Developer Guide

Development & Installation on a system

Sweden Connectivity AB, S2 Grupo,
University of Granada, HiTec,
University of Geneva

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement No. 318508.

Contents

Chapter 1

Introduction

This guide describes everything a developer needs to know to start developing for the MUSES system.

The MUSES System [1] has been developed based on the client-server architecture.

The client or device is related to the end user, usually an employee who uses a mobile or a portable device possibly personal device to access enterprise resources. From the enterprise security point of view, the system should prevent the user from using the device incorrectly and make sure that the employee follows the company policees. Therefore, MUSES monitors the user's context and behaviour, and controls their actions, allowing or forbidding them depending on company policies.

The MUSES server is controlled by an enterprise security manager, the Chief Security Officer (CSO), who defines the security policies to be considered in the MUSES system. The security policies are used by the MUSES server to identify which behaviour is allowed and which one is not. The server then receives, stores, and processes all the gathered information from users' devices. After that, it analyzes the data, performing a real-time risk and trust analysis. In addition, the system detects policy violations through event correlation techniques, adapting to changes in the environment, as well as non-secure user behaviours.

1.1 MUSES on Github

All the MUSES system code is available at `https://github.com/MusesProject`. That is the Github organisation page for MUSES, and it contains the following repositories:

Muses-Developer-Guide Repository with the TeX files which consist of a manual for users that may want to develop for the MUSES project.

Muses-Security-Quiz Contains the first version of the Spring roo project that implements the Security Quiz for MUSES project.

MusesCommon This repository contains the java classes which are common for client and server projects, to reduce duplication of classes in projects.

MusesServer This repository contains all the files in the Java-Maven MUSES server project.

MusesClient Repository containing the first prototype of the MUSES client, developed for Android.

MusesSituationPredictionStudy Contains the very first prototype of an Android app that uses supervised learning to identify, whether the current device usage is private or professional, with on device classification.

MusesClientIOS Repository containing the first prototype of the MUSES client, developed for iOS.

MusesAwareLibIOS A library that can be used by applications to retrieve information relevant for MUSES.

MusesAwareApp It contains the files of an Android project which consists of a MUSES-aware application. A MUSES-Aware application is an application that uses the MUSES API in order to notify MUSES of user interactions and adapt its behaviour based on MUSES provided commands/security decisions [2].

MusesAwareAppTest Its purpose is testing the MUSES-Aware application.

This guide is structured as follows. First, Chapters 2 and 3 detail how to build MUSES, in order to be able to star developing and testing the system, either you want to develop for the client (Chapter 2), the server (Chapter 3), or both. Then, Chapter 4 specifies how to integrate new sensors to the MUSES client, for better monitoring user's actions and making the *context*[1] *observation* more complete. Finally, Chapter 5 enumerates the steps to install MUSES in a real company environment.

[1] Any information that can be used to characterize the situation of the user.[4]

Chapter 2

Building MUSES Client

The first prototype of MUSES has been developed for Android. This chapter describes the needed tools, as well as building instructions, for developing inside the MUSES client. For instructions about how to install the whole system, please refer to Chapter 5.

1. Prepare eclipse (or any other IDE) for Android
2. Building the Common project
3. Deploy the app on a device or emulator (Building Client project)
   3.1. Create an emulator
   3.2. Prepare your device for usb debugging
   3.3. Deploy

2.1 Installing Android SDK

As we will use Eclipse for MUSES development, the first step is to install the Android Development Tools (ADT) plugin. The requiremenst are [6]:

- Eclipse 3.7.2 (Indigo) or greater.
- Java Platform (JDK) 6.
- Eclipse Java Development Tools (JDT) plugin. To install this pluging, in Eclipse go to *Help > Install New Software...*, and make sure the *Eclipse <youreclipsename> Update Site* is available by clicking in *Find more software by working with the Available Software Sites preferences.*. Then select it and on the package list, look for *Programming Languages > Eclipse Java Development Tools*. Finally, select Eclipse Java Development Tools and click *Next* and *Finish*. Eclipse will need to be restarted.

