Project Acronym:	**FIRST**
Project Title:	**Large scale information extraction and integration infrastructure for supporting financial decision making**
Project Number:	**257928**
Instrument:	**STREP**
Thematic Priority:	**ICT-2009-4.3 Information and Communication Technology**

D7.4 Final Version of Integrated Financial Market Information System

Work Package:	*WP7 – Integrated Financial Market Information System*	
Due Date:	30/09/2013	
Submission Date:	30/09/2013	
Start Date of Project:	01/10/2010	
Duration of Project:	36 Months	
Organisation Responsible for Deliverable:	ATOS	
Version:	1.0	
Status:	Final	
Author(s):	Mateusz Radzimski Martin Riekert Achim Klein Lilli Gredel	ATOS UHOH UHOH UHOH
Reviewer(s):	Matjaž Jursic	JSI
Nature:	☐ R – Report ☒ P – Prototype ☐ D – Demonstrator ☐ O - Other	
Dissemination level:	☒ PU - Public ☐ CO - Confidential, only for members of the consortium (including the Commission) ☐ RE - Restricted to a group specified by the consortium (including the Commission Services)	
Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		

Revision history			
Version	**Date**	**Modified by**	**Comments**
0.1	20/08/2013	Mateusz Radzimski (ATOS)	First version of ToC provided
0.2	15/09/2013	Mateusz Radzimski (ATOS)	First version of components' list and infrastructure overview.
0.3	26/09/2013	Martin Riekert, Achim Klein, Lilli Gredel (UHOH)	Description of WP4 components build and deployment.
0.4	27/09/2013	Martin Riekert, Achim Klein, Lilli Gredel (UHOH), Mateusz Radzimski (ATOS)	Description of Ontology Update service.
0.5	27/09/2013	Matjaz Jursic (JSI)	Internal review
0.6	28/09/2013	Mateusz Radzimski (ATOS)	Addressing reviewer's comments, providing missing content.
1.0	30/09/2013	Mateusz Radzimski (ATOS)	Final release.

Executive Summary

This report describes the final version of the FIRST technical infrastructure. The document is accompanied by the final release of all components comprising the Integrated Financial Market Information System (referred to as IFMIS). While the previous WP7 deliverables focus more on providing rationale for technical design and implementation overview, this report aims at giving concrete instructions for acquiring FIRST results: downloading source code of the technical components, building them and deploying in a user-specified environment.

Many pieces of IFMIS Infrastructure were released as an effort of WP3, WP4, WP5 and WP6. And many of them are described in detail in their corresponding deliverables. This document aims at recapping those instructions, provide a comprehensive list of all IFMIS components and in some cases, provide updated instructions for deployment.

This deliverable also concludes the work done within WP7 by providing an integrated set of interoperable components, working seamlessly together in order to fulfil the objectives of the FIRST project.

Table of Contents

Index of Figures

Index of Tables

Index of Listings

Abbreviations and acronyms

IFMIS	Integrated Financial Market Information System
WP	Workpackage
ID	Identifier
DB	Database
RDBMS	Relational Database Management System
DACQ	Data Acquisition Pipeline
XML	eXtensible Markup Language
JAR	Java Archive
JDK	Java Development Kit
JVM	Java Virtual Machine
JEE	Java Enterprise Edition
SQL	Structured Query Language
SOAP	Simple Object Access Protocol
IDE	Interactive Development Environment
UI	User Interface
GUI	Graphical User Interface
DAL	Data Access Layer
HTML	HyperText Markup Language
WAR	Web application ARchive

1. Introduction

This document describes the final release of the Integrated Financial Market Information System (IFMIS).

The deliverable is accompanied by the software prototypes that comprise all the necessary modules of the IFMIS that are needed to replicate the FIRST infrastructure. Those modules are: pipeline components, pipeline integration component, storage services and visualisation components. The FIRST GUI (called “Sentify Portal” or IFMIS GUI) is thoroughly described in (FIRST D7.5 Final Version of Integrated Financial Market Information System GUI, 2013) that is released in parallel to this document. Therefore IFMIS GUI and will be mentioned here only in the context of presenting the big picture and deployment details. The GUI is heavily based on the underlying infrastructure and provides an end-user access point for showcasing the system features. For details on the “Sentify Portal” please refer to (FIRST D7.5 Final Version of Integrated Financial Market Information System GUI, 2013).

The overall integration effort is based on the following preceding deliverables: (i) (FIRST D2.2 Conceptual and technical integrated architecture design, 2011), where the architecture and technical design was presented, (ii) (FIRST D7.2 Early prototype of Integrated Financial Market Information System, 2012) which shows the early prototype of pipeline implementation and integration of components and (iii) (FIRST D7.1 Integration Infrastructure Release, 2012) which describes the pipeline integration component.

Besides covering the final description of the created infrastructure, this release provides also the technical instructions for acquiring, building and deploying the FIRST infrastructure.

In the following chapter we present the overview of the components comprising the final version of IFMIS and later we describe necessary steps for FIRST deployment.

Please note that for the components that already have deployment details provided, we only repeat most important parts, while for further details, we refer to the appropriate documentation or deliverable. For this reason, this deliverable partly recaps the information presented already in some of the M33 deliverables. Some components have also an on-line documentation, which is always most up-to-date.

On the other hand, WP5 and WP4 deployment details have been extended due to the fact that either they were not yet covered (WP5) or that their components further evolved during the last reporting period (WP4) and were significantly updated.

2. The map of FIRST component infrastructure

This chapter provides an overview of the software components that constitute the FIRST Integrated Financial Market Information System as of M36. The produced system is based on the next iteration of the “Intermediate Prototype” system developed in M18 and reported in D7.2 (FIRST D7.2 Early prototype of Integrated Financial Market Information System, 2012).

This document aims at describing the IFMIS infrastructure (from the deployment point of view) by providing guidelines for understanding dependencies and component-oriented view.

The goal of WP7 is to ensure integration of technical components that were created as the data processing pipeline (see D2.2, D7.2 for details).Besides this goal WP7 produces no additional components for data processing. Instead, on the one hand, it provides means for joining the components together (by using the pipeline integration component, see D7.1 for details), and on the other hand, it supervises that the technical components produce satisfactory data for IFMIS GUI and Use Cases (see D7.3).

2.1. Overall picture

The overall conceptual FIRST system architecture has been presented in (FIRST D2.2 Conceptual and technical integrated architecture design, 2011) and first prototype experiments explained in (FIRST D7.2 Early prototype of Integrated Financial Market Information System, 2012).

Conceptually, FIRST system architecture can be decomposed into the following parts:

- Backend Services and integrated technical components constitute the core of the FIRST technical work. They consist of the following building blocks: (i) pipeline components (WP3, WP4, WP6) provide the core technical means for processing a document stream (ii) pipeline integration component (WP7) provides lightweight infrastructure for components communication (FIRST D7.1 Integration Infrastructure Release, 2012), (iii) Information Integration (WP5) provides storage services for pipeline operations (FIRST D5.1 Specification of the information-integration model, 2011)
- Use Cases and GUIs: consist of two separate parts: (i) implementation of GUIs for the IFMIS that is a common access point for showcasing FIRST functionalities, (ii) implementation of the three FIRST Use Cases.
  GUI development is a part of the work within WP7 resulting in additional deliverables (FIRST D7.3 Early prototype of Integrated Financial Market Information System GUI, 2012) and (FIRST D7.5 Final Version of Integrated Financial Market Information System GUI, 2013). While Use Case development is a subject of WP8 work, WP7 is ensuring the definition and provisioning of necessary services for facilitating Use Case implementation on top of the FIRST infrastructure.

Figure 1 presents the system overview that constitutes the IFMIS infrastructure. The integration between WP3 data acquisition and WP4 sentiment extraction is based on WP7 messaging component. Both WP3 and WP4 store data in WP5 storage for later retrieval. WP6 provide means for visualising news and blog posts data streams and WP5 offers sentiment data access services for the FIRST GUI. The end user can access both GUI and visualisations depending on her goal.

