Final Release of the Policy Engine

Editors: Slim Trabelsi (SAP)

Reviewers: Tobias Pulls (KAU)

Identifier: D5.3.3
Type: Deliverable
Class: Confidential
Date: May 18, 2011

Abstract

This document is a user manual to install and test the PrimeLife Policy engine.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 216483 for the project PrimeLife.

Members of the PrimeLife Consortium

1.	IBM Research GmbH	IBM	Switzerland
2.	Unabhängiges Landeszentrum für Datenschutz	ULD	Germany
3.	Technische Universität Dresden	TUD	Germany
4.	Karlstads Universitet	KAU	Sweden
5.	Università degli Studi di Milano	UNIMI	Italy
6.	Johann Wolfgang Goethe – Universität Frankfurt am Main	GUF	Germany
7.	Stichting Katholieke Universiteit Brabant	TILT	Netherlands
8.	GEIE ERCIM	W3C	France
9.	Katholieke Universiteit Leuven	K.U.Leuven	Belgium
10.	Università degli Studi di Bergamo	UNIBG	Italy
11.	Giesecke & Devrient GmbH	GD	Germany
12.	Center for Usability Research & Engineering	CURE	Austria
13.	Europäisches Microsoft Innovations Center GmbH	EMIC	Germany
14.	SAP AG	SAP	Germany
15.	Brown University	UBR	USA

List of Contributors

Chapter	Author(s)
Introduction	Slim Trabelsi (SAP)
Chapter 1	Slim Trabelsi (SAP)
Chapter 2	Slim Trabelsi (SAP)
Chapter 3	Slim Trabelsi (SAP)

Contents

List of Figures

Chapter 1

Introduction

This document presents the implementation and integration results obtained in the context of Work Package 5.3. The main goal of the deliverable is to develop the final release of the PPL (PrimeLife Policy Language) policy engine proof of concept . The concept and the specifications of this language are defined in the final public deliverable D5.3.4 [1]. The PPL language is specified as an extension of the XACML (eXtensible Access Control Markup Language) [2] language, then the PPL engine is designed to run together with the HERAS-XACML engine [3] (that only handles XACML access control rules). The architecture chosen for the deployment of the PPL engine is symmetric because Data Subjects and Data Controllers (have similar requirements: deciding whether a given personal identifiable information (PII) can be shared with a data controller (resp. third party); handling obligations associated with data; storing data and associated preferences (resp. sticky policies). Using the same architecture everywhere to handle scenarios where one party can have multiple roles (e.g. collecting data and next disclosing it to third parties). The PPL engine executes multiple tasks in an automated way like: enforcing access control policies, generating and verifying cryptographic proofs related to credential requirements, matching between data handling preferences and data handling policies, generating and enforcing sticky policies, checking authorization, controlling the downstream usage of data, handling obligations. Compared to the second release of the PPL engine, the final one addresses the performance problems and adds a new functionality related to preference group management described in [1] Section 7.6. This document is a manual explaining to the reader how to install the system environment, launch the PPL engine and test it trough a web based scenario.

This deliverable is organized as follows. In the first chapter we describe the use case scenario that will be the framework of the demonstrator. In chapter 2 we describe the installation steps of the execution environment and the PPL engine. Finally in chapter 3 we explain how to use the engine according to the use case.

Chapter 2

Use Case Scenario

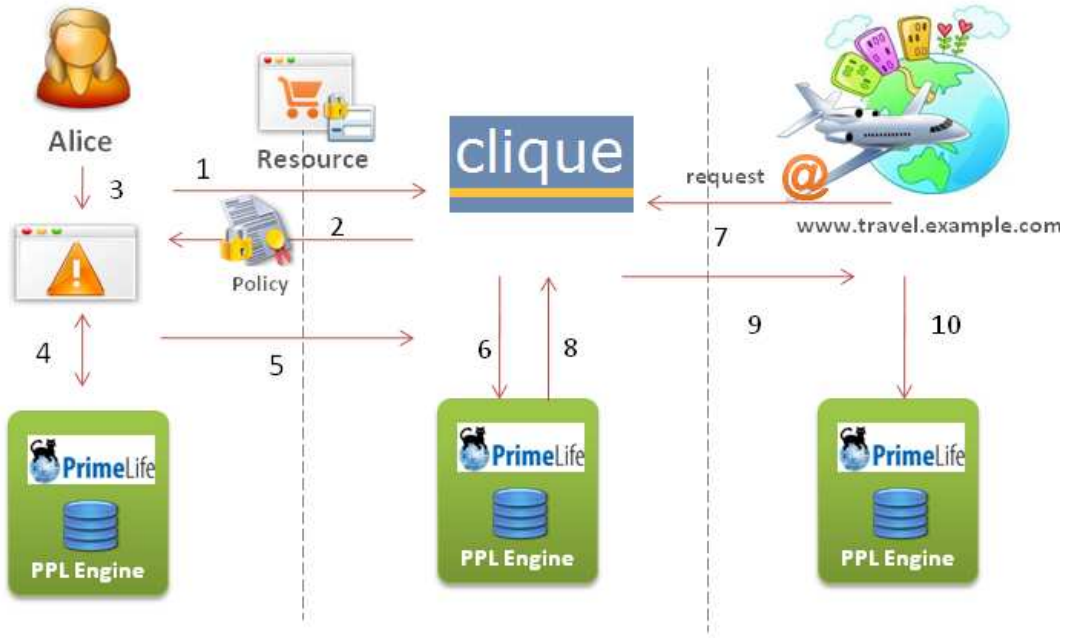

Figure 1: Use case scenario: connection to a social network website

To explain in a more easy way how the PPL engine works, we present a scenario that describes a subscription to social network shopping website. We present in Figure1 the different information exchanged between the different parties using the entities format presented in the domain class diagram above (Figure 1).

- The Data Subject Alice is a privacy-aware user who is quite active in the Web 2.0 websites, but who is concerned about what happens to the data that she provides about herself. Before starting her social networking activity, she has to create an account at an

online social network https://clique.primelife.eu/[1] (step 1). Clique is playing the role of Data Controller.

- In order to validate her subscription, the website will need to collect some personal information, like her name, her birth date, her e-mail address and her street address... This information is contained in an access control policy on the website side (ie. In order to create an account the user has to provide a list of credentials). In order to explain the conditions of usage of such collected information the website sends a privacy policy (written in PPL language) (Step 2).
- Initially Alice created privacy preferences related to her personal data using the PrimeLife Privacy Tuner shown in Figure 2 (Step 3). This tuner is a graphical tool used to edit Privacy preferences in PPL language defined in the deliverable [4].
- After receiving the website's privacy policy, the PPL privacy engine of Alice automatically check if the private data requested by the server is stored on Alice machine. If it is the case, the engine will enforce the access control rules related to the requested data (ex. does the domain primelife.eu can access my e-mail address?). If the domain is allowed to access this data the engine match the privacy policy of the website with the preferences of Alice. This matching will compare the data handling conditions expressed by the two parties (Step 4).
- Alice has the possibility to decide if she accepts or refuses to send her data. This decision is supported by a graphical representation of the matching result implemented as a browser plug-in (See Figure 2). The result of this matching is contained in a sticky policy that will be sent to the server of the website together with the private data of Alice (Step 5).
- There the server will store and use the data according to the obligations stated in the sticky policy (Step 6).
- The online travel agency www.travel.example.com (third party Data Controller) decided to start an e-mail advertising campaign. In order to target a wide scope of persons, the www.travel.example.com admin asked his partner clique.primelife.eu (Step 7) to provide him with valid e-mail addresses for marketing and statistics purposes. The request will contain a resource query for e-mail and a privacy policy.
- The policy engine of clique.primelife.eu will match the privacy policy of travel.example.com with the sticky policy related to the e-mail of Alice (step 8), and will check if the sticky policy allows to forward for the purpose of statistics for example. In general, the full matching occurs between travel.example.com's policy and Alice sticky policy.
- The travel.example.com website receives (Step 9) the e-mail address of Alice with a sticky policy and configures the actions and the triggers related to the obligations, after storing the e-mail and the sticky policy in a secure manner (Step 10).