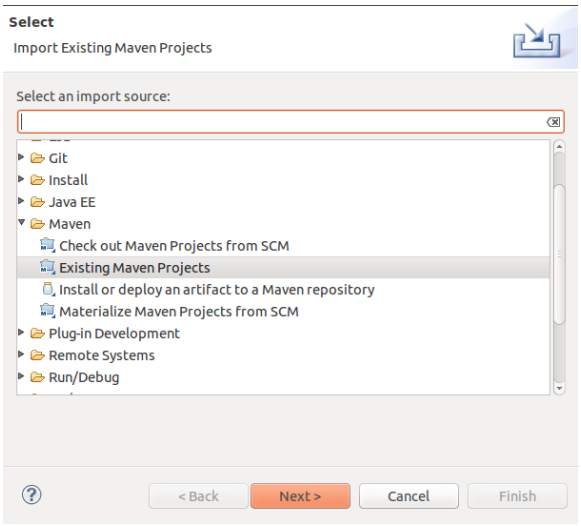

Figure 2.1: Shown dialog after clicking on *File* > *Import...* inside Eclipse.

Now we can install the ADT plugin. This can be done by selecting again *Help* > *Install New Software...* and then looking at this reposiroty `https://dl-ssl.google.com/android/eclipse/`. After that, selecting *Developer Tools* from the package list for installing it (by clicking *Next* and *Finish*), and then restart Eclipse one more time.

2.2 Building Common project

In order to build the server project, we need to build the common project first as the server project has references to common project. As said in Section 1.1, *MusesCommon* is located inside a repository of the MUSES organisation in GitHub (`https://github.com/MusesProject`). You can either download it as a ZIP file, or simply clone or fork the repository. For more information about Git, please refer to GitHub Help [5].

Please note that this is a Maven project (`http://maven.apache.org/`). For this reason, Eclipse should have Maven integrated. Following the steps for installing the JDK od ADK plugins, we should go to Eclipse*Help* > *Install New Software....* Maven will be at `http://download.eclipse.org/technology/m2e/releases` repository [3].

Having this done, next step is to import the common project. On Eclipse, click on *File* > *Import....* A dialog will show up, to select the import source. A folder with the name "Maven" should appear, like in Figure 2.1. If not, please make sure that you have installed Maven for Eclipse, and successfully restarted Eclipse.

Click on *Existing Maven Projects* and *Next*. Then *Browse...* the folder

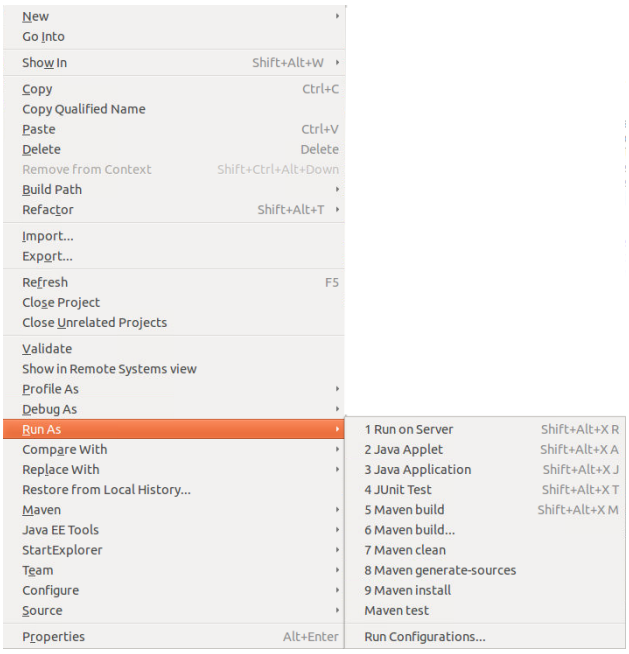

Figure 2.2: How to run a Maven project by making *Run As > Maven clean*, and then *Run As > Maven install*.

where the project is stored. Finally, click *Next* and *Finish*. For running projects with Maven, you must right-click on the project, and then go to *Run As > Maven clean*, as it is shown in Figure 2.2. In the console, you will see a "BUILD SUCESS", and then, repeat going to *Run As > Maven install*. Now we are ready to import and run the client.