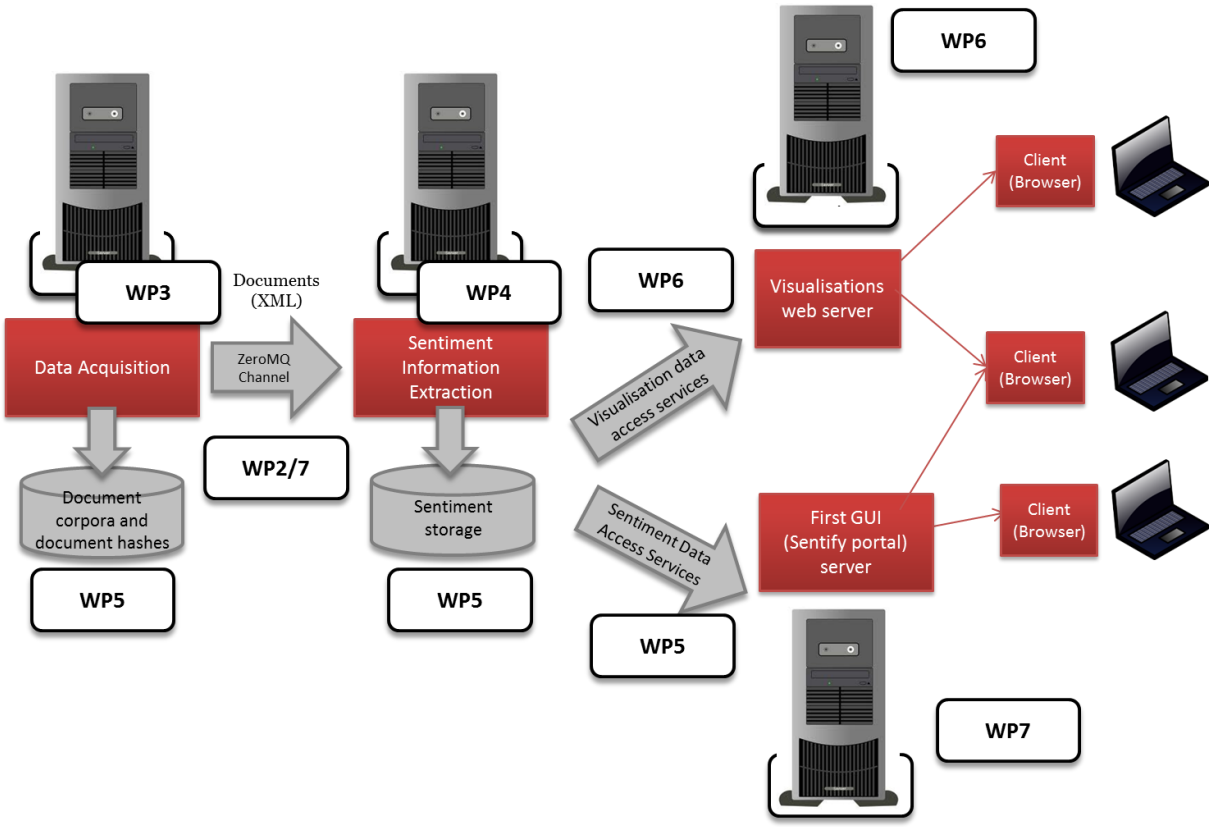

Figure 1: Overview of the components deployment and dependencies

2.2. Components not being part of IFMIS

The principal idea of the IFMIS infrastructure is based on developing open source components which enable deployment in an individual setup. The FIRST project maintains the "production" version of the FIRST infrastructure (hosted on `first.ijs.si` machine, with FIRST GUI available under: http://sentify.project-first.eu/), but it is possible to reproduce the whole system on the local machine in order to run one's own instance of sentiment analysis pipeline.

However, this scenario does not apply to the Use Case prototypes. As Use Cases of the FIRST project are using some very sensitive data, those prototypes have been defined as confidential. For the same reason, WP6 decision support components are also excluded from the IFMIS open source components list. Although some of the developed libraries such as JDEXi2[1] are available as open source, the developed DEXI models are Use Case-specific (see (FIRST D6.3 Highly scalable machine learning and qualitative models v2, 2013) for details), thus not feasible for hosting in a public repository.

In the middle ground there are components targeting the twitter as an alternative data source. While they were not envisaged at the project inception, some twitter analysis components were incorporated into the FIRST infrastructure. As a result, these components are provided to be downloaded and deployed locally, but only as a compiled executable (see (FIRST D6.5 Highly scalable interactive visualization of textual streams, 2013) for more). They are therefore listed as parts of IFMIS infrastructure due to the fact that the IFMIS GUI offers some also some of the twitter data for comparison reasons. However twitter components are not first class citizens of the FIRST project.

[1] http://kt.ijs.si/MarkoBohanec/jdexi.html

3. Deployment of the IFMIS infrastructure

This chapter provides the list of all components that form the IFMIS infrastructure. As mentioned in the Introduction, most of them have been thoroughly described in previous or complementary deliverables. Nevertheless, the list provided in this chapter provides information about all components in a clean and systematic form of a complete list of all pieces that form the IFMIS infrastructure. The following sections are structured in a workpackage-oriented way.

For the reader convenience, we mention every component, even though some of them were already described in their corresponding deliverables. In such case, a shorten recap is provided, and reader is pointed to the extended description in the corresponding document.

3.1. General environment requirements

The IFMIS infrastructure has been tested and run under the Windows operating system. In the production setup it is run under Windows Server 2008 R2 Enterprise, and is assumed as the bottom-line OS version. The DB engine used for relational data storage is SQL Server 2008 R2 Service Pack 1 (version 10.50.2500). Java version used by WP4 and WP7 is JDK 6u45. .NET framework for WP3/WP6 components is v4.0.30319.

Please note that the pipeline components that are integrated using the pipeline integration component described in (FIRST D7.1 Integration Infrastructure Release, 2012) can be distributed on multiple machines, with different environment settings, depending on individual components' needs.

The pipeline processing components need enough processing power in order to keep up with the on-line document stream from defined list of news sources (see WP1 deliverables for sources list). The IFMIS infrastructure is currently run on machine with 24 cores (at 2.3 GHz) and 256 GB of RAM. The processing power used by distinct components can vary but for each component shouldn't be lower than 2 full cores and ~24GB RAM (e.g. see (FIRST D3.3 Large-scale ontology reuse and evolution, 2013) chapter 4.2.1 for details).

3.2. Data Acquisition (WP3)

This section gathers deployment details on data acquisition pipeline components developed within WP3. It consists of the following data sources: (i) news and blog posts and (ii) Twitter data.

3.2.1 Dacq (Data Acquisition Pipeline)

The complete Data Acquisition Pipeline (Dacq) prototype deployment is described in (FIRST D3.3 Large-scale ontology reuse and evolution, 2013), section 4.1.

The DacqPipe source code is now available under the LGPL1 open-source license. It is hosted on GitHub at https://github.com/project-first.

The following projects are required to compile DacqPipe:

- DacqPipe, git://github.com/project-first/dacqpipe.git
- LATINO, git://github.com/project-first/latino.git
- LATINO Workflows, git://github.com/project-first/latinoworkflows.git

- SharpNLP, git://github.com/project-first/sharpnlp.git
  Original project Web site: http://sharpnlp.codeplex.com
- SemWeb.NET, git://github.com/project-first/semweb.git
  Original project Web site: http://razor.occams.info/code/semweb

The following steps clone the required repositories and create the directory structure required to compile the solution:

1. Create a root directory, e.g., C:\FIRST.
2. From within the root directory, execute the following commands:

```
C:\FIRST> git clone –b FIRST_final git://github.com/project-first/dacqpipe.git DacqPipe

C:\FIRST>   git   clone   –b   FIRST_final   git://github.com/project-first/latino.git
Latino/Latino

C:\FIRST> git clone –b FIRST_final git://github.com/project-first/latinoworkflows.git
Latino/LatinoWorkflows

C:\FIRST> git clone git://github.com/project-first/sharpnlp.git SharpNLP

C:\FIRST> git clone git://github.com/project-first/semweb.git SemWeb
```

3. Open DacqPipe\DacqPipe.sln in Visual Studio (ver. 2008 or later, Visual Studio Express versions are freely downloadable from http://www.microsoft.com/visualstudio/eng/products/visual-studio-express-products). You should be able to compile and run the code.