[1] Clique (https://clique.primelife.eu/) is an experimental social network developed in the context of PrimeLife. In this Demo we are using only the login functionality of this social network.

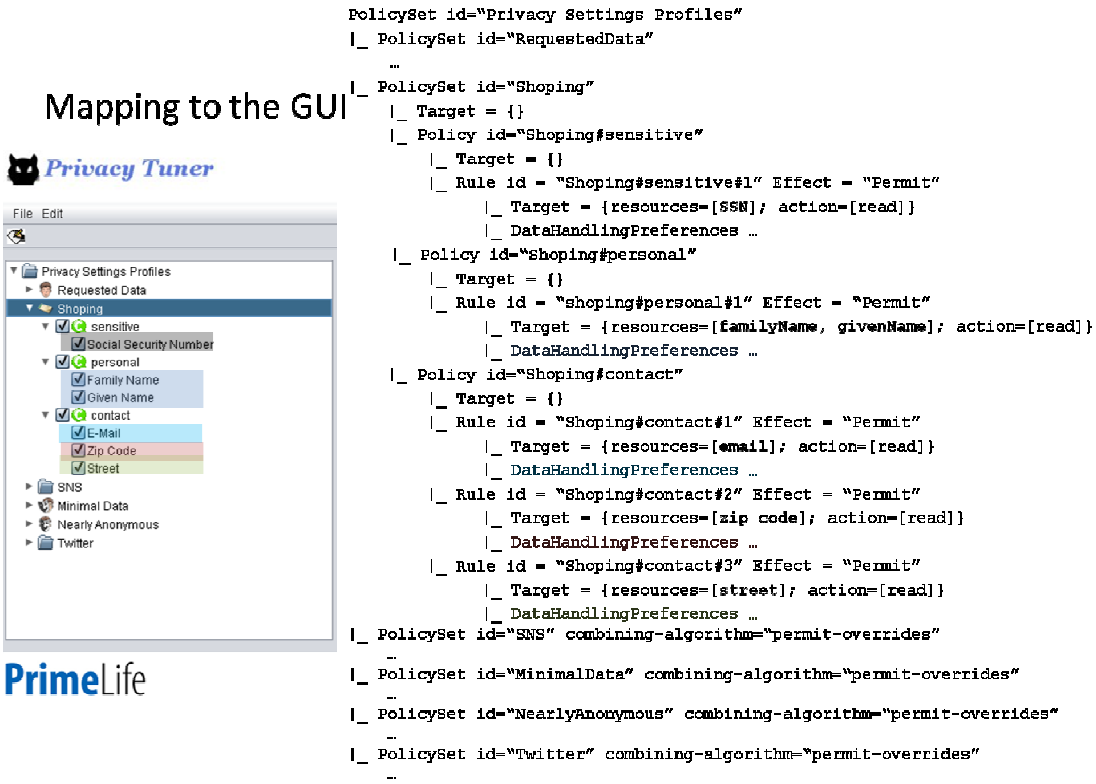

Figure 2: Privacy Tuner Snapshot

In the chapter 3 we will guide you through the different steps described before in order to test the PPL engine capabilities.

Chapter 3

Installation of the Engine

In this section we explain how to install the system environment of the PLL engine. We decided to add a web interface in the scenario in order to make more user friendly the interaction with the engine. For this version of the engine, the tester has to change manually the XML policies that are used by the engine in order to perform some tests. The idea here is to deploy the PPL engine on the Data Controller and the Data Subject sides and execute a simple scenario of a registration to the clique website. The scenario is quite simple, but the idea here is to play with this tool by verifying the generated policies, sticky policies and claims. The user of this demo can modify the policies and the preferences and observe the behaviour of the engine.

3.1 System Requirements

First of all copy the zip file , extract the archive on your local folder that we will call $root. All the install files of these software are copied in $root\softwares

3.1.1 Database

We used the MySQL 5.1.43 database. In order to manage this database we used the EasyPHP 5.3.5 package that includes the Apache HTTP web server, PHP, MySQL and phpMyAdmin. The install including MySQL and PHP can be found here $root\softwares\EasyPHP-5.3.5.0-setup.exe

3.1.2 .Net Framework

The obligation handler and the obligation matching engine are launched as Web Services running over the .Net framework. You can find the installation file in $root\softwares\dotNetFx40_Full_setup.exe

3.1.3 Java VM

The executable for the PPL engine requires Java version 1.6. In order to install it you can use the JDK file in $root\softwares\jdk-6u25-windows-i586.exe. 6. In order to configure correctly

your calsspath you can add a new variable PATH with the following value: \Program_Files\Java\jdk1.6.0_21\bin as shown in Figure 3

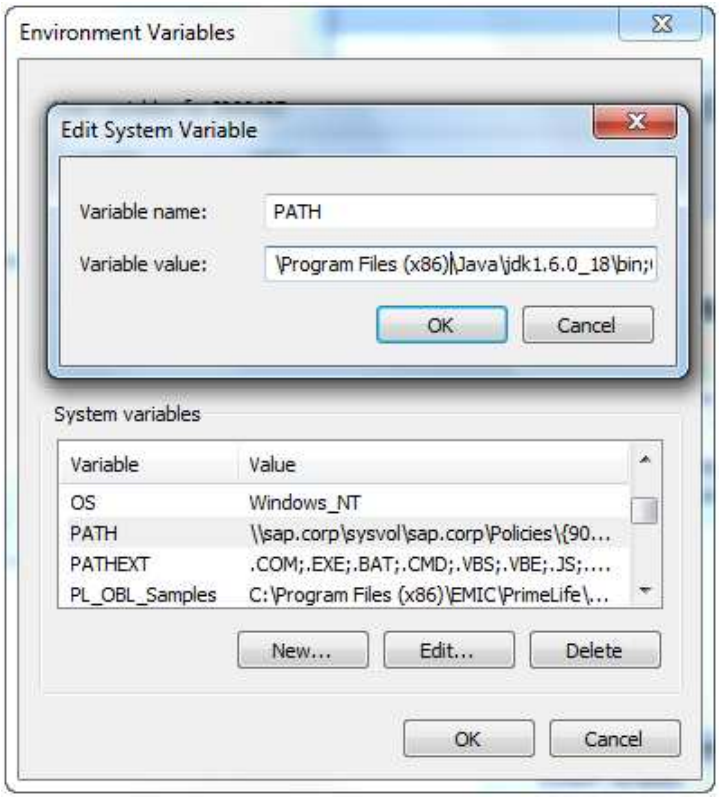

Figure 3: Configuring the PATH variable of the system

3.1.4 Mozilla Firefox

The demo is web-based and the user interface should be displayed in a browser. The user interface is a Firefox Plug-in that is compatible with the Firefox 4 Beta 3 version. For this reason this version of the browser must be installed and the setup file can be found in: $root\softwares\Firefox Setup 4.0 RC 1.exe

3.1.5 PrimeLife PolicyUI: Firefox Plug-in

This plug-in is only available in the Install directory under the name PrimeLifePolicyUI.xpi. the file can be found in $root\partner_software\firefox_plugins\PrimeLifePolicyUI.xpi. In order to install the plug-in you just have to drag and drop this file into the Firefox browser. In order to configure it please set Set the PPL Engine URL to : http://localhost:9477/api/ and make sure that the suffix is empty. The default is .php as shown in the Figure 4.

