Design and implementation of a trust policy decision point (Trust PDP)

Lesky Deni Saputra Anitas, L.

Award date:
2010
MASTER’S THESIS

DESIGN AND IMPLEMENTATION

OF

A TRUST POLICY DECISION POINT

(TRUST PDP)

By

Lesky Deni Saputra Anatias

0641652

Tutors:

dr. Nicola Zannone

Daniel Trivellato M.Sc.

Supervisor:

dr. J.I. (Jerry) den Hartog

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
AT
EINDHOVEN UNIVERSITY OF TECHNOLOGY
EINDHOVEN
2009
To my beloved parents, my sister and my beloved wife, Nelly Martin. Without their patience, understanding and support, the completion of this work would not have been possible.
# Table of Contents

Table of Contents iii  
List of Tables vii  
List of Figures viii  
Glossary xi  
Abstract xii  
Acknowledgements xiii  

1 Introduction 1  
  1.1 Motivation ........................................... 1  
  1.2 Thesis Objective .................................... 2  
  1.3 Methodology ........................................ 3  
  1.4 Contribution ........................................ 4  
  1.5 Thesis Structure .................................... 4  

2 Background 6  
  2.1 Overview ............................................. 6  
  2.2 Trust Management .................................... 7  
     2.2.1 Structural Trust Management .................... 7  
        2.2.1.1 POLIPO CTM .................................. 8  
        2.2.1.2 SAML Attribute Query and Assertion ....... 9  
        2.2.1.3 Fedora Repository Services ................. 10  
     2.2.2 Behavioral Trust Management ................... 12  
        2.2.2.1 Centrality RTM ............................... 13  
  2.3 Trust Management in Healthcare Domain ............... 14
2.4 The Trust Management Architecture of TAS Project  
   2.4.1 Trust Policy Architecture  
   2.4.2 The Trust Policy Language  
      2.4.2.1 Type1 Trust Policy Statement  
      2.4.2.2 Type2 Trust Policy Statement  
      2.4.2.3 Type3 Trust Policy Statement  
2.5 Approaches to Trust Management Integration  
2.6 XACML  
   2.6.1 Combining Algorithms  
   2.6.2 XACML in Healthcare Domain  
   2.6.3 Sun’s XACML Implementation  

3 Scenarios  
   3.1 Overview  
   3.2 Home Telemonitoring Program for Patients with Chronic Heart Failure  
   3.3 Emergency Scenario  

4 Requirements  
   4.1 Overview  
   4.2 The Trust PDP Interface  
   4.3 The Trust Service Interface  

5 Design  
   5.1 Overview  
   5.2 Architectural Requirements  
   5.3 Architectural Design  

6 Implementation  
   6.1 Design Specification  
      6.1.1 Notation  
      6.1.2 Terminology  
      6.1.3 XACML Profile for Multiple PDPs  
         6.1.3.1 Policies  
         6.1.3.2 Example of Master PDP <PolicySet> for the Nurse Role  
      6.1.4 XACML Profile of Trust PDP  
         6.1.4.1 Subject Attribute Roles  
         6.1.4.2 Policies  
         6.1.4.3 Rules and Functions
List of Tables

2.1 Example of reputation feedback ............................................ 14
6.1 Mapping SAML elements to FOXML DC Metadata Element .......... 70
7.1 Comparison of TM integration solutions ................................. 81
G.1 The Trust PDP required components ................................. 121
G.2 The Centrality RTM trust service required components .............. 122
G.3 The POLIPO CTM trust service required components .............. 123
List of Figures

2.1 Fedora Repository services [LPSW06] ............................... 11
2.2 XACML data flow diagram [OAS05a] ............................... 21

3.1 Home telemonitoring solution from Phillips [SEP06]. .............. 30
3.2 Home telemonitoring program for patients with Chronic Heart Failure (CHF) ............................................ 31
3.3 Emergency scenario for home telemonitoring program for patients with Chronic Heart Failure (CHF) ............................................ 34
3.4 Trust evaluation flow for Bob’s trust policy ............................................ 35