The compiled version of the software can be obtained from http://first.ijs.si/software/DacqPipeJun2013.zip.

For detailed instructions on configuration, technical details and deplyment, please refer to (FIRST D3.3 Large-scale ontology reuse and evolution, 2013), chapter 4.1.

3.2.2 TwitterDacq (Twitter Data Acquisition Pipeline)

The complete Twitter Data Acquisition Pipeline (TwitterDacq) prototype deployment is described in (FIRST D3.3 Large-scale ontology reuse and evolution, 2013), section 4.2. As mentioned in section 2.2, this component is not published as open source and therefore not part of FIRST development. However, the compiled executable is available for deployment. It can be downloaded from: http://first.ijs.si/software/TwitterDacqJun2013.zip.

The chapter 4.2.1 of deliverable D3.3 contain all necessary details on downloading, deploying and configuring this component.

3.2.3 FIRST Ontology

FIRST ontology for detecting named entities has been described in (FIRST D3.3 Large-scale ontology reuse and evolution, 2013), section 2. All files are available for download at: http://first.ijs.si/FIRSTOntology/. The ontology consists of 3 versions, representing the incremental growth of the number of concepts, with distinct release for each year of the project (Y1, Y2 and Y3).

For the sentiment extraction ontology, used within WP4 sentiment extraction components please refer to section 3.3.2.1 of this deliverable.

3.3. Semantic Information Extraction (WP4)

In the FIRST project the conceptual work of (Klein, Altuntas, Kessler, & Häusser, 2011) has been implemented, extended and optimized leading to the WP4 Semantic Information Extraction Components described in (FIRST D4.1 First semantic information extraction prototype, 2011), (FIRST D4.2 Semantic information extraction components, addressing noise and uncertainty, 2012) and FIRST D4.2. The latter presents the steps that have been accomplished to optimize the WP4 components to noisy and uncertain data. In addition, deliverable (FIRST D4.3 Large-scale semantic information extraction components, 2013) explains how the WP4 components have been scaled to large scale.

The WP4 Semantic Information Extraction Components are part of the FIRST pipeline. Generally, the components can be viewed as a system that gets a document in GATE-XML format – as input – and returns – as output – semantic information about the document. In the FIRST project the input is provided by the WP3 DACQ-Pipeline (see (FIRST D3.1 Semantic resources and data acqusition, 2011)) and the output is stored in a database as defined by WP5 (see (FIRST D5.3 Large-scale integrated knowledge base, 2012) with resubmission in 2013).

The WP4 Semantic Information Extraction Components are released under the GNU Lesser General Public License (LGPL-License)[1]. The source code is available at the public repository at: `git://github.com/project-first/semanticinformationextraction.git`.

In section 3.3.1 we first describe the directory and file structure followed by the instructions for component deployment. Concretely, the following steps are necessary to be able to deploy and run the WP4 Semantic Information Extraction Components:

1. The *required resources* have to be downloaded and stored locally (section 3.3.2).
2. The software has to be *built* with the provided ant script from the given source code. The result will be a jar-file and a lib folder (section 3.3.3).
3. The software has to be *re-configured* according to the deployment (section 3.3.4).
4. Finally the software has to be *deployed* and can be executed (section 3.3.5).

3.3.1 Directory Structure

Table 1 describes the core folders of the WP4 Semantic Information Extraction components. The location is the path to the folder that is described in the description. If the location has no filename extension it is of type folder. Otherwise it is of the type that is defined in the filename extension. Therefore, `hybrid.cfg.xml` is a XML file, `run.bat` and `updatePipline.bat` are batch files and the remaining rows are folders.

Table 1: Source code structure of WP4 Semantic Information Extraction component

Path and File Structure	Description
`antScripts`	This folder contains the **ant script** that is required to build the software.
`database_schema`	This folder contains the **database schema**. The database schema is called `db_schema.sql`. It is required to create a database with the correct tables.

[1] https://www.gnu.org/licenses/lgpl.html

`hibernate.cfg.xml`	This is the configuration file for hibernate. This file configures in which database the results should be stored.
`hibernateConf`	Object/relational mapping xml files used by hibernate. This includes the definition how to map the defined classes to the corresponding database tables.
`lib`	The lib folder contains external **JAR libraries**. In the downloadable source code this folder will be empty. You have to add the required files manually into this folder. This will be described in a later section.
`logs`	This folder contains the logs that are returned by the pipeline.
`resources`	The resources folder contains the **resources** that are required for the source code to be executed properly. The subfolders are described in the following rows.
`resources/config`	The configuration folder contains the **WP4 pipeline configuration** and the configuration for **ZeroMQ configuration**, which has been described in the previous section. The config folder allows you to configure the pipeline according to your requirements.
`resources/JSI+UHOH`	The JSI+UHOH folder contains subfolders for the rules, the ontology and the machine learning classifiers. The subfolders are described in the following rows.
`resources/JSI+UHOH /application`	The application folder contains the **machine learning classifiers**. In the HybridFuzzy subfolder the two machine learning classifiers that are required for fuzzy classification are stored. The classifiers are called “HybridFuzzyNeg” and “HybridFuzzyPos”. The classifier “HybridFuzzyPos” refers to the positive classifier which distinguishes between the five positive fuzzy labels, see (FIRST D4.2 Semantic information extraction components, addressing noise and uncertainty, 2012). The classifier “HybridFuzzyNeg” refers to the negative classifier and distinguishes between the respective negative 5 negative labels. The structure of the negative and the positive classifiers are both the same. Therefore, only the positive classifier will be described. In the subfolder of HybridFuzzyPos is a subfolder called hybrid_TOxSS. This subfolder contains (i) the configuration of this specific classifier `hybrid_TOxSS.xml` (ii) the trained classifier in the folder `savedFiles`. The configuration and the classifier are both explained in the GATE manual. Please read the GATE Machine Learning guide[1] for further information.
`resources/JSI+UHOH /Jape`	The knowledge based approach of the Information Extraction System relies on **JAPE rules**. These JAPE rules are utilized to extract sentiment regarding sentiment objects. How the JAPE rules are used in the Semantic Information Extraction System is described in (Klein, Altuntas, Kessler, & Häusser, 2011) and the deliverables (FIRST D4.1 First semantic information extraction prototype, 2011) and (FIRST D4.2 Semantic information extraction components, addressing noise and uncertainty, 2012).
`resources/JSI+UHOH /ontology`	The **FIRST Ontology** is used to represent the knowledge which is required for the Semantic Information Extraction System to infer sentiment. The file name of the ontology is `FIRSTOntology.owl`.

[1] http://gate.ac.uk/sale/tao/splitch18.html#chap:ml

`run.bat`	Please start the pipeline with this batch file, since it contains additional parameters that are required to execute the pipeline correctly.
`src`	The `src` folder contains the **source code** that is required to build the project. The code is written in java and can be edited, e.g. with eclipse.
`updatePipline.bat`	This `bat` file is used to update the resources of the pipeline. It allows updating of the machine learning classifiers, the ontology, and the JAPE rules.

3.3.2 Setup and Download of the Required Resources

There are several external resources that are required to use the WP4 Semantic Information Extraction Components. In the subsequent section the different resources will be described. Additionally, this section will provide the download links to the external resources that are required to build all components.

The source code is provided in the GitHub repository `git://github.com/project-first/semanticinformationextraction.git`. Please download the source code and save it in a folder with a path without whitespaces, where you can make additional changes to the source code and configurations.

3.3.2.1 Information Extraction Ontology

The WP4 Semantic Information Extraction Components require orientation terms in the `FIRSTOntology.owl` (Klein, Altuntas, Kessler, & Häusser, 2011). In the FIRST project the orientation term dictionary “General Inquirer” has been utilized for this purpose.