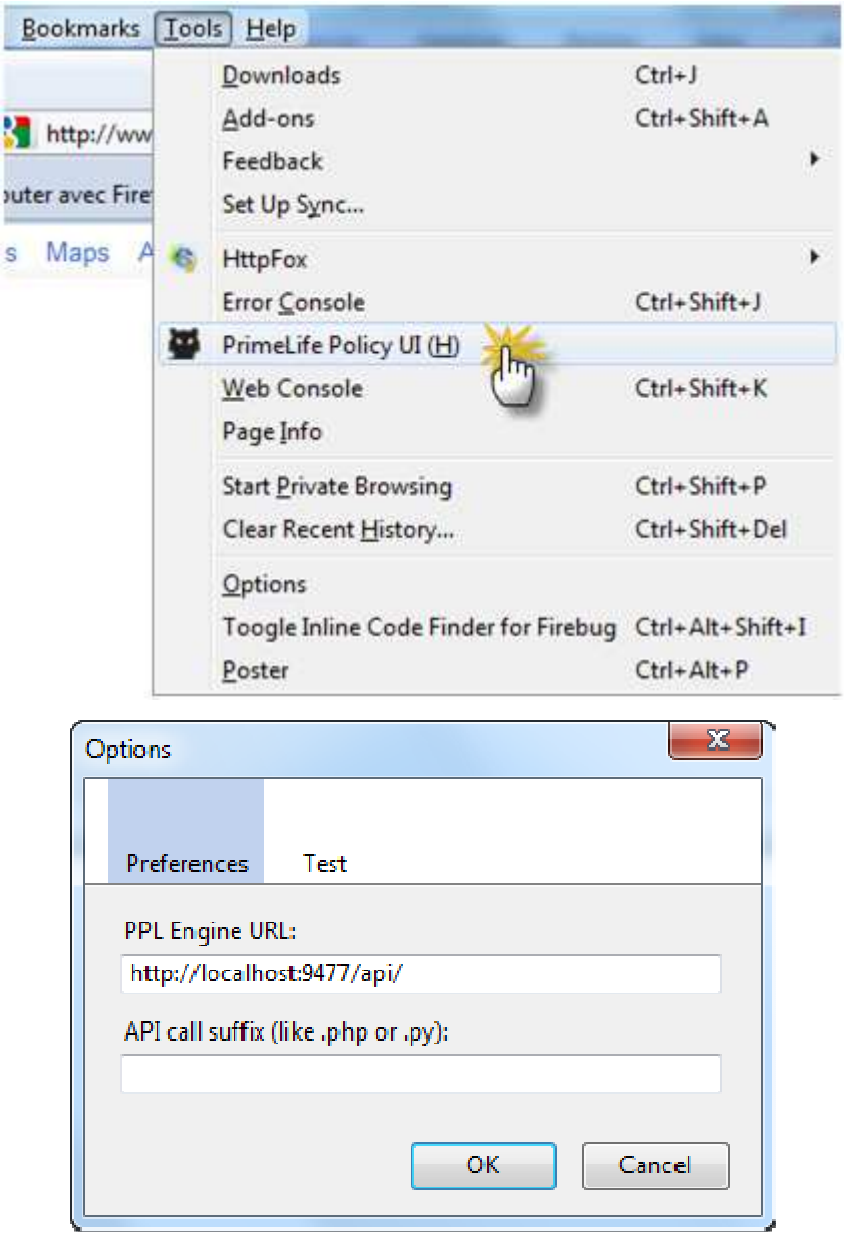

Figure 4: Configuring the PolicyUI

3.1.6 Obligation Enforcement and Matching Engine

Before starting to use this engine it has to be installed as a Windows service.The istall file can be found in $root\partner_software\EMIC\PrimeLife Obligation Matching Engine.msi

3.2 Configuring the PPL Engine

3.2.1 Initialize the Database

Launch your MySQLAdmin session from you're my SQL application (Figure 5 shows the admin interface of the EasyPHP application). Open phpMyAdmin in your browser: http://127.0.0.1/home/mysql/. Create 3 databases with the names 'ppl'(Data Subject), 'ppl-dc' (Data Controller), 'ppl-3p' (third Party). Run the script $root\database-reset\reset-productive.bat.

This may take some minutes. It resets the databases needed for production mode. If you only need the ppl-dc database for example, call `$>java –jar database-reset.jar dc`

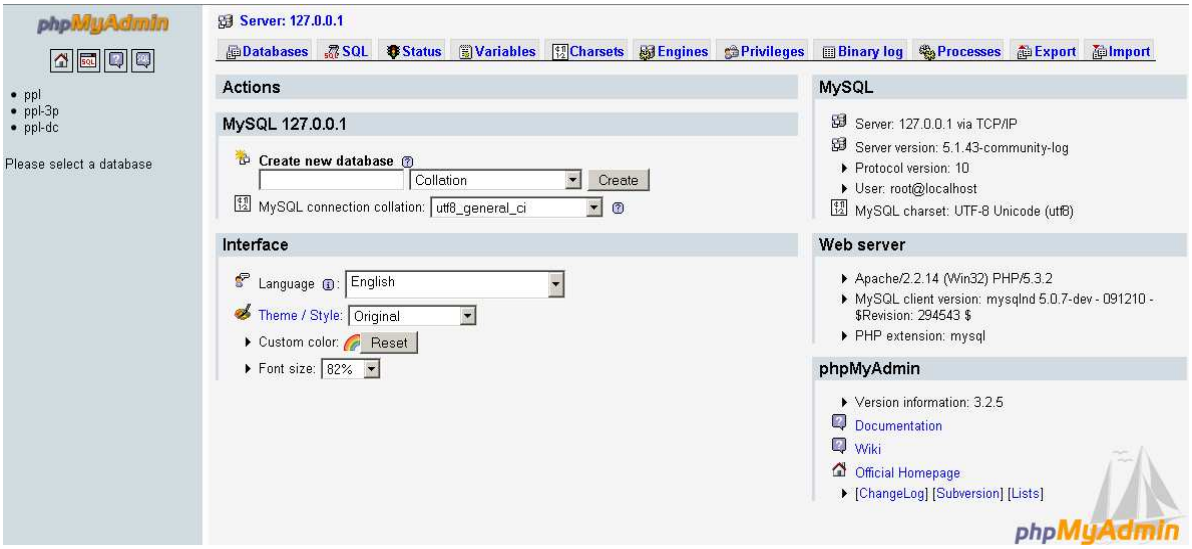

Figure 5:MySQL interface

3.2.2 Start the PPL Engine

First make sure EasyPHP is running and all required databases have the 228 tables. Launch the Firefox 4 browser. Run the Data Controller: $root\ppl-engine\dc.bat (may take ~70sec). Run the Data Subject: $root \ppl-engine\ds.bat (may take ~70sec). Run the Third Party demo: $root\ppl-engine\3p.bat.