2.3 Building Client project

This section is split into three parts *1. Create an emulator*, *2. Prepare your device for usb debugging*, and *3. Deploy*, each of them describing the necessary steps.

2.3.1 Create an emulator

This step is optional and should just be performed if you don't own an Android device to test with, because the emulator of Android has a bad performance and does not support the full functionality of a device, which can influence your debugging experience.

In order to create an emulator, you have to open the Android Virtual Device Manager (ADV). In Figure 2.3.1 you can see that you can open the AVD

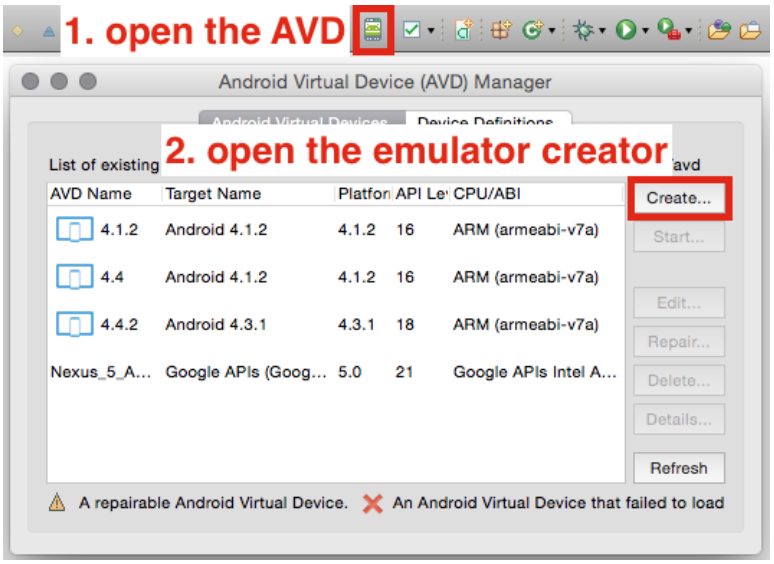

Figure 2.3: Open the emulator window

by following step 1, which shows the corresponding icon in the toolbar of eclipse. Alternatively, you can open this dialog in eclipse by following: *Windows -> Android Virtual Device Manager*. Open the emulator creator (step 2 in Figure 2.3.1) and fill in the fields with your preferences. Remember, we use the target SDK level 19 in the MUSES project.

2.3.2 Prepare your device for usb debugging

Android devices do not support debugging by default, you have to activate the debugging menu on the device manually. Go to the settings of your device and search for the *Build Number*, the location on this setting depends on your device; then you need to click on the *Build Number* seven times. After this, the developer menu is visible in the settings; open this menu and activate *usb debugging*. If your operating system is Windows, you need to download the *usb driver* from the Android SDK Manager (manual: `http://developer.android.com/sdk/win-usb.html`), if you are using a mac with OS X, you do not need to perform any further step. If you are using Linux you need to add a *udev* rule (more information about i on `http://developer.android.com/tools/device.html#setting-up` step 3).

2.3.3 Deployment

The deployment process is straight forward, you just need to right click you project and select *Run As -> Android Project*. Alternatively, you can also configure the deployment of the app and choose whether, you want to deploy

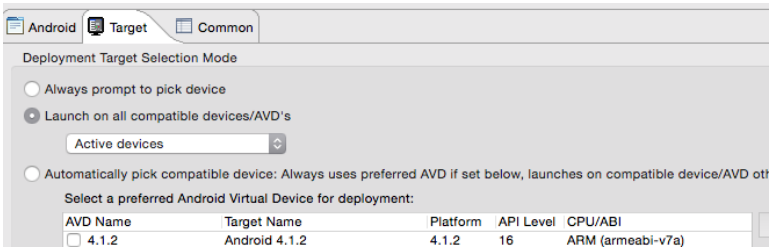

Figure 2.4: Optional run configuration

the app on the emulator or the device by default, or if you would like to choose it individually for each deployment. Figure 2.3.3 shows the dialog of the configuration, you can open this dialog by right clicking on the project *Run As -> Run configurations...*

Chapter 3

Building MUSES Server

MUSES server has been developed using open source guidelines. It implements an Apache Tomcat server, and a MySQL database. This chapter describes all necessary steps to follow in order to build the server and run tests in it.