The following section describes how the General Inquirer can be added in order to use the WP4 components.

The General Inquirer can be downloaded from http://www.wjh.harvard.edu/~inquirer/. The orientation terms of the General Inquirer have to be added into the ontology into following classes:

1. Add all *negative adjectives* from General Inquirer as individuals to http://project-first.eu/FIRSTOntology_OrientationTerm.owl#General_Inquirer_NegativeAdjective class.
2. Add all *negative nouns* from General Inquirer into this class as individuals to http://project-first.eu/FIRSTOntology_OrientationTerm.owl#General_Inquirer_NegativeSubstantive class.
3. Add all *negative verbs* from General Inquirer as individuals to http://project-first.eu/FIRSTOntology_OrientationTerm.owl#General_Inquirer_NegativeVerb class.
4. Add all *positive adjectives* from General Inquirer as individuals to http://project-first.eu/FIRSTOntology_OrientationTerm.owl#General_Inquirer_PositiveAdjective class.
5. Add all *positive nouns* from General Inquirer as individuals to http://project-first.eu/FIRSTOntology_OrientationTerm.owl#General_Inquirer_PositiveSubstantive class.
6. Add all *positive verbs* from General Inquirer as individuals to http://project-first.eu/FIRSTOntology_OrientationTerm.owl#General_Inquirer_PositiveVerb class.

To add individuals to the above described classes the tool Protégé (see http://protege.stanford.edu/) can be used, furthermore, the version 3.4.8 is tested to be compatible with the provided files.

It should be noted that the words that are added to the classes are interchangeable. Thus, you can add it to another class if you have a different opinion concerning a word, e.g., you could add a word that is defined by General Inquirer as “positive” to the class “negative” and the WP4 Semantic Information Extraction Components would still work. This allows the user to tailor the semantic extraction behavior according to her purposes.

3.3.2.2 External JAR Libraries

The external JAR libraries which are required need to be manually downloaded. Please download all JAR libraries that are given in the following table. Please use exactly the listed JAR files and do not change the file names. This is required to execute the ant script.

Please download the GATE 6.1 Build 3913 from http://sourceforge.net/projects/gate/files/gate/6.1/gate-6.1-build3913-ALL.zip/download. It is strongly recommended to use this exact build version, otherwise incompatibilities might occur. First, copy the lib-folder from the gate zip file to the lib folder in the downloaded WP4 source code. Second, copy the plugins folder from the gate zip file into the lib folder of the WP4 source code. Third, download the following jar libraries from the given web locations and make sure that the version number and file size are correct.

JAR-Files	Version	Library	Web location	File size
antlr-2.7.6.jar	2.7.6	ANTLR (ANother Tool for Language Recog-nition)	http://www.antlr.org/	433 KB
commons-collections-3.1.jar	3.1	Apache Commons Collections	http://commons.apache.org/proper/commons-collections/	546 KB
commons-collections-3.2.1.jar	3.2.1	Apache Commons Collections	http://commons.apache.org/proper/commons-collections/	561 KB
commons-configuration-1.6.jar	1.6	Apache Commons Configuration	http://commons.apache.org/proper/commons-configuration/	291 KB
commons-io-2.0.1.jar	2.0.1	Apache Commons IO	http://commons.apache.org/proper/commons-io/	155 KB
hibernate3.jar	3.6.6. Final	Hibernate	http://www.hibernate.org/	3.94 MB
hibernate-jpa-2.0-api-1.0.1.Final.jar	1.0.1. Final	Hibernate	http://www.hibernate.org/	100 KB

jargs.jar	1.0.0	JArgs	http://jargs.sourceforge.net/	11.1 KB
javassist-3.12.0.GA.jar	3.12.0.GA	Javassist	http://www.csg.ci.i.u-tokyo.ac.jp/~chiba/javassist/	618 KB
jcifs-1.3.17.jar	1.3.17	Java CIFS	http://jcifs.samba.org/	386 KB
jcl.over.slf4j-1.6.1.jar	1.6.1	Simple Logging Facade for Java	http://www.slf4j.org/	17.2 KB
jeromq-0.3.0-SNAPSHOT.jar	0.3.0	Zero MQ	http://zeromq.org/	218 KB
jta-1.1.jar	1.0.1.GA	Java Transaction API	http://www.oracle.com/technetwork/java/javaee/jta/index.html	10.6 KB
log4j-1.2.16.jar	1.2.16	Apache Log4J	http://logging.apache.org/log4j/1.2/	470 KB
messenger-nj-0.0.3-SNAPSHOT.jar	0.3.0	Zero MQ	http://zeromq.org/	38.8 KB
ojdbc14.jar	10.2.0.5	Oracle10g JDBC Driver	http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html	1.49 MB
opencsv-2.3.jar	2.3	Opencsv	http://opencsv.sourceforge.net/	13.8 KB
slf4j-api-1.6.1.jar	1.6.1	Simple Logging Facade for Java	http://www.slf4j.org/	24.8 KB
slf4j-log4j12-1.5.11.jar	1.5.11	Simple Logging Facade for Java	http://www.slf4j.org/	9.46 KB
sqljdbc4.jar	4.0.0	Microsoft JDBC Driver for SQL Server	http://technet.microsoft.com/en-us/library/ms378749.aspx	570 KB
sqljdbc.jar	4.0.0	Microsoft JDBC Driver for SQL Server	http://technet.microsoft.com/en-us/library/ms378749.aspx	549 KB

Table 2: Required external jar files that are not provided by GATE

3.3.2.3 Database

The WP4 Semantic Information Extraction Components store the extracted information into a database. The database schema has been described in (FIRST D5.3 Large-scale integrated knowledge base, 2012). The script to create the required database schema is provided in the directory `database_schema\db_schema.sql`.

Before running the script, it needs to be provided with a target database. This is accomplished by creating an empty database (with the name of your choice) on your server and subsequently adjusting the first row of the provided script from "`DATABASENAME`" to the name of your created database. The provided script uses MS-SQL syntax which means it can be easily executed on a SQL Server[1] with SQL Management Studio[2]. If the database schema should be used with other RDBMS, the SQL script might to be adapted to the corresponding database syntax.

3.3.2.4 Environment

The WP4 Semantic Information Extraction Components require a specific Java Development Kit (JDK) to work properly. The officially supported version of the JDK is 1.6.0_38. Please use the software with this specific JDK version, since the software was not thoroughly tested on lower or higher versions of the JDK. Not using JDK v1.6.0_38 can potentially lead to serious incompatibilities. Additionally, it is also recommended to use an Eclipse Standard version 4.3 or higher.

3.3.3 Building the Software

Follow these steps to create the WP4 Semantic Information Extraction Components:

1. The external resources have to be added to the lib folder (i.e. `\FIRST_PIPELINE\lib`).
2. General Inquirer has to be added to the Ontology as described above.
3. The ant script has to be executed. The ant script can be found in the folder `antScripts` and is called `build_ZEROMQ_CompleteJSIKnowledgeBasedCrispClassificationReceiver.xml`. This ant script is used to build the software.
4. The database schema has to be executed as described previously.

3.3.4 Configuration

This section describes the configurations that are needed for setting up the WP4 Semantic Information Extraction components.

3.3.4.1 Hibernate Configuration

The Hibernate configuration can be found in the main folder and is called `hibernate_cfg.xml`. For using Hibernate[3] with the database the user data like user name, database name, password and other connection parameters have to be provided in the file `hibernate.cfg.xml`.

3.3.4.2 ZeroMQ Configuration

Another part to configure is the messaging middleware based on ZeroMQ[1]. The ZeroMQ configuration can be found in the folder `\resources\config` and is called `config.xml`.

