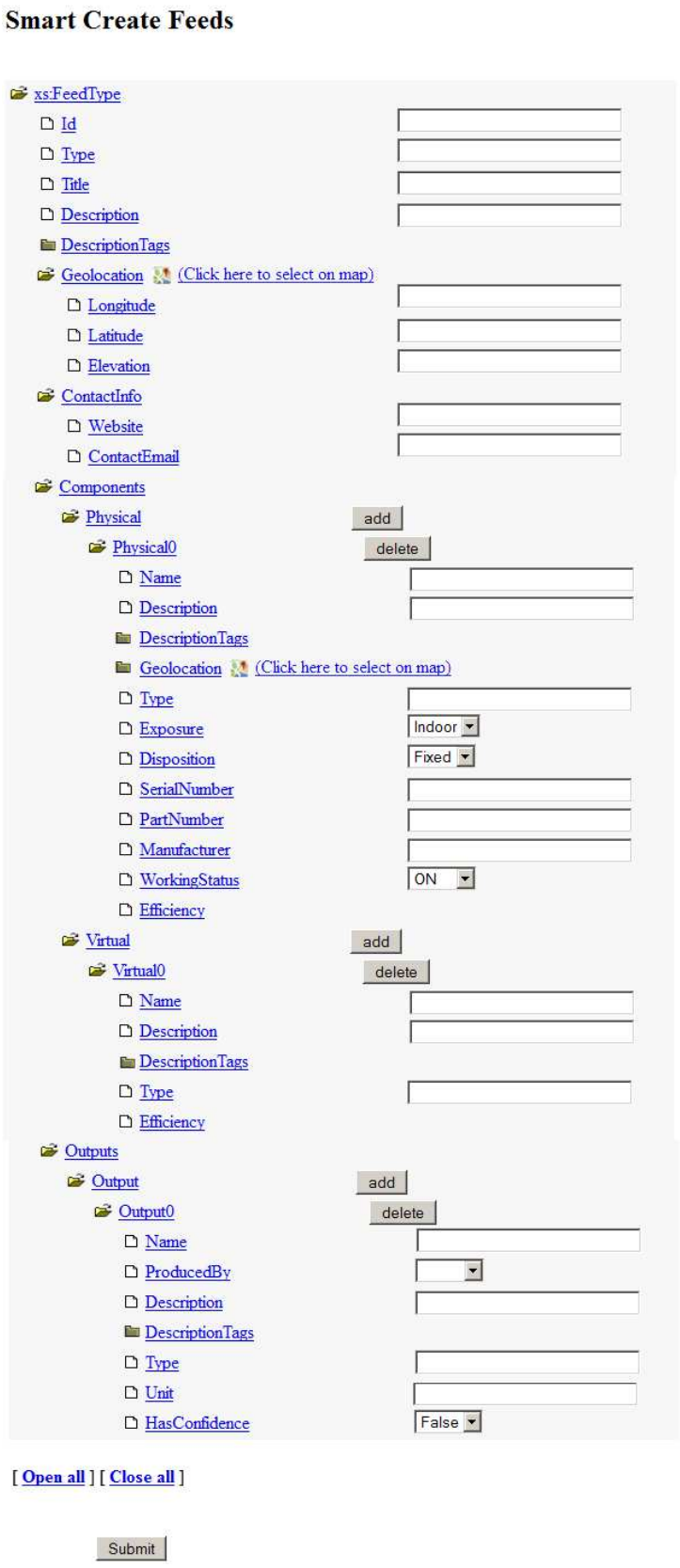

Figure 3. Automated tool for creating XML description of new feeds.

5 Conclusion

In this deliverable, an overview of the edge nodes' architecture, components and a set up guide was given. In order to avoid a reproduction of the on-line documentation, pointers to this documentation were given. This deliverable will be completed in a second version due in month 30. In this second version, we intend to give an overall overview of the edge node where more components and configuration options shall be fully integrated. Indeed, most of the partners' efforts are scheduled in these tasks for this period.

6 BIBLIOGRAPHY AND REFERENCES

1. Matthew Richardson and Pedro Domingos (2006). "Markov Logic Networks" in *Machine Learning* **62** (1-2): 107–136. http://www.cs.washington.edu/homes/pedrod/papers/mlj05.pdf
2. Stanley Kok, Marc Sumner, Matthew Richardson, Parag Singla, Hoifung Poon, Daniel Lowd, Jue Wang, Aniruddh Nath and Pedro Domingos (2010). "The Alchemy System for Statistical Relational AI", Technical Report, Department of Computer Science and Engineering, University of Washington, Seattle, WA. http://alchemy.cs.washington.edu
3. SMART project, D.4.1, “SMART Distributed Knowledge Base and Open Linked Data Mechanisms”
4. SMART project, D.3.1, “Sensors and multimedia data knowledge representation”
5. SMART project, D.5.1.a, “SmartReduce Engine”
6. SMART project, D.6.2, “Integrated Open Source Framework”