3.1 Necessary tools and configuration

First of all, and as MUSES server has been developed with Eclipse, these are the required tools to be able to finally build the server project. Therefore, the following are needed:

- Eclipse, no specific version.
- Package *Eclipse IDE for Java EE Developers*. To do this, there are two choices:
  1. In Eclipse go to *Help > Install New Software...*, and select the *http://download.eclipse.org/releases/<youreclipsename>* in the *Work with:* drop-down. Then look for *Web, XML, Java EE and OSGi Enterprise Development* on the package list, and select it. Click *Next* and *Finish*, and restart Eclipse.
  2. The package is available at `https://eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/lunasr2`, and the only thing to take into account is selecting the download according to your Eclipse version (name). On the website, simply select the correct one on "Releases", download it, and install it.
- StartExplorer Eclipse plug-in. This plug-in is available at `http://basti1302.github.io/startexplorer/`, and you can just drag-and-drop the *install* button on your Eclipse menu bar, or following the *Help > Install New Software...* procedure.

Finally, as said in Section 1.1, *MusesServer* files are located inside a repository of the MUSES organisation in GitHub (`https://github.com/MusesProject`). You can either download it as a ZIP file, or simply clone or fork the repository. For more information about Git, please refer to GitHub Help [5].

Before building the server project, MusesCommon project should be built. Please follow instructions of Section 2.2 before continue doing anything else.

3.2 Bulding server project

If Section 2.2 steps were successfully completed, we should have Maven correctly installed, and the MusesCommon project already builded. Now, the MusesServer project should be imported into the workspace in the same way, as a Maven project.

As the database is implemented on MySQL, we have to make sure that we have the *mysql-connector-java* library properly installed. By right-clicking on the server project, at the end of the emerging window click on *Properties*. The library should appear in the list which is in *Java Build Path > Libraries tab*. If it does not appear, click *Cancel* and in Eclipse, go to *Window > Show view > Other....* In the new window, inside the *Java* folder, select *Package Explorer* and then *Ok*. A new tab will appear next to "console", "error log", ..., tabs. This tab shows the projects, like the *Project Explorer* does. But in this view, we are able to use the *StartExplorer* plug-in. By right-clicking in the MusesServer project, go to *StartExplorer > Start Shell Here*, and a terminal will appear. If you cannot find this option, make sure that the *StartExplorer* plug-in is installed (see Section 3.1). To obtain all the necessary configuration for managing MySQL databases and, in general, a Maven configuration for Eclipse, type into the terminal: `mvn eclipse:eclipse`. Wait for it to finish.

3.2.1 Preparing the database

The script for creating the MUSES server database is located at `MusesServer/db/startup_db.sql`, and it should be loaded in the MySQL server at our machine. First, make sure you have MySQL server installed. If not, open a terminal and type `sudo apt-get install mysql-server-5.6`. During the installation, you will be asked to set a password for the root user. Few basic steps [9] should be followed to load the MUSES database:

`$ sudo /etc/init.d/mysql start`

`$ mysql -u root -h localhost -p` and then enter the chosen password.

`SOURCE /home/yourusername/foldertoMusesServer/db/startup_db.sql;` which loads the database structure from the script.

`SOURCE /home/yourusername/foldertoMusesServer/db/startup_db.sql;` which populates the loaded database.

`SHOW DATABASES;` should list the databases, and also MUSES database.

`USE muses;`

`SHOW TABLES;` should list the tables inside MUSES database.

`CREATE USER 'muses'@'%' IDENTIFIED BY 'muses11';` creates a remote user for MUSES.

`GRANT ALL ON muses.* TO 'muses'@'%';` which gives permission to the MusesServer project to access the content in the MUSES database. Replace '%' with localhost to make it only accessible from localhost.