[1] Microsoft provides an express edition of the SQL Server. See www.microsoft.com/express/sql/
[2] Microsoft provides an express edition of the Management Studio. See http://www.microsoft.com/en-us/download/details.aspx?id=8961
[3] http://www.hibernate.org/

```
<messenger-definition>
	<messaging>
		<ID>1</ID>
		<ReceiverNumber>2</ReceiverNumber>
		<ProducerNumber>4</ProducerNumber>
		<connection>
			<!--Receive from Dacq-Pipeline -->

	<MessageReceiveAddress>tcp://first.ijs.si:5563</MessageReceiveAddress>
			<SendLoadBalancingAddress>tcp://first-
				vm3.ijs.si:5564</SendLoadBalancingAddress>
			<FinishReceive>tcp://first.ijs.si:5560</FinishReceive>

			<RECEIVE_COMMAND_FILTER></RECEIVE_COMMAND_FILTER>
			<QueueName>WP3_QUEUE</QueueName>

			<!-- Enable monitoring capabilities -->
			<MonitoringSocket>tcp://*:4440</MonitoringSocket>
		</connection>

		<!--1 - we use a blocking queue that waits for the receiver
			(when the queue is full) or 0 - might drop messages
			if a data peak makes the queue overflow -->
		<BLOCK_ON_SEND>0</BLOCK_ON_SEND>
		<MessagingType>0</MessagingType>
		<Broker>0</Broker>
		<MAX_QUEUE_SIZE>250</MAX_QUEUE_SIZE>
		<MIN_QUEUE_SIZE>1</MIN_QUEUE_SIZE>
		<BLOCKING_QUEUE>FALSE</BLOCKING_QUEUE>
		<IGNORE_QUEUE_OVERFLOW>TRUE</IGNORE_QUEUE_OVERFLOW>
		<MESSAGE_REQUEST>R</MESSAGE_REQUEST>
		<WAIT_COMMAND>WAIT</WAIT_COMMAND>
		<FINISH_COMMAND>FINISH</FINISH_COMMAND>
		<CONTINUE_COMMAND>CONTINUE</CONTINUE_COMMAND>
		<outFileStorageAddress>outMessageStorage</outFileStorageAddress>
		<inFileStorageAddress>inMessageStorage</inFileStorageAddress>
		<MAX_FILE_STORAGE_SIZE>100000</MAX_FILE_STORAGE_SIZE>

		<logging>
			<DBLogging>false</DBLogging>
			<DBLoggingReceiver>tcp://*:5561</DBLoggingReceiver>
			<DBLoggingLocalWrite>false</DBLoggingLocalWrite>
		</logging>
	</messaging>
</messenger-definition>
```

Listing 1: Example configuration file for ZeroMQ-based messaging component

To receive messages, the file `config.xml` in the `resources` directory has to be edited.

For a discussion on lightweight integration based on ZeroMQ see (FIRST D2.2 Conceptual and technical integrated architecture design, 2011) and (FIRST D7.1 Integration Infrastructure Release, 2012), section 2.3.2 for a description of all parameters. Listing 1 presents an example configuration of the WP4 components, which receives documents from the WP3 Dacq components.

[1] http://zeromq.org/

3.3.4.3 Component Configuration

Next, the WP4 Semantic Information Extraction Components have to be configured. The pipeline configuration can be found in the folder `\resources\config` and has the following name `pipeline_config.xml`. In the `pipeline_config.xml`, the following parameters can be configured.

Name	Description	Default x
Graceful timeout	Graceful timeout is a parameter that allows you to define the timeout in milliseconds before the execution on a document is gracefully stopped. XML-Syntax: `<entry key="gracefulTimeout"> x </entry>`	12000
Max message buffer size	The parameter sets the number of temporarily stored messages for processing in the component. XML-Syntax: `<entry key="maxMessageBufferSize"> x </entry>`	50
Transaction timeout	Sets the time until the transaction of the results for the document will be aborted. XML-Syntax: `<entry key="TransactionTimeout"> x </entry>`	10
Maximum message length	This parameter sets the maximum message length in characters of a document. If the document contains more characters than the specified value the document will be ignored. XML-Syntax: `<entry key="maxMessageLength"> x </entry>`	2000000
Execute fuzzy classification	This parameter controls if the fuzzy classification is executed. If it is set to false only the crisp classification will be executed. XML-Syntax: `<entry key="executeHybridFuzzyClassifier"> x </entry>`	true

File storage configuration	The parameter `serverIP` must point to the serverIP or the fully-qualified server-name, where the target document and other output will be stored. The parameter has to constitute a complete network path where the document and output files will be stored. Please set the `x` to the `serverIP` of your server.
	
	XML-Syntax:
	`<entry key="serverIP"> x </entry>`
	
	The `shareName` has to be associated with the shared folder name. The parameter has to constitute a complete network path where the document and output files will be stored. Please replace `x` with the complete network path.
	
	XML-Syntax:
	`<entry key="shareName"> x </entry>`
	
	Additionally the user name and the password are required. Please replace the `x` with the name of the user that has access to the server and `y` with the password of this user.
	
	XML-Syntax:
		-

	<entry key="user"> x </entry> <entry key="password"> y </entry>	
Write input file	The `writeInputFile` Parameter allows you to write the input file to the File Storage. XML-Syntax: <entry key="writeInputFile"> x </entry>	false
Write output file	The Parameter `writeOutputFile` allows you to write the GATE result file, which contains the annotations to the document. This will contain the results of the Semantic Information Extraction Pipeline. XML-Syntax: <entry key="writeOutputFile"> x </entry>	false
Convert XMLs to zip file	The `zipFile` parameter allows you to define whether or not the input file, output file and the result should be stored as zip files. XML-Syntax: <entry key="zipFile"> x </entry>	true

Table 3: Configuration of the WP4 Semantic Information Extraction Components

3.3.5 Deployment and Running

Finally, copy the whole directory and file structure (see section 3.3.1) including the built jar, resources and libs to a folder and start `run.bat`. The `run.bat` contains also JVM parameters to optimize the WP4 components (such as heap memory limits).

3.4. Data Access Services (WP5)

The most important outcomes of WP5 for the IFMIS infrastructure are: (i) relational data storage and (ii) data access services. On top of relational database, a set of sentiment retrieval services has been deployed in order to facilitate uniform data access for other IFMIS component (FIRST GUI) and Use Cases. The following subsections describe deployment of the Data Access Services.

3.4.1 Database schema

The SQL schema for structured sentiment database is distributed along with the WP4 Semantic Information Extraction components. Refer to the section 3.3.2.3 for more details.

3.4.2 Sentiment Services

Deliverable (FIRST D5.3 Large-scale integrated knowledge base, 2012) describes a set of SOAP services developed in order to provide sentiment data to the different Uses Cases and the FIRST GUI. This section details the process of acquiring source code, build, configuration and deployment of a new instance of these services.

The prerequisites for deploying the services are the following:

- A SQL Server instance with the schema of the sentiment database,
- Java 1.6 or higher,
- Tomcat Application Server 6[1],
- Axis 2 deployed as web application inside Tomcat[2],
- Eclipse JEE IDE with the Axis 2 Service Archive Generator Wizard[3].

The first step is to configure the database used by the service (for description on DB creation, see section 3.4.1) this task is performed modifying the file `src/hibernate.cfg.xml` in the following way:

```
…
<property
name="hibernate.connection.url">jdbc:sqlserver://[DBServer];databaseName=[DBname]
</property>
<property name="hibernate.connection.username">[DB USER]</property>
<property name="hibernate.connection.password">[DB PASSWORD]</property>
…
```

Listing 2: Hibernate configuration

Where:

- [DBServer] – name or address of database server,
- [DBname] – name of database,
- [DB USER] – username with access to the database,
- [DB PASSWORD] – password for given username.

[1] http://tomcat.apache.org/
[2] http://axis.apache.org/axis2/java/core/docs/installationguide.html#servlet_container
[3] http://axis.apache.org/axis2/java/core/tools/eclipse/servicearchiver-plugin.html

Once the database is configured it necessary to proceed to build the services, this task is executed by Eclipse using the Axis2 Service Archive Generator Wizard following the next steps:

1. Open the “File>New” menu in Eclipse IDE and choose the “Other” option.
2. Select Axis 2 Wizards>Axis2 Service archiver.
3. Introduce the location of the .class files (default location: `[Workspace location]/StorageService/bin`) and be sure that the “include .class files only” is not selected.
4. Select the “Skip WSDL” option.
5. Add one by one all the jar files contained in the lib folder.
6. Select the “generate service xml automatically” option.
7. Put `SentimentService` in the “Service name” text field and `eu.projectfirst.wp5.services.SentimentService` in the “Class name” text field. Also, check the “Search declared methods only” and select all the methods in the list.
8. Put the name of the service file (.aar) and the destination folder and finally, put the finish button.