5.1 Extended XACML architecture [OAS05a] in the TAS³ trust policy architecture settings ............................................ 46

6.1 Trust PDP component diagram ............................................ 64
6.2 Trust PDP class diagram ............................................ 65
6.3 POLIPO CTM component diagram ............................................ 67
6.4 Credential release ground query sequence diagram in POLIPO CTM client ............................................ 68
6.5 Credential release subject-var and attribute-var sequence diagram in POLIPO CTM client ............................................ 69
6.6 Centrality RTM component diagram ............................................ 71
6.7 Reputation sequence diagram in Centrality RTM client ............................................ 71
6.8 Graphical User Interface (GUI) for testing trust evaluation ............................................ 72
## Glossary

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>Attribute Authority</td>
</tr>
<tr>
<td>CHF</td>
<td>Chronic Heart Failure</td>
</tr>
<tr>
<td>CTM</td>
<td>Credential-based Trust Management</td>
</tr>
<tr>
<td>DCMES</td>
<td>Dublin Core Metadata Element Set</td>
</tr>
<tr>
<td>DCMI</td>
<td>Dublin Core Metadata Initiative</td>
</tr>
<tr>
<td>FOXML</td>
<td>Fedora Object XML</td>
</tr>
<tr>
<td>HCP</td>
<td>Health Care Provider</td>
</tr>
<tr>
<td>KPITM</td>
<td>Key Performance Indicator TM</td>
</tr>
<tr>
<td>Master PDP</td>
<td>Master Policy Decision Point</td>
</tr>
<tr>
<td>MERC</td>
<td>Medical Emergency Response Center</td>
</tr>
<tr>
<td>METS</td>
<td>Metadata Encoding and Transmission Format</td>
</tr>
<tr>
<td>MOP</td>
<td>Medical Observation Provider</td>
</tr>
<tr>
<td>MSP</td>
<td>Medical Service Provider</td>
</tr>
<tr>
<td>PAP</td>
<td>Policy Administration Point</td>
</tr>
<tr>
<td>Glossary</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>PDP</td>
<td>Policy Decision Point</td>
</tr>
<tr>
<td>PEP</td>
<td>Policy Enforcement Point</td>
</tr>
<tr>
<td>PHRP</td>
<td>Personal Health Record Profile</td>
</tr>
<tr>
<td>PIP</td>
<td>Policy Information Point</td>
</tr>
<tr>
<td>POLIPO</td>
<td>Policies &amp; OntoLogies for Interoperability, Portability, and autOnomy</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>RTM</td>
<td>Reputation-based Trust Management</td>
</tr>
<tr>
<td>SAML</td>
<td>Security Assertion Markup Language</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>TAS³</td>
<td>Trusted Architecture for Securely Shared Services</td>
</tr>
<tr>
<td>TM</td>
<td>Trust Management</td>
</tr>
<tr>
<td>Trust PDP</td>
<td>Trust Policy Decision Point</td>
</tr>
<tr>
<td>TXACML</td>
<td>Trust XACML</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>XACML</td>
<td>eXtensible Access Control Markup Language</td>
</tr>
</tbody>
</table>
Abstract

We analyze two approaches of modeling trust, a POLIPO Credential-based Trust Management (CTM) and Reputation Trust Management (RTM) and argue that an integrated approach would improve the flexibility and extensibility of trust management systems. To create this integrated approach we apply the concept of a Policy Decision Point (PDP) also to establishing trust. A Trust PDP is designed which provides the system with a unified interface to a broad range of trust services. This design is validated by implementing a prototype of the Trust PDP.
Acknowledgements

I am heartily thankful to my supervisor, dr. Jerry den Hartog, for giving me an opportunity to carry out the research at University of Technology Eindhoven, The Netherlands and for his excellent guidance and comments throughout this thesis from the initial to the final level that enabled me to develop an understanding of the subject. I am also very thankful to dr. Nicola Zannone from University of Technology Eindhoven for his vital input throughout this thesis.

I would also like to thank Daniel Trivellato for his support in the initial phase of this thesis. Furthermore, I am also very grateful to dr. Marc Voorhoeve from University of Technology Eindhoven for taking part in my assessment committee.