Finally, go to Eclipse again, right-click on the MusesServer project, and then go to *Run As > Maven clean*, as it was shown in Figure 2.2. In the console, you will see a "BUILD SUCESS", and then, repeat going to *Run As > Maven install*. Check that there were no errors, specially this type of error: *`Error calling Driver#connect`*, which means that the code cannot connect to the database. Now, you can implement functionalities and test them locally by doing *Run As > Maven install* or *Run As > Maven test*.

3.3 Integrating Tomcat server

Now that we can have a running client, by following the described steps on Section 2.3, and that we just builded the server part (see previous section), it is time for integrating Tomcat on Eclipse. After that, we will be able to test our development in a client-server mode, and locally. And this is an important point, because this step is for development purposes only. To configure Apache Tomcat in a system, in order to work as a MUSES server, please refer to Section 5.1.

To integrate Apache Tomcat on Eclipse three steps are basically needed [7]: to install Apache Tomcat, to add it as a runtime environment, and to add the Tomcat server, strictly speaking.

3.3.1 Installing Apache Tomcat

Apache Tomcat powers numerous large-scale, mission-critical web applications across a diverse range of industries and organizations. But first of all, minimum Java Runtime Environment (JRE) 6 is required for Tomcat 7. For being able to install the last available version of JRE on the machine, open a terminal and follow this command:

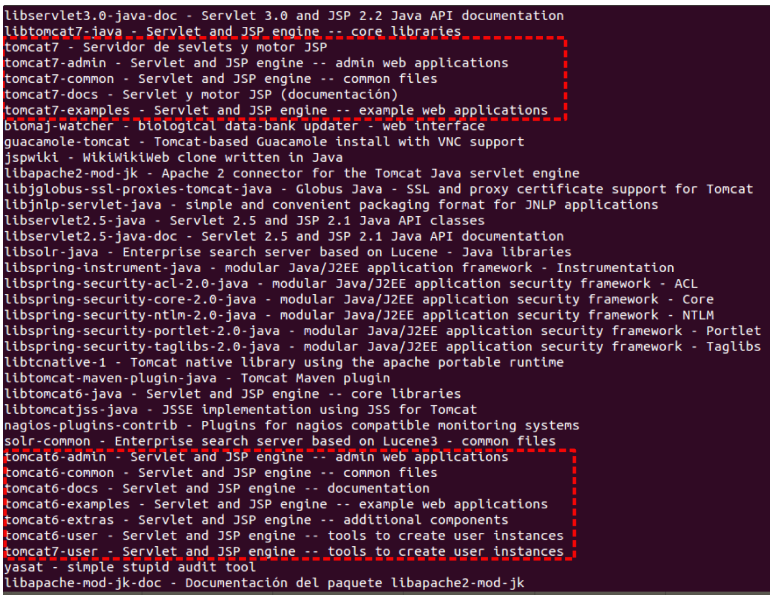

Figure 3.1: What your terminal should show after introducing the `$ apt-cache search tomcat` command. The highlighted packages are the ones we have to focus in.

```
$ sudo apt-get install openjdk-7-jre
```

If you find problems with the dependencies, the command `$ sudo apt-get -f install` fixes the possibly broken dependencies and finishes the installation. Also, you may need to update your system package list, for what you have to do the following:

$ sudo apt-get update to update the lists.

$ sudo apt-get upgrade to install the new version of the packages, found by *updating* the lists.

Find a correct Tomcat package to install by typing `$ apt-cache search tomcat` and looking at the highlighted parts on Figure 3.3.1. In this case, the needed version is version 7.

Then, to install the Tomcat server package, the following command is needed [8]:

```
$ sudo apt-get install tomcat7
```

And for now, what we need to know is that the configuration files will be at `/etc/tomcat7`, and the installation directory is `/usr/share/tomcat7`.