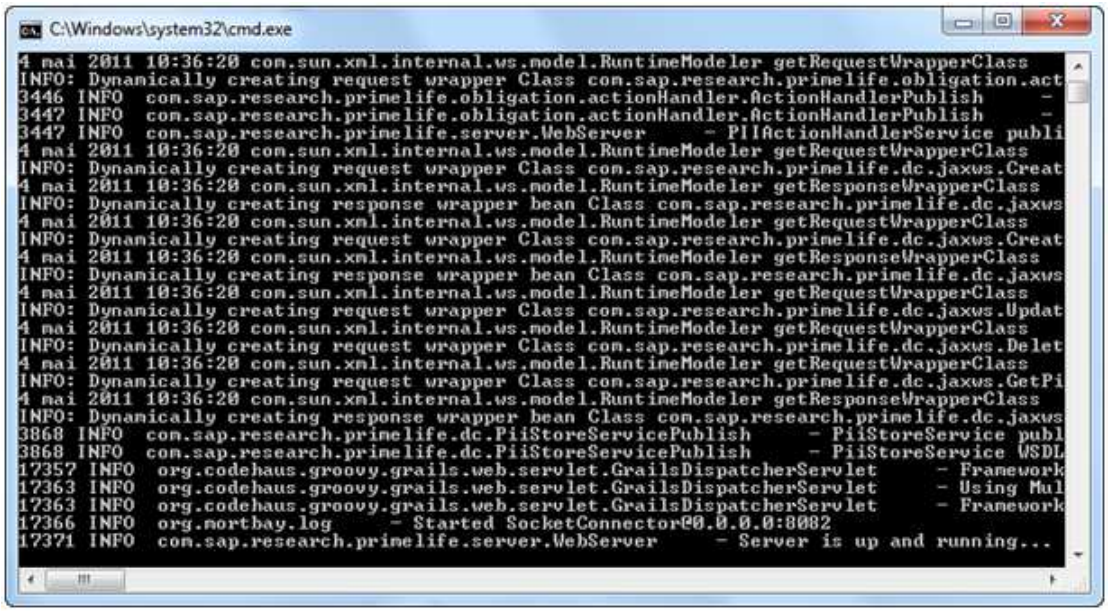

Figure 6: Launching the PPL engine

Chapter 4

Using the PPL Engine Demo

In this section we explain step by step how to use the PPL engine according to the scenario described in Chapter 2.

4.1 Data Subject: Configuring PIIs and Preferences

The Data Subject has to store his PIIs and the related Preferences in to the engine Database. This configuration will permit to the engine to automatically deal with the Data Controller requests.

4.1.1 Creating PIIs

Browse to http://localhost:9477 .You should create the three PII entries for each of the required PII type as shown in Figure 7. You can create for example an e-mail address preset these values:

Attribute Name= http://www.w3.org/2006/vcard/ns#email

Attribute Value= test@example.org

In order to be aligned with our scenario you have to create two other PIIs (display_name and user_name) as shown in Figure 8.

At any time the user can list all his PIIs by clicking on the PII Store menu.