Once you have generated the service file, it is necessary to copy this file to the `[Tomcat HOME]/webapps/axis2/WEB-INF/service` folder. In that moment the services are deployed, after that, it is possible to check if the services are deployed accessing to `http://[Tomcat-server]:8080/axis2/services/listServices`.

If everything is correctly configured as described above, a list of available services should be displayed (see Figure 2).

Additionally, you can use a SOAP client (e.g. SOAP UI[1]) to call any of the service operations (e.g. `getDocumentCount`). A proper SOAP response must be obtained with the result. In the case of any errors, it is necessary to repeat the provided step by step instructions and check in particular if the database credentials are correct.

[1] http://www.soapui.org/

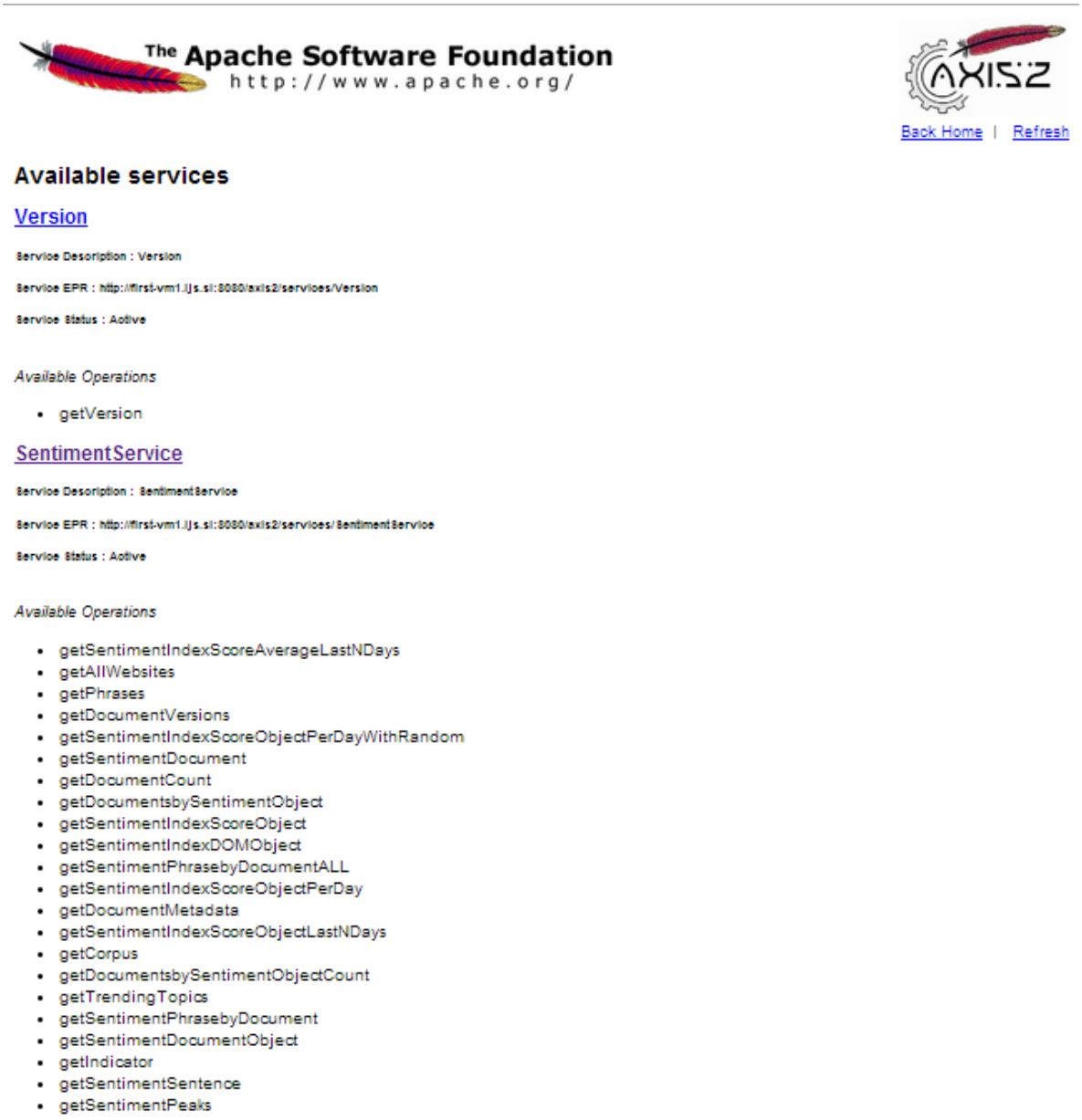

Figure 2: Axis2 list of services

3.4.3 Ontology update service[1]

The Resources Update Service allows end-users of the FIRST system to update the FIRST information extraction ontology (see FIRST D4.1), information extraction rules (see FIRST D4.1 and D4.2), the machine-learning sentiment classifier that is part of the hybrid classifier (see FIRST D4.2), and also WP4 sentiment analysis pipeline configurations (see this deliverable). All these pieces are considered “resources” in WP4 and reside in a resource folder for each deployment of the WP4 pipeline.

By means of the Resources Update Service, end-users can make updates to reputation indicators and reputation topics in the FIRST information extraction ontology. In addition to this,

[1] The update service was requested by the reviewers in the second project review. Reviewers’ statement was the following: “There should also be an explicit consideration of updating the classifiers in the data stream analysis in a delivered FIRST service, as typically many reputation related terms and phrases change over time”.

the Resource Update Service allows to update also any other part of the resources folder that is used by WP4 software components.

The Resources Update Service consists of two parts: (1) a web application, and (2) a batch script. Both parts have been deployed in the FIRST infrastructure. See section 3.3.1 of (FIRST D7.4 Final version of Integrated Financial Market Information System, 2013) for the detailed description.

3.4.3.1 Web application for ontology resources

The web application is the part to be used by end-users. The web application allows uploading a zip file of the resources folder to the FIRST infrastructure and deploying it to all running processes. The zip file does not necessarily need to contain all files of the resource folder and its subfolders. For instance, one could include only the ontology OWL file. In any case, the folders and filenames that are contained in the zip file need to resemble exactly the ones from the deployment.

The upload service is located under: http://sentify.project-first.eu/update. It is a simple upload form that allows selecting file from a local computer and uploads its content to the server. The front-end service is shown in the Figure 3. Once the web application uploads the resources zip file to a web server it will call the subsequently described batch script with a reference to the zip file to deploy the uploaded new resources.

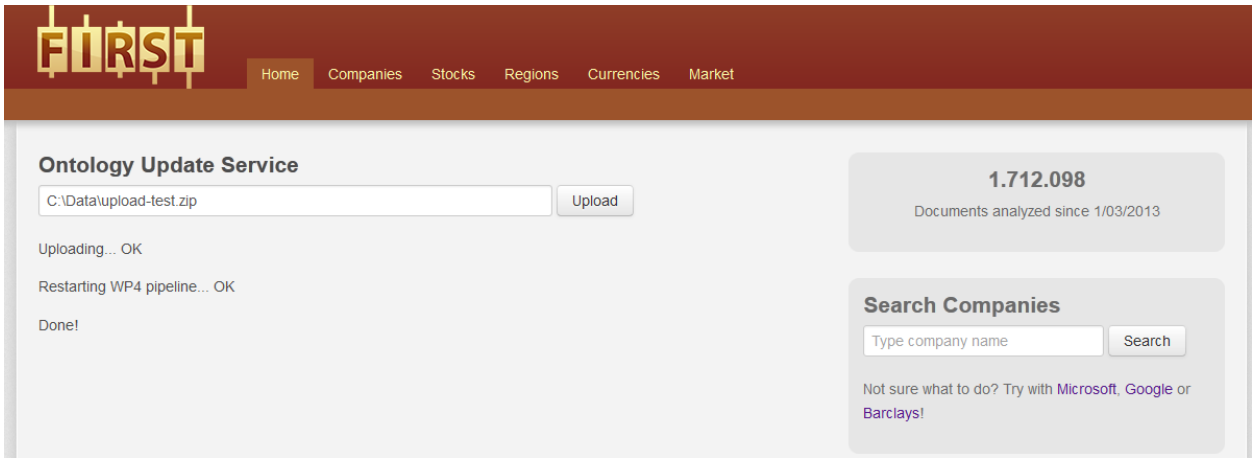

Figure 3: Manual update of FIRST ontology.