Lastly, I offer my regards and blessings to my family, friends, and colleagues at University of Technology Eindhoven for their utmost support during the completion of the project.

Lesky Deni Saputra Anatias
Eindhoven, The Netherlands
December 2009
Chapter 1

Introduction

This chapter describes the motivation behind this thesis, the scope of this work, the methodology that has been used in this project and its contribution. This chapter ends with an overview of the thesis structure.

1.1 Motivation

Heart failure is a growing public health issue. In line with most countries in Europe, the incidence and prevalence of heart failure in The Netherlands are growing and heart failure continues to be a significant cause of hospital admissions [JHRSV05]. In USA, the estimated direct and indirect cost of heart failure for 2008 is more than 34 billion dollars, of which approximately 2/3 are spent on readmissions to hospital [RFF+08, CPR+04], while in The Netherlands costs related to cardiovascular diseases amounted to 10.1% of the total costs of the Dutch Healthcare system in 1999, of which 28% was related to stroke, 26% to ischemic heart disease, and 8% to heart failure [JHRSV05].

One of the approaches to answer this problem is by adopting a home telemonitoring care program for patients with chronic heart failure (CHF). This model is argued by [CPR+04] as a more effective solution towards CHF treatment than the usual care in reducing health care demands for CHF patients. Furthermore, this model has produced greater inter-dependence between patient and health care practitioner. This inter-dependency is particularly important in the management of many chronic diseases where success depends on changes that the patient can make which requires a partnership and trust between patient and health care practitioner [RC06]. As the new treatment encourages the use of different services and technology provided by different service providers, the need of trust is also increasing. For example, in home telemonitoring program for CHF patients, the service may be involving the following professionals: general practitioner (GP), cardiologist, heart failure nurse,
home care, internist, dietician, pharmacist, social worker, psychologist, physical therapist, geriatrician, and nurse practitioner in GP [JHRSV05] and devices: sensors and wearable technologies to provide patients vital signs and symptoms (weight, systolic blood pressure, heart rate, dyspnoea, asthenia, oedema, therapy changes, blood urea nitrogen, creatinine, sodium, potassium, bilirubine) [JHRSV05, SEP06].

Without trust realization between patients and different service providers in the home telemonitoring care program setting, the patients may well not access the service at all and disclose all medically relevant information. Therefore, trust in this healthcare management setting is important.

The Trusted Architecture for Securely Shared Services or TAS\textsuperscript{3} framework of Trust Management Architecture Design proposes a suitable approach in implementing a scalable and generic way of trust establishment amongst users, service providers and information repositories [TAS]. While the TAS\textsuperscript{3} framework is emphasizing the architecture within the employment context, this thesis presents the analysis in healthcare setting.

The healthcare setting relies on personally identifiable information. Consequently, the users in this setting must have the possibility to select a wide range of trust policies to build a trust [TAS]. The trust build in this particular context might be based on structural information using credentials or behavioral information based on the reputation of participants. For example, a patient trusts a medical treatment for her disease because it is given by a certified health practitioner (e.g. cardiologist) and based on the fact that the trustee has a good reputation among patients in her hospital.

1.2 Thesis Objective

The main focus of this thesis is to combine different approaches to trust management, namely structural, behavioral and novel trust managements, into integrated solution which is easy to scale up and has a flexible design. This solution should also answer the following problems:

1. What are the different notions of trust?
2. What are the approaches to combine different sources of trust?
3. Which combining approach that is used in this thesis?
4. Which language that is used to express different trust statements from different sources of trust?
In order to provide the integrated solution, this thesis is aimed in designing and implementing a system with a unified interface to a broad range of trust services which we called a Trust Policy Decision Point (Trust PDP).

1.3 Methodology

The interface is applying a concept of Policy Decision Point (PDP) which act as an entity that evaluates applicable set of policies and renders an authorization decision [OAS05a]. While the aim of an authorization PDP is to produce an authorization decision based on authorization policies, a Trust PDP is designed to provide a trust decision based on trust policies.