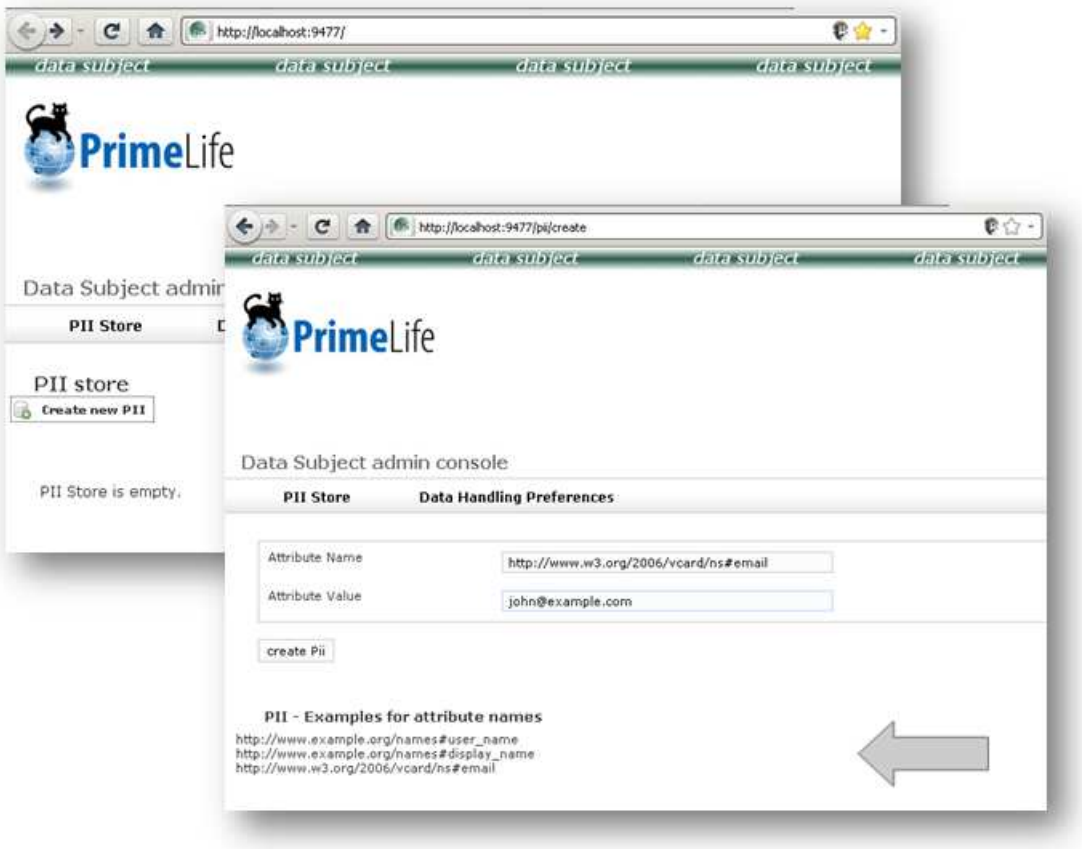

Figure 7: Creating PIIs

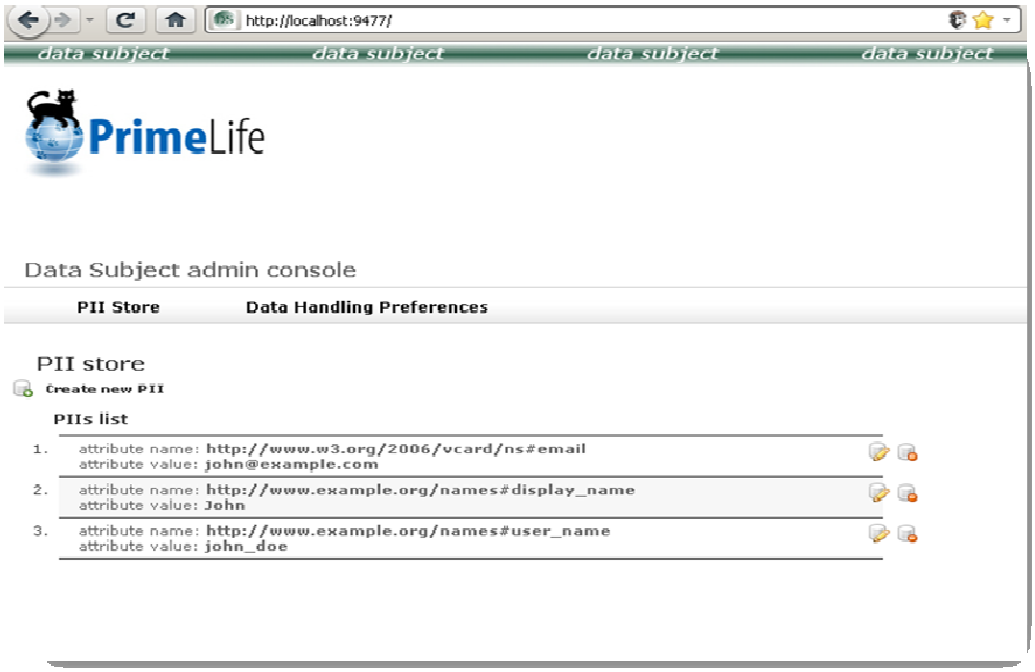

Figure 8: PII list

4.1.2 Associating Preferences to PIIs

The user can edit the preferences related to each of his PII. We use the Privacy Tuner to edit such preferences. To simplify the task we pre-defined the preferences for the selected PIIs. To affect this preferences to the PIIs click on the “Data Handling Preferences” menu, then browse to select a Preference. We recommend you to use the one stored in $root \samples\datahandling_preferences.xml as shown in Figure 9.

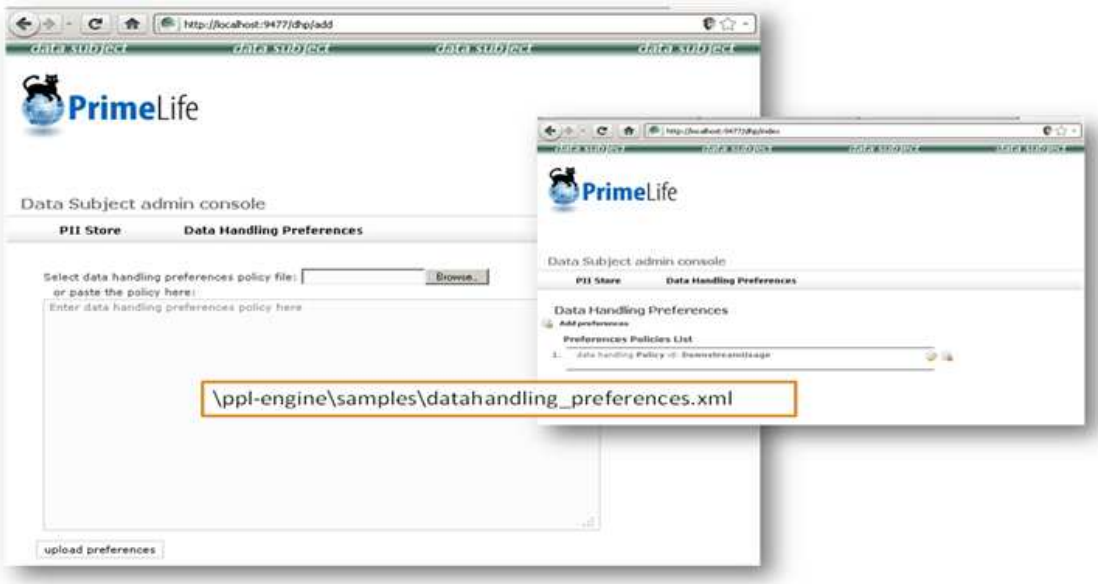

Figure 9: Adding Preferences

4.2 Data Controller: Configuring the Privacy Policy

The Data Controller has also to configure the access control rules related to his resources, and also the data handling policy related to the usage conditions of the collected data. All these rules are contained in a single policy that we call privacy policy.

The Data controller admin console is running in this url: http://localhost:8082/

In order to affect the resources pages to a privacy policy click on the “Resource Policy Store” menu or browse to the url: http://localhost:8082/resource/index
Select type the name of the resource to protect then browse the sample repository in order to select the privacy policy that should be affected (as shown in Figure 10). We recommend you to use the policy contained in $root \samples\datahandling_policy.xml

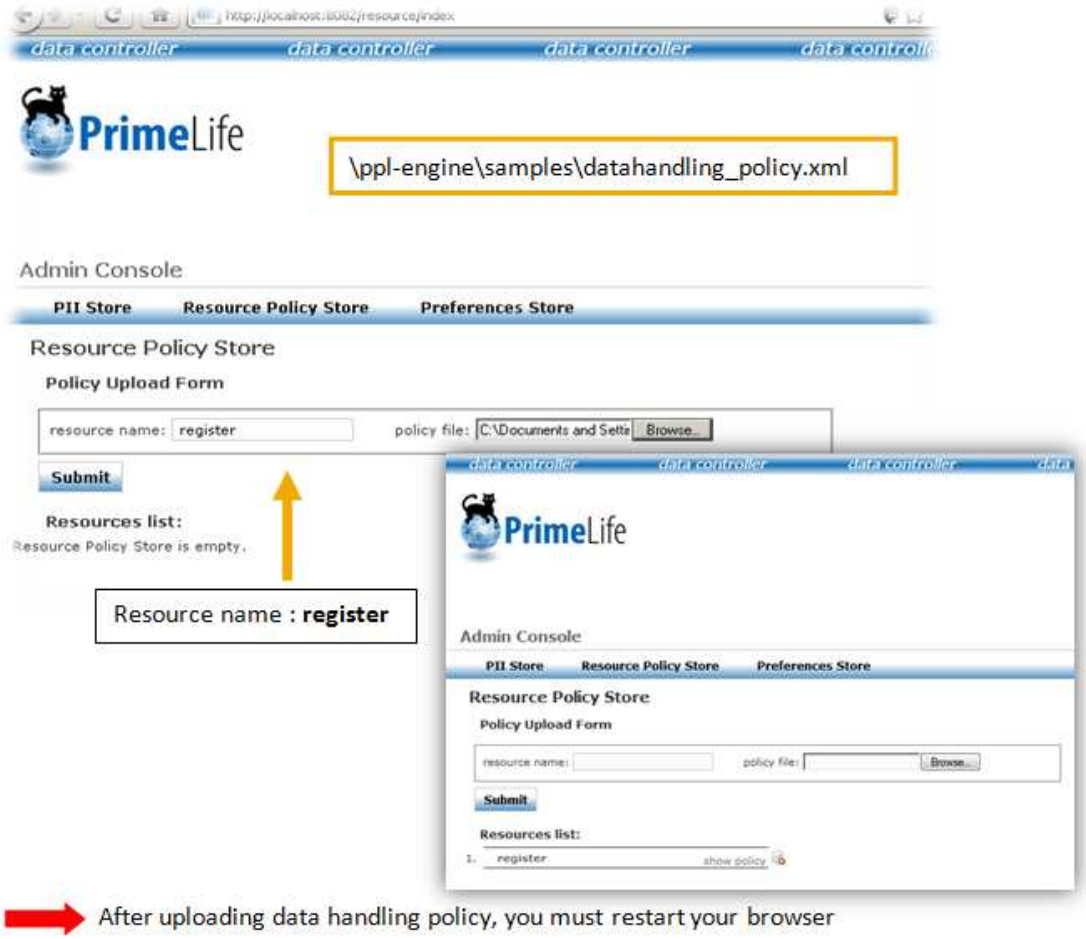

Figure 10: Configuring Privacy Policy

4.3 Registering to the Clique Website

At this point of the demo everything must be functional. Please **re-start your browser** and go to the Clique website on http://localhost:8082/clique/index.