3.4.3.2 Batch Script

The batch script for ontology update is the back-end worker for performing all the ontology update tasks. The purpose of the batch script is to:

- Stop all running processes that are deployed and running in the FIRST infrastructure,
- Backup the current resources folder in each of the deployed processes,
- Unzip the new resource folder zip file to each of the deployments,
- Re-start each of the processes.

The batch script named `updatePipeline.bat` assumes that each of the processes is deployed under a common root folder in a folder that starts with a common name on the same machine. For instance, the folder `F:\UHOH\` could be the root folder with subfolders `Pipeline1` and

`Pipeline2` (with “Pipeline” being the common name). Further, the batch script assumes a 7-zip installation.

To make the batch script operational, the variable `pipelineDir` has to be set to the root folder + the common name + “*”. Considering the example above, `pipelineDir` would be `pipelineDir=”F:\UHOH\Pipeline*”`. Next, the variable `zipInstallationDir` has to be set to the 7-zip installation, for instance: `zipInstallationDir="C:\Program Files\7-Zip"`.

The backup zips of a resource folder will be stored in the root folder of each deployment of the pipeline with the current date included in the filename.

3.5. Visualisation Services (WP6)

WP6 Visualisation services provide a set of datasource interfaces for GUI widgets for displaying and graphing sentiment data streams. The technical design and functionalities has been presented in M33 deliverable (FIRST D6.5 Highly scalable interactive visualization of textual streams, 2013). The following subsections only recap most important details on how to obtain the software and point reader to further, more detailed descriptions.

3.5.1 News and blogs visualisations

The news and blogs visualisation services comprise pipeline components, storage database, pipeline monitoring component and data access layer. Full descriptions of the infrastructure and deployment details are present in (FIRST D6.5 Highly scalable interactive visualization of textual streams, 2013), section 2. This suite of components depends on the data acquisition pipeline, described in section 3.2.1. On top of that, the following components are to be deployed:

- MonitorPipeline – an pipeline extension for news and blogs documents preprocessing
  available at: git://github.com/project-first/monitopipeline.git
- NewsMonitorDAL – data access services for providing visualisation data.
  available at: git://github.com/project-first/newsmonitordal.git

For complete instructions, dependencies list, database creation scripts and configuration details please refer to (FIRST D6.5 Highly scalable interactive visualization of textual streams, 2013). There is also on-line documentation for news and blogs sentiments API at http://first.ijs.si/VisAPI/.

3.5.2 Twitter visualisations

Twitter visualisation services provide insight into the twitter sentiment data stream. The infrastructure for twitter visualisations resembles that of news and blog posts. Most noticeably, it requires an operating twitter data acquisition pipeline and provides DAL services on top of it:

- TwitterMonitorPump – twitter data stream preprocessing
  available at: git://github.com/project-first/TwitterMonitorPump.git
- TwitterMonitorDAL – data access layer for twotter visualisation data
  available at: git://github.com/project-first/twittermonitordal.git

For complete instructions, dependencies list, database creation scripts and configuration details please refer to (FIRST D6.5 Highly scalable interactive visualization of textual streams, 2013), chapter 3. There is also on-line documentation for twitter sentiment API at http://first.ijs.si/VisAPI/.

3.5.3 Visualisation Demos

Visualisation demos provide a set of HTML pages, with graphical widgets designed to showcase WP6 visualisation services' capacities. They can be also used as tools for analysing sentiment data from different perspectives and angles. Visualisation demos are using visualisation APIs as a data source for creating interactive graphs on the HTML pages. For more details please refer to (FIRST D6.5 Highly scalable interactive visualization of textual streams, 2013), section 4.4.

Visualisation examples can be browsed online at: http://first.ijs.si/VisAPI/indexVis.html, or downloaded from http://first.ijs.si/software/VisApiDemosJul2013.zip and executed locally.

3.6. Integrated infrastructure and FIRST GUI (WP7)

WP7 provides the “glue” for integrating other technical components info the common IFMIS infrastructure, spanning across workpackages 3, 4, 5 and 6. Also, apart from integration tasks, it provides a common GUI for showcasing integrated infrastructure. The work towards the final release of software components has been described from various perspectives: technical design and architecture: (FIRST D2.2 Conceptual and technical integrated architecture design, 2011) and (FIRST D2.3 Scaling Strategy, 2011), IFMIS Integration: (FIRST D7.2 Early prototype of Integrated Financial Market Information System, 2012) and IFMIS GUI: (FIRST D7.3 Early prototype of Integrated Financial Market Information System GUI, 2012), (FIRST D7.5 Final Version of Integrated Financial Market Information System GUI, 2013).

This deliverable concludes that work and aims at releasing all software components comprising IFMIS. While WP7 itself does not directly process data, it has very important role of bringing all technical components into the whole integrated infrastructure and ensuring the it works in a interoperable way together.

The “visible” proof of the underlying integrated infrastructure is the IFMIS GUI, also developed within WP7 and released as a part of this deliverable. More information is included in the (FIRST D7.5 Final Version of Integrated Financial Market Information System GUI, 2013), and recapped information is also presented in the following subsections.

3.6.1 Pipeline integration component

The pipeline integration component (sometimes referd to as “Streamer” or “ZeroMQ streamer”) has been described already in deliverables (FIRST D2.2 Conceptual and technical integrated architecture design, 2011) and released with deliverable (FIRST D7.1 Integration Infrastructure Release, 2012). Since then some fragments of the codebase has been modified (bug fixing and improvements) but largely it provides the same functionality. Most important modifications are:

- Transform the project into the maven archetype, enabling automatic management of dependencies and substantially simplifying the task of building the software.
- Switching from binary ZeroMQ dependency (previously the JNI Java binding) to fully native java ZeroMQ implementation, called JeroMQ. The ZeroMQ project can be obtained from https://github.com/zeromq/jeromq. For the building purposes it is fetched from the maven repository.
- Inclusion of “blocking mode” (`<BLOCK_ON_SEND>` configuration parameter) – experimental mode for lossless streaming of documents, for instance for the purpose of streaming batches of offline documents. In thismode, the sending component will always wait for receiver to avoid document queue overflows.
- Simple (optional) monitoring capability for counting received and sent documents and detecting queue overflows. The monitoring capability is configured using the `MonitoringSocket` XML node in the configuration file (see Listing 1 for sample streamer configuration file), for example: `<MonitoringSocket>tcp://*:4440</MonitoringSocket>` will instruct the components to provide monitoring statistics on port 4440.
- The .NET version of the streaming component has not been modified.

The code of the component can be retrieved from the FIRST public repository:

- The Java version: git://github.com/project-first/streamer-java.git
- The .Net version: git://github.com/project-first/streamer-net.git

To download sources and build the project, execute commands presented in the Listing 3

```
C:\FIRST> git clone git://github.com/project-first/streamer-java.git
ifmis-workspace
C:\FIRST> cd ifmis-workspace\streamer-java
C:\FIRST> mvn install
```

Listing 3: Downloading and building messaging component

After successful build, the compiled JAR file is located in `streamer-java\target` folder. Please note that you will find 2 versions: one JAR with dependencies (`messenger-nj-0.0.3-SNAPSHOT-jar-with-dependencies.jar`) and other without (`messenger-nj-0.0.3-SNAPSHOT.jar`). It is advised to use the latter (without dependencies).