A trust policy can be expressed in three different ways: verbally unstructured, structured using schemata, or formally modelled. For inter-operability reasons, a policy must be formulated and encoded in a way enabling its correct usage and interpretation. Therefore, policies have to be constrained regarding syntax, semantics, vocabulary, and operation of policy documents. A common way to express constraints is to use the specification of user-defined schemata such as XML schemata or a formal model as a way to express the trust policy [Blo04]. For the aforementioned purpose, a standardized policy schema based on OASIS eXtensible Access Control Markup Language (XACML) specification along with a subset model of a formalized XACML is used in this project. This policy defines a set of rules, an identifier for the combining algorithms and (optionally) a set of obligations [OAS05a] along with trust metrics define by each applicable trust service. In addition, the trust policy must implement the policy language specified by TAS\textsuperscript{3} framework of Trust Management Architecture Design [TAS].

For designing and implementing a Trust PDP, the following principles of software engineering and different types of Unified Modeling Language (UML) diagrams [UML] are being used to specify the Trust PDP architecture:

**Separation of Concerns.** This principle is applied to each Trust PDP component. The Trust PDP architecture consist of a Trust PDP component and trust services. Each trust service has a responsibility to provide a unique functionality that can be used by another trust service and acted as an independent entity which can be modified or replaced without disturbing the whole system and compromising other trust services. The design is also applying modularity and abstraction principle as a specialization of the principle of separation of concerns. With modularization, the Trust PDP and trust services define their own interfaces, while the abstraction principle indicates that all functionality must be accessed through these interfaces. Thus separating the behavior of the components from their implementation.
Extensibility. A system design principle where the implementation of the Trust PDP takes into consideration its future growth. Since only the Trust PDP has to be aware of the available trust services, the addition of new functionality can be achieved without modification of existing trust services. The idea of this design is to provide for change while minimizing impact to existing system functions.

Class Diagram and Component Diagram. These diagrams are used to describe the structure of Trust PDP and trust services by showing their attributes, the relationships between classes and the dependency between components.

Sequence Diagram. This diagram is used to show the interaction within trust services in terms of a sequence of messages. It also uses to indicate the lifespan of each object.

Furthermore, the healthcare scenarios and system requirements based on [TAS] are used as a starting point for designing the Trust PDP. Existing implementation, such as SunXACML [Sun] and OpenSAML [Opea] are investigated as building blocks for the implementation of the Trust PDP architecture. Finally a demonstrator has been defined and implemented. The implementation is based on the predefined architecture and provides a proof of concept.

1.4 Contribution

Our approach shows a capability to evaluate trust decisions from different trust services and to combine these decisions using the policy language specified by [TAS]. In addition, the Trust PDP system also provides the following features:

1. The Trust PDP interface is able to express trust metrics from different trust services using trust policy specification defined by OASIS eXtensible Access Control Markup Language (XACML) standard.

2. The Trust PDP system has an extensibility feature that conforms to the TAS requirements [TAS], so that the addition of new trust services can be achieved without interrupting the whole system and modification of existing trust services.

1.5 Thesis Structure

The rest of the thesis is organized as follows. We introduce the state-of-the-art of the trust management, the use of trust management in healthcare domain and the
works related to different approaches in trust management integration in Chapter 2. Chapter 3 describes a trust establishment in home telemonitoring program where an emergency scenario in this setting is being used to illustrate the need to combine different sources of trust. From this scenario, in Chapter 4, the approach for designing and implementing the Trust PDP are determined by defining the requirements of the system. Chapter 5 covers the design of the Trust PDP as an approach to trust management integration. Chapter 6 presents the implementation of the Trust PDP and Chapter 7 validates the Trust PDP through requirement fulfilment analysis and evaluation of the system. Finally, Chapter 8 presents the conclusion.
Chapter 2

Background

This chapter describes the ongoing research on trust management, namely structural and behavioral trust management. This chapter also presents the use of trust management in healthcare domain, the Trust Management Architecture of TAS$^3$ Project [TAS] which describes the policy architecture and the policy language that are used in TAS$^3$ project, approaches to trust management integration and the language specifications which have been used to implement the approaches. This chapter also describes the use of other supporting tools, namely SAML attribute query and assertion and Fedora Repository service which serves as a digital repository for credentials received from a structural trust management service. This chapter ends with an explanation of XACML as a policy language specified in this thesis.