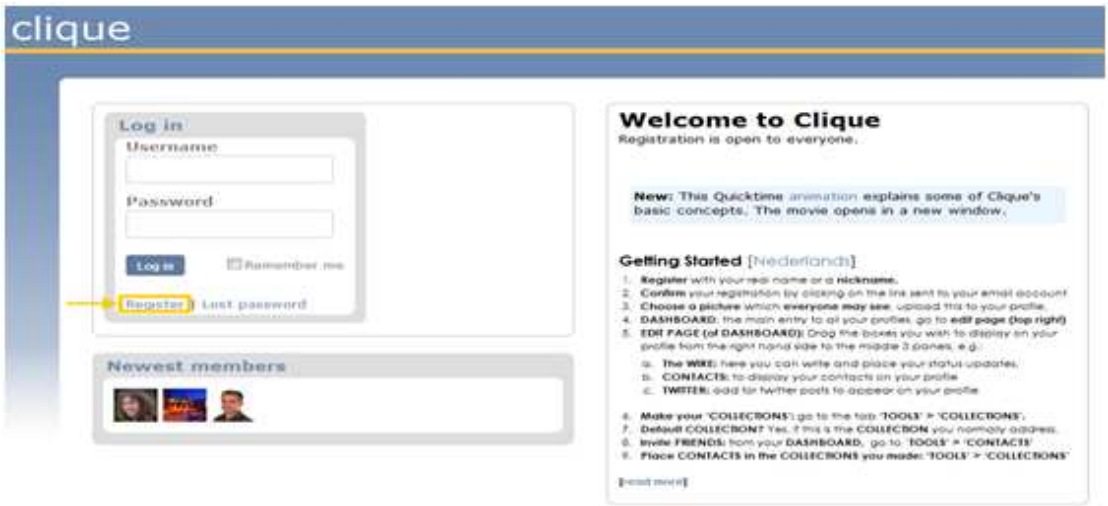

Figure 11: Clique welcome page

In the Welcome page (Figure 11) the user can create an account by clicking on the register button.

Automatically the browser will recognize that the website is compliant with the PPL engine and launch the matching plug-in (PolicyUI) as shown in Figure 12. This dialog box shows to the user the result of the matching between the his preferences and clique's privacy policy. The result of the matching is then used as a sticky policy. In case perfect matching the use has just to send his data with the sticky policy.

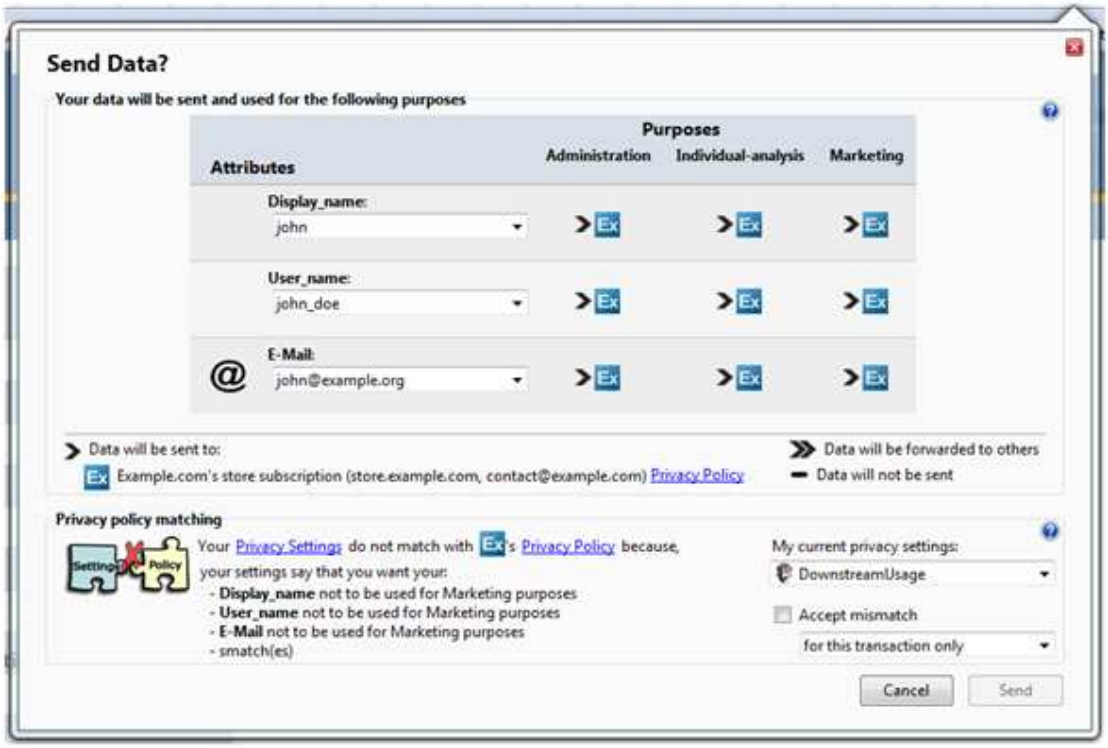

Figure 12: Matching dialog box

If there is any mismatch, the dialog box will display the list of mismatching elements in order to inform the user. Then the Data subject has to decide whether he accepts to send the data or not as shown in Figure 13.

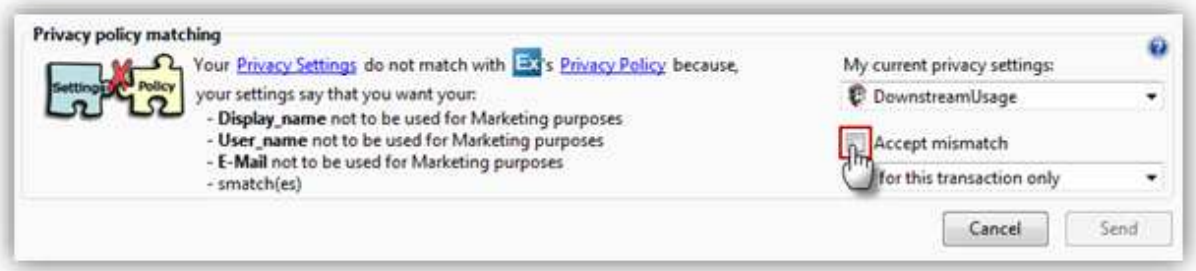

Figure 13: mismatching result

If the registration is successful the Data Subject can click on the list of PIIs he sent to the website and check the sticky policies that are affected to the data. See Figure 14.