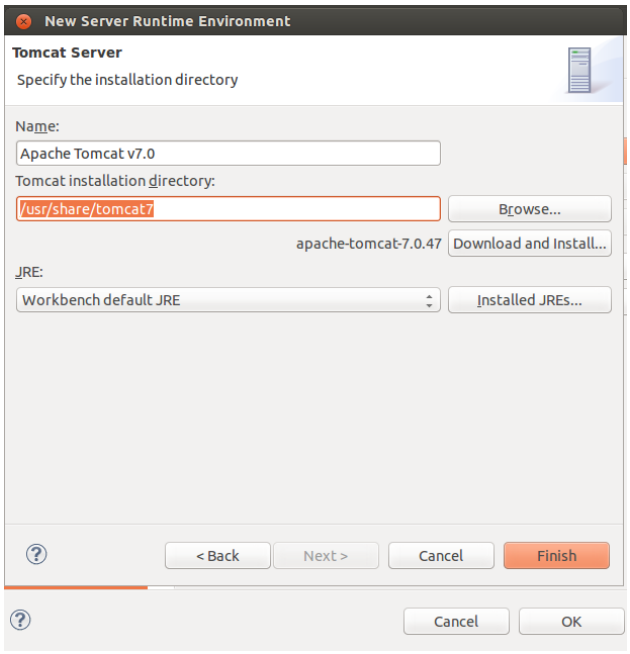

Figure 3.2: Eclipse window which helps to add a new Runtime Environment. In our case, it will be Apache Tomcat version 7.

3.3.2 Adding Apache Tomcat as a Runtime Environment

On Eclipse, go to *Window* > *Preferences* > *Servers* > *Runtime environment* > *Add...* and after selecting *Apache Tomcat v7.0*, and clicking *Next*, we should see a window like the one shown in Figure 3.3.2. As can be seen, we put the installation directory where is required and click *Finish*. It should now appear on the window before we clicked *Add...*. If not, check the steps before finally clicking *OK*.

3.3.3 Adding Apache Tomcat as a server

This is the last step. On Eclipse, right where the tabs of "console", "error log", ..., are, there is a tab called *Servers*. A message like this should appear: "No servers are available. Click this link to create a new server...". Just click on this link or, if you have previously configured servers, just right-click and select *New* > *Server*. We will see a list similar to the previous one, and we select again *Apache Tomcat v7.0*. Finally, we write a name for our server and click *Finish*. It should now appear in the list of servers at the Servers tab, as shown in Figure 3.3.3.

If all went well, and by making double-click on the server which was just added, a new window should appear, and it should look like Figure 3.3.3.

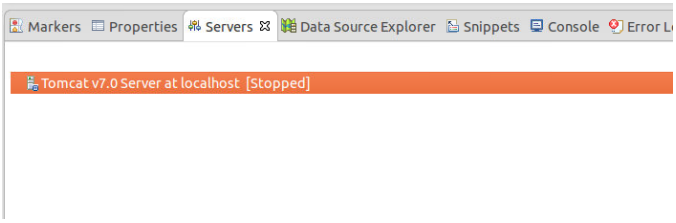

Figure 3.3: Apache Tomcat installed and added into Eclipse.

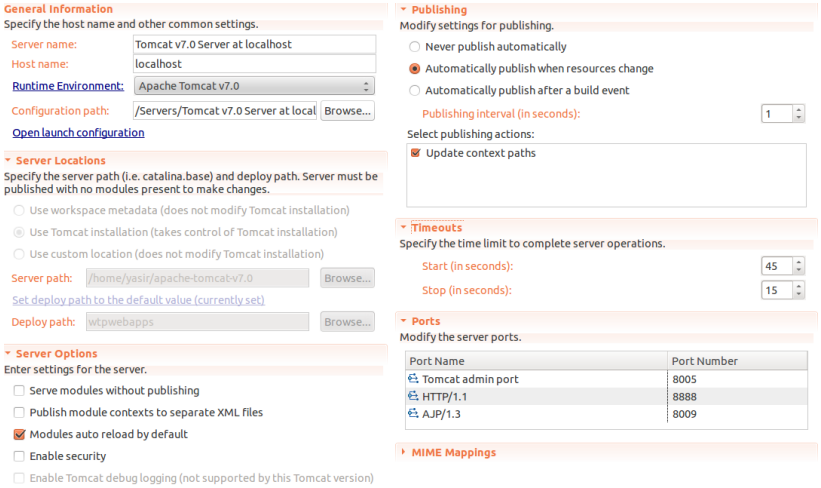

Figure 3.4: Editing basic information about the Apache Tomcat server.