2.1 Overview

Trust is an integral component in many kinds of human interaction, allowing people to act under uncertainty and with the risk of negative consequences [AG07]. However, trust is not only important to the human users but also software agents and increasingly, the machines that provide services which need to be trusted in various applications and situations. These services are often protected using various traditional security mechanisms.

In traditional security mechanisms, the security measurements typically protect resources from malicious users, by restricting access to only authorized users. However, in many situations we have to protect ourselves from those who offer services, for example a service provider can act deceitfully by providing false or misleading service, and traditional security mechanisms are unable to protect against this type of threat [JIB07]. Therefore, it raises the need of trust in both directions. In this context, trust can be used to protect data, to find accurate information, to get the best quality service, and even to assist other trust evaluations [AG07]. The trust in
this setting also best describes as a provisional trust\cite{JIB07} since the relying party acts a user who is seeking protection from malicious or unreliable service providers. In this context, it is the utmost important to address how trust as a security measurement is able to protect users from malicious service providers. The following sections will explain how trust is used to convey security decisions from various sources in different settings and domains.

\section{Trust Management}

Trust management (TM) is an approach that is used to make decisions when we are about to interact with something or someone we do not completely know and to establish whether we should proceed with the interaction or not. In trust management system\cite{BFL96,LMW,AG07,JIB07}, the decision to be made is based on the policy statements of various principals and stored in distributed manner. The decision who can be trusted is taken not only by a single principal but by taking into account information from other principals. The key concept to this process is that the trust is delegated to another principal by transferring limited authority of resources. The delegation mechanism is particularly important in the presence of autonomous systems which typically operate in dynamic and decentralized environments.

To date, the interest in trust management techniques greatly increases from both the industrial and the academic sectors. As described in \cite{TAS}, the research in this area has also determined by a duality in the very definition of TM system which can lead to confusion. This duality reflected into two broad TM categories: credential-based TM (CTM) or structural TM and reputation-based TM (RTM) or behavioral TM. The two approaches have been developed within different settings and targeting different environments. The following two sections describe both approaches.

\subsection{Structural Trust Management}

Let us first illustrate a simple example of structural trust management.

Consider the situation where a healthcare provider offers free enrolment to any patient from accredited hospitals. In our scenario, Bob comes to enrol at the healthcare provider and claims he is entitled to the free enrolment. The healthcare provider will not have a list of all patients from all accredited hospitals. Instead it will have a delegation rule stating that the a particular government body are trusted to decide which are the accredited hospitals. Furthermore, the healthcare provider will not have a list of all patients in each accredited hospital. Instead it will delegate this decision to the accredited hospitals. In this way the healthcare provider can capture its policy in the following rule:
Example 2.1.

\[
\begin{align*}
  & \text{freeEnrolment}(eo, X) \leftarrow \text{accreditedHospital}(gov, Y), \text{patient}(Y, X). \\
  & \text{accreditedHospital}(gov, gr). \\
  & \text{patient}(gr, bob). 
\end{align*}
\]

This example says: if a particular government body states that \( Y \) is an accredited hospital and \( Y \) states that \( X \) is a patient then \( X \) is entitled to free enrolment.

The above illustration describes that the structural trust management involves the exchange or verification of credentials, which are information issued by one entity, and may describe qualities or features of another entity. For example, having the credential of a patient means its holder has been recognized as a patient by the issuing hospital. This associates the holder with the hospital and possibly its services. In addition, credentials are sometimes implemented using security certificates with digital signatures. A certificate can be used as a credential if it includes properties about an entity.

The previous example also implies a credential chain mechanism as mentioned in [LWM01, CEE+01]. Finding such chains of credentials is a central research topic in rule-based TM and is strongly related to the decisions on the storage location of credentials. Another important research topic in CTM is trust negotiation [Win02, BO05] which relates to the issue of protecting credentials themselves. This topic is important since credentials are often confidential (e.g., medical record or credit card information), and should not be automatically disclosed to anyone requesting them.