Figure 14: Registration to Clique

An admin console is also available on the Data Controller side in order to access the list of collected PIIs (PII Store menu). If you click on one of these PIIs you can access to a human readable version of the sticky policy as shown in Figure 15.

Figure 15: Admin Console

At any time for testing purpose, using the admin console, we can force the execution of a trigger by clicking on the (trigger) button. For example in Figure 16 we forced the logging obligation by simulating an access for “marketing” purpose. This access will then be logged in the bottom of the page on the “Logs” table.

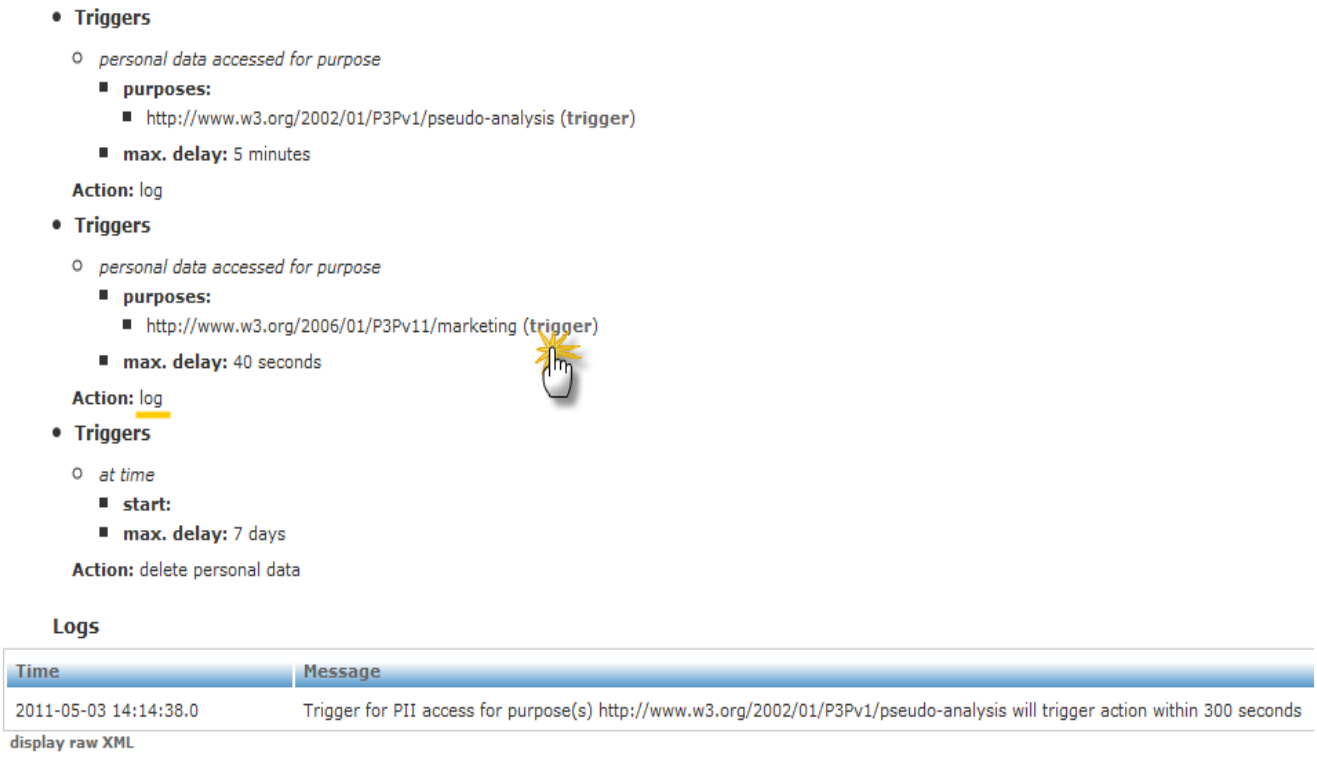

Figure 16: Forcing a trigger

4.4 Third Party Access

The third party Data Controller is running under this URL: http://localhost:8083/

When a third party Data Controller wants to collect some data from the Clique website, a request containing the required PIIs and their related data handling policy is sent to the initial Data Controller. In order to load this request, you have to browse your directory, select the correspondent XML request and send it as shown in Figure 17. We recommend you to use the request contained in $root \samples\request_third_party.xml.

The result is displayed as a list of PIIs like in the Data Controller console (see Figure 18). When you click on a PII to see the related sticky policy you will notice that this one is corresponding to the *downstreamusage* policy contained in the Data Subject’s preference.

On the Data Controller admin console you can also notice that the access of the third party triggered and obligation. This obligation logged the access of the third party as shown in Figure 19.

Figure 17: Third party request

Figure 18: Third party PII list

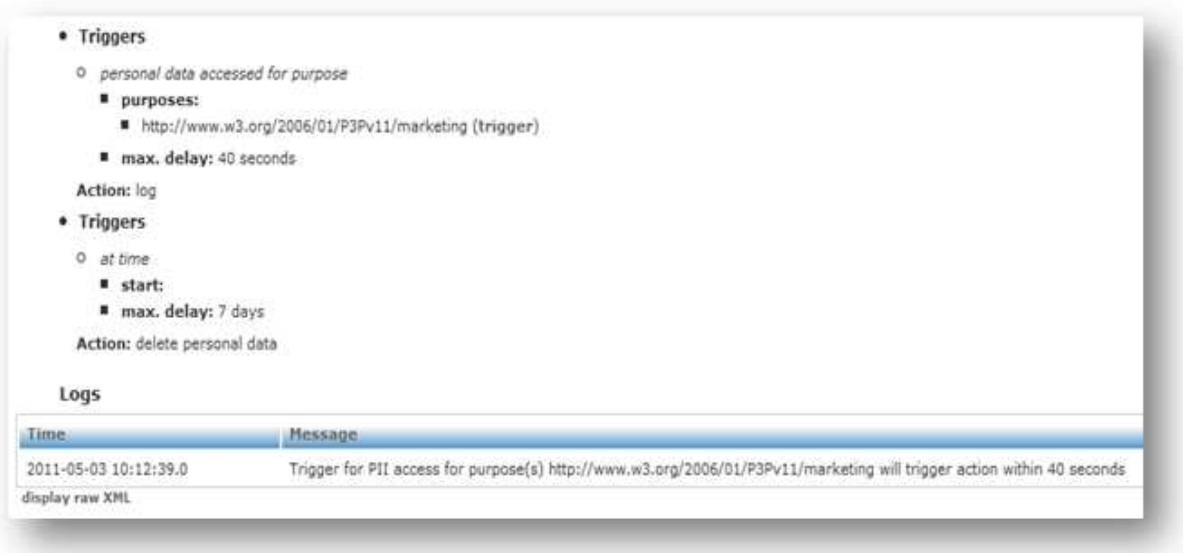

Figure 19: Obligation execution on the DC side

4.5 Preference Groups

As defined in [1] preference groups is a functionality that offers the possibility to maintain multiple browsing profiles with a different privacy settings. For example the "secure" profile which is not allowing any PII to be revealed or "trusted" profile which contains a list of trusted websites that are allowed to collect user's data. Different preferences can then be added through the interface described in Figure 9. Each of these preferences has to use different names. When the matching dialog box we can at any time change the current navigation preference group as shown in Figure 20. The matching result will automatically change with the selected preference groups.