In this view, basic information about the Tomcat server on Eclipse can be changed.

Chapter 4

Creating and integrating new sensors

A new sensor must implement the `eu.musesproject.client.contextmonitoring.sensors.ISensor` interface. This interface has six methods:

void addContextListener(ContextListener listener); This method takes a ContextListener object, the listener method **onEvent(ContextEvent contextEvent)** must be called whenever the sensor shall send an event to the MUSES system.

void removeContextListener(ContextListener listener); Should contain a null object to remove the reference to the previous ContextListener object. Usually it is not necessary to call this method.

void enable(); The startup process of the sensor should be written or triggered in this method.

void disable(); Contains code to stop the sensor so that no more context events are fired to the system.

ContextEvent getLastFiredContextEvent(); Each sensor has a history of its fired context events; therefore a List<ContextEvents> must be implemented. The size of the list is limited by the static field CONTEXT_EVENT_HISTORY_SIZE, which is located in the ISensor class. The developer has to take care to update the list probably.

void configure(List<SensorConfiguration> config); If a sensor needs to be configured as it might expect specific values in order run as intended, the developer has to take care about the List<SensorConfiguration> config object. The SensorConfiguration class is located in the package: `eu.musesproject.client.db.entity`.

Each sensor must provide a **public static final String** *TYPE* field. This field is used to identify the sensor. The naming convention is: `CONTEXT_SENSOR_<SensorIdentifier>`. In addition, and in order to create properties for context events, the sensor must contain a **public static final String** field for each property identifier.

4.1 How to integrate new sensors

First, new sensors should be placed within the MUSES Client project in the following package:

```
package eu.musesproject.client.contextmonitoring.sensors;
```

Second, in the class:

```
eu.musesproject.client.contextmonitoring.SensorController
```

the new sensor has to be initialized. And this has to be done inside the method:

```
public void startAndConfigureSensors(List<String> enabledSensor){}
```

Within this method the initialization has to be done as in the following example:

```
else if(sensorType.equals(SettingsSensor.TYPE)) {
sensor = new SettingsSensor(context); }
```

Chapter 5

Installing MUSES

Table 5.1 shows the hardware requirements to run MUSES server applications. The following sections will explain how to install and configure these requirements but the database, which was explained in Section 3.2.1.

Table 5.1: MUSES server infrastructure.

MUSES server		
Description	**Name**	**Version**
Hardware	Desktop PC	Intel Core 64 bit
OS	Ubuntu Desktop	12.04 LTS
Virtual Machine	JVM[1]	1.6.0_27
Servlet Container	Apache Tomcat	7
WEB Server	Apache	2.2.22
Database	MySQL	5.5.31

5.1 Configuring Apache Tomcat (Application Server)

Section 3.3 explained how to install the last version of Apache Tomcat server, because it was needed for developing purposes. Once you have followed the steps in that section, we continue configuring the server in order to finally have a working MUSES environment on our system.

As said in that section, Tomcat configuration files are at `/etc/tomcat7` directory. Among the configuration files, *server.xml* can be found. Also, Figure 3.3.3 shown the configuration of Apache Tomcat, which includes the listening port. The default listening port for Apache is 8080, but we see in the figure that we have configured it at port 8888. The file *server.xml* offers the possibility of changing the default port:

```
<Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
...
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

By simply changing *connector port* 8080 for 8888. Please note that you may have to open *server.xml* under superuser rights.

Then, we have to install the Tomcat web application administrator by this command [8]:

```
$ sudo apt-get install tomcat7-admin
```

In order to add manager privileges, and to be able to use Tomcat web applications like *host-manager* [8], we need to define a *"manager-gui"* user in *tomcat-users.xml* file. This file is also among the configuration files at `/etc/tomcat7`. Open this file under superuser rights and make sure to change it until it looks like:

```
<?xml version='1.0' encoding='utf-8'?>
  <tomcat-users>
    <role rolename="manager-gui"/>
    <role rolename="manager"/>
    <role rolename="admin-gui"/>
    <role rolename="admin "/>
    <user username="admin" password="admin"
          roles="manager, manager-gui"/>
  </tomcat-users>
```

5.1.1 SSL configuration Tomcat

MUSES is using HTTPS in order to make the communication encrypted, and secure. SSL encryption has to be enabled in Tomcat server configuration with an SSL certificate location, on the server machine. Back again in the Tomcat configuration folder, at `/etc/tomcat7`, the *web.xml* file allows to configure HTTPS on tomcat.

This way, open the *web.xml* file and add the following lines:

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>ComMainServlet</web-resource-name>
    <url-pattern>/commain</url-pattern>
  </web-resource-collection>
```

```
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

This will allow the MUSES servlet to listen for HTTPS connections. However, we also have to modify the file *server.xml*, which is the same that have to be modified to change the listening port (see the beginning of this section). What we need now is to enable SSL authentication by adding:

```
<Connector
  SSLEnabled="true"
  ciphers="SSL_RSA_WITH_RC4_128_SHA"
  clientAuth="false"
  keystoreFile="/etc/tomcat7/keystore.jks"
  keystorePass="MUSES11"
  maxThreads="150"
  port="8443"
  protocol="HTTP/1.1"
  scheme="https" secure="true" sslProtocol="TLS" />
```

Finally, Apache Tomcat has to be restarted in order to successfully apply the changes. This is done by:

```
$ sudo service tomcat restart
```

5.2 Starting MUSES

To be able to start MUSES, the WAR file (Web Archive file) is required. The Tomcat 7.0 application server setup requires deploying the war file in the tomcat container. Follow below instructions.

Open a web browser and go to this link `http://localhost:8080/manager/html` It will ask for the user name and password. Enter the credentials which have been set during tomcat configuration[2], then it will open the manager page for tomcat. Now in section ***WAR file to deploy*** click on *Browse...* and select the war file, then click *Deploy*. After doing this, the application should be listed in the applications column. Now click on the application button to start the server.

5.3 MUSES client

The following steps can be followed either from Windows, MacOS, or Ubuntu.

[2]In our case, we used admin/admin at *tomcat-users.xml*, but they can be changed.

First of all, in order to install the MUSES application on an Android phone, the MUSES APK file is needed. As said in Section 2.3, this is a file that results from compiling the client code. Then, for installing the APK, the MUSES APK file has to be stored into the SD card of the device. Finally, take the Android phone, click on the file once it is stored in the SD card and, wait for it to be installed.

Next step is to add the server certificate also to the SD card. This certificate has been named *localhost.crt* and can be found at the MusesClient Github repository (for more details, please see Section 1.1), inside the *Assets* folder. It is important to store this certificate at the device SD card root directory, so it should be at the location */sdcard/localhost.crt*.

Finally, we must add the configuration file. This file is named *muses.conf* and can be found at ___. Like the server certificate, the configuration file should be stored at the device SD card root directory.

Bibliography

[1] H. Arfwedson, M. Burvall, Y. Ali, A. Mora, P. de las Cuevas, S. Zamarripa, J.-M. Seigneur, and Z. Hodaie. D2.1 architecture and prototype specification. 2013.

[2] A. Esparcia, F. Barceló, A. Mora, P. de las Cuevas, J.-M. Seigneur, M. Busch, F. Freschi, M. Burvall, and S. Piccione. D2.4 muses glossary of terms. 2013.

[3] T. E. Foundation. M2eclipse.

[4] Z. Hodaie, N. Mannov, S. Zamarripa, Y. S. V. D. Sype, M. Burvall, and W. Maalej. D6.1 conceptual model of user observation, application monitoring tools, and context actuators. 2014.

[5] G. Inc. Github help.

[6] G. Inc. Installing the eclipse plugin.

[7] I. MuleSoft. Apache tomcat eclipse integration.

[8] U. D. Team. Ubuntu documentation, apache tomcat.

[9] C. White, S. Mani, and X. Neys. Mysql/startup guide.