One of the structural trust management approaches is a POLIPO[TSZE09a] CTM which will be discussed in the following section.

2.2.1.1 POLIPO CTM

The Policies & OntoLogies for Interoperability, Portability, and autOnomy or POLIPO [TSZE09a, TSZE09b] presents a policy language and an architecture for policy evaluation that combines distributed access control with ontologies. The ontologies that are used in this context are to provide a means for establishing common vocabularies and capturing domain of knowledges.

The POLIPO policies[TSZE09a] are specified by using four types of basic constructs: (1) ontology atoms which are used to query the knowledge base represented by ontologies, (2) credential atoms, which represent digitally signed statements made by an issuer about an attribute of a subject, (3) authorization atoms which denote the permission of a subject to perform an action on an object, and constraints (e.g., =, >, <, etc.), (4) which are specified using Constraints Logic Programming (CLP). POLIPO distinguishes two types of policies: credential release policies and authorization policies. In this thesis, we focus on credential release policies in order to query
credentials provided by POLIPO CTM trust service. A credential release rule itself is a *Horn clause* where the head is a credential atom and the body can contain positive credential and ontology atoms, and constraints. Furthermore, the POLIPO policies are defined as sets of POLIPO rules described above. For illustrative purpose, the following example illustrates POLIPO credential atoms in POLIPO policy.

**Example 2.2.** The following is a POLIPO CTM policy:

```prolog
% Directorate−General (Government Body)
cred('DG', 'hospital', 'yellowHosp').
cred('DG', 'hospital', 'greenHosp').

% Green Hospital Policy
cred('greenHosp', 'doctor', 'Julie').
cred('greenHosp', 'nurse', 'Alice').

% MSP−1 Policy
cred('MSP1', 'firstAidOfficer', X) :- cred('DG', 'hospital', Y), cred(Y, 'doctor', X).
cred('MSP1', 'firstAidOfficer', X) :- cred('DG', 'hospital', Y), cred(Y, 'nurse', X).
```

The current POLIPO CTM trust service prototype provides a Java interface which corresponds to SAML standard and non-standard attribute queries. The credentials are returned in the form of an SAML response containing one or more assertions, each of which corresponds to the credential atoms. These credentials will then be stored in the persistent repository storage for later purpose. In this setting we utilize Fedora Repository service as our credential repository service. Both SAML attribute query and assertion, and Fedora Repository are incorporated into the POLIPO CTM trust service which will be explained in the following two sections.

### 2.2.1.2 SAML Attribute Query and Assertion

The Security Assertion Markup Language or SAML is an XML-based OASIS standard for communicating user authentication, entitlement and attribute information [OAS05e]. SAML is used to exchange assertions between entities. In the process of exchanging such assertion, SAML defines its own profiles and bindings. SAML profile is used to specify how SAML is employed in any given context. In the other hand, SAML binding is used to specify how SAML assertions and/or protocol messages are transmitted over another protocol. Typically, a SAML profile specifies the SAML bindings that may be used in its context. SAML also defines a request and response protocol for obtaining assertions [OAS05e].

A typical SAML assertion consists of the following information: (1) issuer, which explains who issued the assertion, when was it issued and the assertion identifier, (2)
subject information, which explains the name of the subject, the security domain and optional subject information, (3) conditions under which the assertion is valid, such as validity period, audience restriction and target restriction.

In case a SAML assertion contains a SAML attribute statement, the interpretation of the assertion goes as follows: an issuing authority is asserting that the subject with a specific name identifier is associated with certain attributes with a specific attribute values. For example, user Bob is associated with the attribute Position, which has the value *Patient*.

In POLIPO CTM trust service setting, the SAML assertions are received as a result of SAML attribute query. For example, a ground query for the statement: Trust PDP asks an entity (the recipient of the request) if Bob has attribute doctor will be translated in POLIPO as `cred(X, 'Physician', 'Bob')`, where X (the issuer of a credential) is bound to the entity receiving the request. This query shall return Yes or No answer together with the credential (if the answer is yes). A full example of credential release queries in POLIPO CTM trust service is shown in Appendix D.