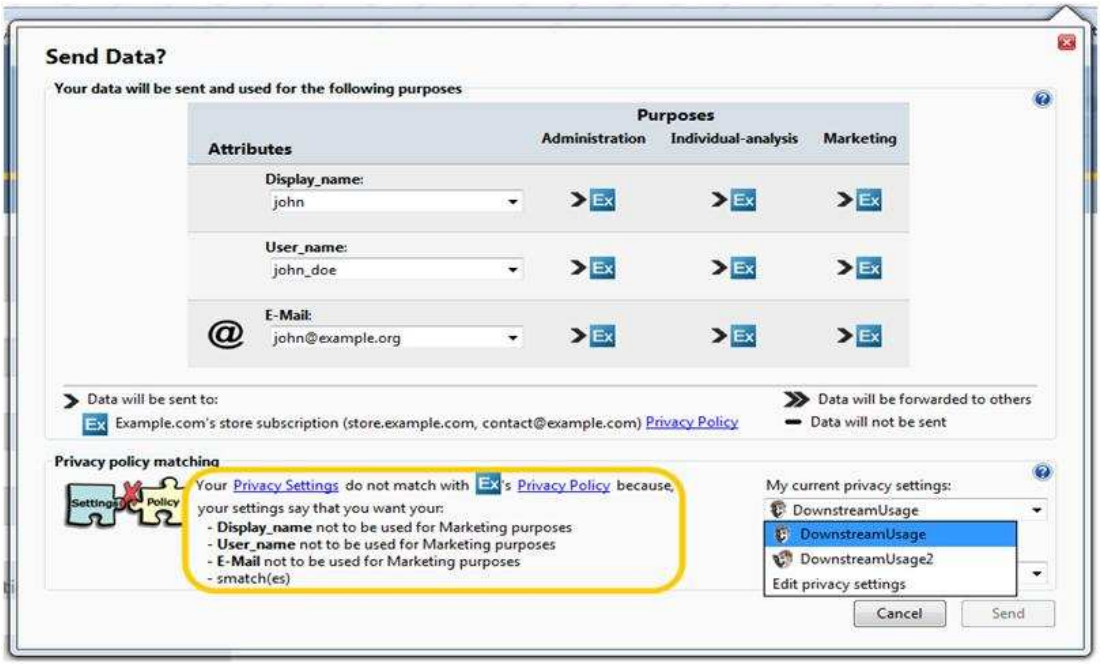

Figure 20: Preference groups

4.6 Configuring the e-mail Notification Obligation

In the list of supported obligation actions we propose the e-mail notification. For example whenever a PII is accessed for any purpose a notification e-mail is sent to the Data Subject. In order to enable such obligation you have to configure the SMTP e-mail server that will send the mail.

For this open the $root\server.jar with WinRAR archiver, open the file systemconfig.txt, and fill the parameters (as explained in Figure 21):

mail.smtp.host : Set SMTP server address

mail.smtp.port : set SMTP port :

from : Set the FROM address

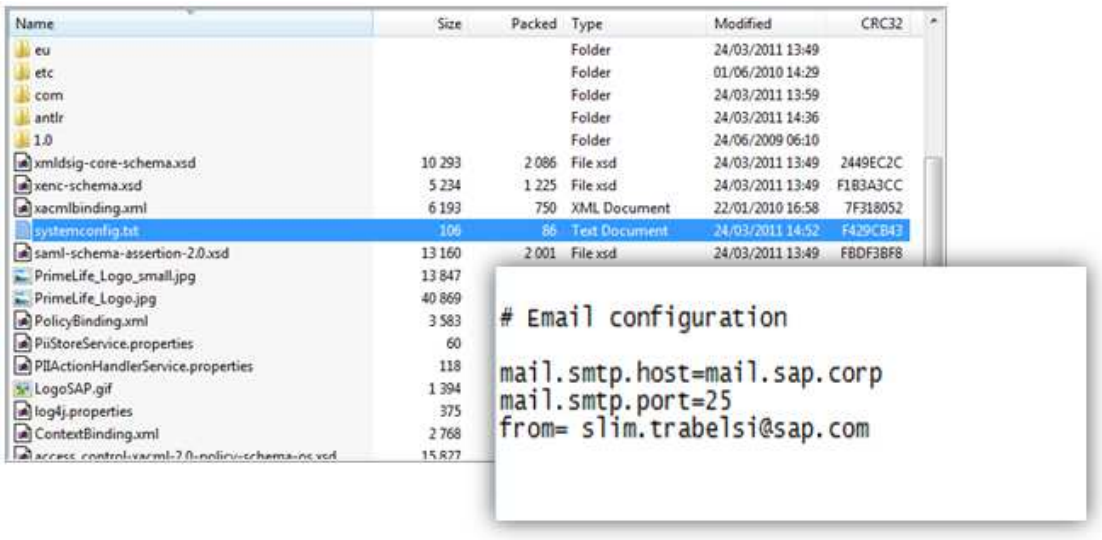

Figure 21: e-mail Obligation configuration

This notification e-mail can then be added to the preference as shown in the example Figure 22

```
<ob:Obligation>

    <ob:TriggersSet>
        <ob:TriggerPersonalDataAccessedForPurpose xmlns="http://www.primelife.eu/PPL/obligation">
            <ppl:Purpose>http://www.w3.org/2002/01/P3Pv1/contact</ppl:Purpose>
            <ob:MaxDelay>
                <ob:Duration>P0Y0M0DT0H5M0S</ob:Duration>
            </ob:MaxDelay>
        </ob:TriggerPersonalDataAccessedForPurpose>
        <ob:TriggerPersonalDataAccessedForPurpose xmlns="http://www.primelife.eu/PPL/obligation">
            <ppl:Purpose>http://www.w3.org/2006/01/P3Pv11/delivery</ppl:Purpose>
            <ob:MaxDelay>
                <ob:Duration>P0Y0M0DT0H15M0S</ob:Duration>
            </ob:MaxDelay>
        </ob:TriggerPersonalDataAccessedForPurpose>
        <ob:TriggerPersonalDataAccessedForPurpose xmlns="http://www.primelife.eu/PPL/obligation">
            <ppl:Purpose>http://www.w3.org/2002/01/P3Pv1/pseudo-analysis</ppl:Purpose>
            <ob:MaxDelay>
                <ob:Duration>P0Y0M0DT0H0M30S</ob:Duration>
            </ob:MaxDelay>
        </ob:TriggerPersonalDataAccessedForPurpose>
    </ob:TriggersSet>

 <ob:ActionNotifyDataSubject>
     <ob:Media>Email</ob:Media>
     <ob:Address>john@example.com</ob:Address>
 </ob:ActionNotifyDataSubject>

</ob:Obligation>
```

Figure 22: Adding the e-mail notification in the preference

References

[1] PrimeLife Deliverable D5.3.4 Report on design and implementation

[2] eXtensible Access Control Markup Language (XACML) Version 3.0, April 2009. See http://www.oasis-open.org/committees/document.php?document_id=32425

[3] Holistic Enterprise-Ready Application Security http://www.herasaf.org/heras-af-xacml.html

[4] PrimeLife Deliverable D4.3.2 UI Prototypes: Policy Administration and Presentation – Version 2