As this component is typically included as a library, it needs to be invoked using its own API. The documentation of streamer API is already described in (FIRST D7.1 Integration Infrastructure Release, 2012).

3.6.2 Simple pipeline monitor

The pipeline monitor is a simple web application that uses the monitoring capability of the Pipeline Integration component. It simply connects to the streamer components (to the monitoring ports defined in the configuration files of each streaming component) and performs monitoring queries in fixed intervals. Such queries read the statistics on messages received and transmitted by the pipeline integration components and also on messages dropped during the process (e.g.: message queue overflow caused by too slow processing of messages).

The configuration file is located in `messenger-monitor-webapp\src\main\resources\config.xml` file. The structure of the file is presented in the Listing 4.

```
<?xml version="1.0" encoding="UTF-8" ?>
<pipelinemonitor>
	<debug>false</debug>
	<interval>60000</interval>
	<component>
		<name>Preprocessing+Classification-1</name>
		<address>tcp://95.87.154.97:4460</address>
	</component>
	<component>
		<name>Preprocessing+Classification-2</name>
		<address>tcp://95.87.154.97:4470</address>
	</component>
	<component>
		<name>Preprocessing+Classification-3</name>
		<address>tcp://95.87.154.97:4480</address>
	</component>
	<component>
		<name>Preprocessing+Classification-4</name>
		<address>tcp://95.87.154.97:4490</address>
	</component>
</pipelinemonitor>
```

Listing 4: Pipeline monitor configuration

Pipeline monitor can monitor simultaneously multiple components. Each component endpoint is specified as a `<component>` node in the configuration file:

- `<name>` name of the component (used only for a GUI purpose)

- `<address>` IP address of the monitoring socket of the streaming component (the same as defined in the streamer configuration file), e.g.:
  `<address>tcp://95.87.154.97:4490</address>`

There are also two global configuration options:

- `<debug>` turn on debug messages (for displaying all monitoring queries). Note that those messages will be logged according to web application server logging configuration. This field accepts values of “true” or “false”.
- `<interval>` interval between monitoring queries (in milliseconds).

The pipeline monitor is a maven application, so no further dependency management should be needed and the overall build instructions are simplified. To download sources and build the project, execute commands presented in the Listing 3.

```
C:\FIRST> git clone git://github.com/project-first/pipeline-monitor-webapp.git ifmis-workspace
C:\FIRST> cd ifmis-workspace/pipeline-monitor-webapp
C:\FIRST> mvn install
```

Listing 5: Downloading and building messaging component

After successful build, the compiled web application will be found in `pipeline-monitor-webapp\target` directory. The deployable file is `messenger-monitor-webapp.war`.This file can be directly deployed in the web application server (e.g. Tomcat or Jetty) in a `/webapps` directory. After successful deployment, the web application should be available under the following URL: http://127.0.0.1/monitor.

3.6.3 Sentify Portal

The code and deployment has been already explained in (FIRST D7.5 Final Version of Integrated Financial Market Information System GUI, 2013), chapter 3. For readers convieniece we recap the instruction for obtaining and building the code.

The source code repository for the Sentify portal is located at git://github.com/project-first/sentify-portal.git. The software has been developed as a multimodule project. To download sources and build the project, execute commands presented in the Listing 6.

The parent module is named `sentify-company`. Executing build within this module will subsequently build all submodules and produce deployable WAR[1] file.

```
C:\FIRST> git clone git://github.com/project-first/sentify-portal.git ifmis-workspace
C:\FIRST> cd sentify-company
C:\FIRST> mvn install
```

Listing 6: Commands for executing maven build against the Sentify codebase

After compiling all submodules, the resulting deployable will be placed in `ifmis-workspace/sentify-company-page/target/sentify-company-page-0.0.1-SNAPSHOT.war`. This is the file that can be directly deployed in the web application server either by

[1] WAR stands for Web application ARchive.

placing in the `/webapps` folder of the webapp server (Jetty, Tomcat) or using Application Manager (Tomcat).

3.6.4 Sentify aggregation jobs

The “Sentify Portal” depends on some data that is computed off-line for performance reasons. This is due to the fact that calculating multiple aggregated values for many sentiment objects results in significant delays that are intolerable in the scope of interactive web applications. To avoid this, some data is calculated offline and cached for the performance reasons. Retrieving cached aggregation data is therefore much faster and less service invocations is made. More information on aggregation jobs is presented in (FIRST D7.5 Final Version of Integrated Financial Market Information System GUI, 2013) in section 3.5.

The source code for aggregation jobs has been integrated into the Sentify portal codebase and it does not have to be compiled separately. The mechanism for running scheduled jobs is based on the Java TimerTask[1] and has been integrated into the portal and is initiated at the portal startup. The cached data of aggregated sentiments is stored in an embedded storage (H2 database[2]) that is shared between portal and the job scheduler.

The access to the cached data is invisible from the end-user point of view, and in case of failure, a failback method is provided (data is read directly from the services). The only visible effect is much faster loading time for GUI widgets showing aggregated data (e.g.: charts).

[1] http://docs.oracle.com/javase/1.5.0/docs/api/java/util/TimerTask.html
[2] http://www.h2database.com/html/main.html

4. Conclusion

This document presents the technical guide for retrieving, configuring and deploying the final version of the Integrated Financial Market Information System, the main result of the FIRST project. While the previous WP7 deliverables focused more on describing the technical design, goals and implementation details, this report is aimed at recapping the overall architecture of IFMIS and providing the final guideline on obtaining the source codes, building the components and deploying each of the IFMIS parts.

This report also concludes the implementation and integration efforts of WP7. As a result providing a solid set of integrated components working together to produce expected financial sentiment data results.

References

FIRST D2.2 Conceptual and technical integrated architecture design. (2011).
FIRST D2.3 Scaling Strategy. (2011).
FIRST D3.1 Semantic resources and data acqusition. (2011).
FIRST D4.1 First semantic information extraction prototype. (2011).
FIRST D4.1 First semantic information extraction prototype. (2011).
FIRST D5.1 Specification of the information-integration model. (2011).
FIRST D4.2 Semantic information extraction components, addressing noise and uncertainty. (2012).
FIRST D4.2 Semantic information extraction components, addressing noise and uncertainty. (2012).
FIRST D5.3 Large-scale integrated knowledge base. (2012).
FIRST D5.3 Large-scale integrated knowledge base. (2012).
FIRST D5.3 Large-scale integrated knowledgebase. (2012).
FIRST D6.4 Interactive visualisation of textual streams v1. (2012).
FIRST D7.1 Integration Infrastructure Release. (2012).
FIRST D7.2 Early prototype of Integrated Financial Market Information System. (2012).
FIRST D7.2 Early prototype of Integrated Financial Market Information System. (2012).
FIRST D7.3 Early prototype of Integrated Financial Market Information System GUI. (2012).
FIRST D7.3 Early prototype of Integrated Financial Market Information System GUI. (2012).
FIRST D3.3 Large-scale ontology reuse and evolution. (2013).
FIRST D4.3 Large-scale semantic information extraction components. (2013).
FIRST D4.3 Large-scale semantic information extraction components. (2013).
FIRST D6.3 Highly scalable machine learning and qualitative models v2. (2013).
FIRST D6.5 Highly scalable interactive visualization of textual streams. (2013).
FIRST D6.5 Highly scalable interactive visualization of textual streams. (2013).
FIRST D7.4 Final version of Integrated Financial Market Information System. (2013).
FIRST D7.4 Final version of Integrated Financial Market Information System. (2013).
FIRST D7.5 Final Version of Integrated Financial Market Information System GUI. (2013).
Klein, A., Altuntas, O., Kessler, W., & Häusser, T. (2011). Extracting Investor Sentiment from Weblog Texts. *Proceedings of the 13th IEEE Conference on Commerce and Enterprise Computing (CEC). Luxembourg.*, 1-9.