To implement SAML attribute query and assertion in POLIPO CTM trust service, we utilize the OpenSAML [Opea] implementation with the following libraries:

**The OpenSAML Core Library.** It provides `org.opensaml.saml2.core` package which contains various interfaces for SAML 2.0 core specification types and elements. This package features several objects, such as `AttributeQuery` object, `Response` Object, `Assertion` object, and several other objects that are used in the POLIPO CTM trust service.

**The XMLTooling library**, contained within OpenSAML. It provides the ability to work with XML.

For more complete list of the APIs, the reader could visit OpenSAML JavaDocs\(^1\).

### 2.2.1.3 Fedora Repository Services

A Fedora Repository [Fed09] provides a general-purpose management layer for digital objects and their containers that aggregate local and distributed content into digital objects (e.g. digital images, XML files, metadata) and the association of services with objects. The Fedora Repository includes a generic Resource Description Framework (RDF) relationship model that represents relationships among objects and their components. Queries against these relationships are supported by an RDF triple store. The Fedora Repository architecture is implemented as a web service, with all aspects of the digital objects and related management and access functions exposed through Representational State Transfer (REST) and Simple Object Access Protocol (SOAP).

\(^{1}\)http://www.opensaml.org/docs/2.1.0/apidocs/
interfaces [LPSW06]. As shown in Figure 2.1, the internal modules and public service interfaces of the Fedora Repository service are made visible through its four web service interfaces.

In order to incorporate the Fedora Repository service into the POLIPO CTM trust service, we use the Fedora REST API\(^2\) that exposes a subset of the Fedora access and management APIs as a RESTful (Representational State Transfer) web service. The typical management and access operations that are used in this purpose are: \texttt{ingest/export/purgeObject}, \texttt{add/modify/purgeDatastream}, \texttt{listDatastreams}, and \texttt{findObjects}. The operations for ingesting and exporting digital objects are used in an XML format with either Fedora Object XML (FOXML), or alternatively Metadata Encoding and Transmission Format (METS), and querying objects will result in a list of object properties and the Dublin Core metadata elements.

The Dublin Core metadata element set, issued by the Dublin Core Metadata Initiative (DCMI)[Dub09] is used to describe digital objects such as video, sound, image, text, and composite media like web pages. The Dublin Core standard includes two different levels, the Simple Dublin Core Metadata Element Set (DCMES) which

\(^2\)\url{http://www.fedora-commons.org/confluence/display/FCR30/REST+API}
consists of 15 metadata elements as follows: (1) Title, (2) Creator, (3) Subject, (4) Description, (5) Publisher, (6) Contributor, (7) Date, (8) Type, (9) Format, (10) Identifier, (11) Source, (12) Language, (13) Relation, (14) Coverage, and (15) Rights, and Qualified Dublin Core which consists of three additional elements, namely (1) Audience, (2) Provenance and (3) RightsHolder.

As mentioned before, the Fedora digital objects can be stored and exported using FOXML. Note that, the FOXML schema defines a `<digitalObject>` root element that contains a set of `<objectProperties>`, one or more `<datastream>` components, and one or more `<disseminator>` components. Although FOXML is the preferred XML serialization format for storing objects in a Fedora Repository, Fedora supports ingest and export of digital objects in other XML formats as well [LPSW06].

After discussing structural trust management and its corresponding CTM trust service that is used in this thesis, we will continue to explain another trust management approach, namely behavioral trust management, in the following section.

### 2.2.2 Behavioral Trust Management

Let us first illustrate a simple example of behavioral trust management.

Consider the situation where Bob, as a patient has already registered to a healthcare provider. Registering to a healthcare provider entails that Bob will get the medical treatment or medical assistant from medical service provider provided by the healthcare provider, however since Bob does not have personal experience with the provider and wants his personal information to be safe. Bob can use a reputation-based system by asking other patients he trusts to provide feedback on providers and to recommend the providers that they trust. Any provider with sufficiently strong recommendations is seen as trustworthy.

Based on the above illustration, Bob incorporates reputation concept to develop trust relation with an unknown party, since Bob has believed in another person’s character or standing [JIB07]. Therefore, we can say that reputation can be used to naturally supports the process of building trust among people [RKZF00]. The reputation system mentions in the example produces entity reputation score that can been seen by whole community, reflects transitivity trust in implicit way and uses rating as input. The reputation level built from this system is made by an assessment based on the history of interactions with or observations of another entity, either directly (by personal experience) or as recommended by others.

The research on reputation-based systems has already matured and contributed to an understanding of how to incorporate trust relations between human users, software agents and machines. Various reputation systems are also used in commercial purpose. For example, eBay auction system. In eBay, every user is welcome to leave a positive, negative or neutral feedback after each transaction. Sellers and buyers in eBay can
rate each other. One of the important studies in reputation-based trust management is addressing the problems of data quality in peer-to-peer (P2P) networks [PBMW99, KSGM03]. For example, the EigenTrust algorithm [KSGM03] computes a global reputation value (using PageRank) for each entity in the networks.

Beside the benefits, problems and proposed solutions in reputation systems, such as unfair ratings, low incentive for providing rating, bias toward positive rating, change of identities, quality variations over time, discrimination and ballot box stuffing (too many unfair ratings) has been described in [JIB07].

One of the behavioral trust management approaches is a Centrality RTM[TAS] which will be discussed in the following section.

2.2.2.1 Centrality RTM

TAS\(^3\) introduces reputation-based trust service based on different centrality measurement [TAS]. This flexibility is provided by the trust service to adjust trust values in response to (feedback on) events using different measurements, for example, in this work, we mainly focus on PageRank algorithm[PBMW99] as the measurement. In addition, this approach implements a centralized solution of reputation systems.

In this centralized reputation system, information about the performance of a given entity is collected as ratings from other members in the community who have had direct experience with that entity. The reputation center that collects all the ratings typically derives a reputation score for every participant, and makes all scores publicly available. Participants can then use each other’s scores, for example, when deciding whether or not to interact with an unknown party.

This Centrality RTM trust service defines the following centrality operator and feedback format, which acts as trust metrics for the trust policy specification. Detailed information on characteristics that is defined in the feedback format are shown in [TAS].

\[
\text{CENTRALITY}[^{\text{name, } A_v, A_s, A_t, A_w, centrality\text{measure}}](R_{\text{vertices}}, R_{\text{edges}})
\]

The attribute name determines how the result column is called. The attribute \(A_v\) specifies the column of the relation \(R_{\text{vertices}}\) that contains the vertices. The attributes \(A_s, A_t,\) and \(A_w\) specify the columns of the relation \(R_{\text{edges}}\) that contain the start and target vertices of the edges as well as the edge weights. Finally, the attribute measure specifies the centrality measure to be used.

The following example illustrates the reputation feedback in the Centrality RTM trust service:

**Example 2.3.** This example say Alice: *I am quite sure that the quality of service S by Bob was good. It was a complex problem.* This informal rating has to be translated
into the formal representation described above. The resulting feedback tuple could look as follows:

<table>
<thead>
<tr>
<th>Rater</th>
<th>Ratee</th>
<th>Value</th>
<th>Context</th>
<th>Facet</th>
<th>Time</th>
<th>Certainty</th>
<th>Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Bob</td>
<td>0.95</td>
<td>S</td>
<td>Quality</td>
<td>12:01:45</td>
<td>0.75</td>
<td>0.8</td>
</tr>
</tbody>
</table>

For clarity, the following example provides an algebraic expression of the reputation trust service which acts as a reputation policy.

**Example 2.4.** The following is a reputation policy of the Centrality RTM trust service: *I trust individuals in context c and facet f, if their average feedback value from the 10 most reputable entities exceeds a specific threshold t.* Use the PageRank centrality measure to rank the entities.

\[
\text{PROJECTION}[\text{ratee}] (\text{SELECTION}[\text{avg.value}>t](\text{GROUP}[\text{avg.value}, \text{AVG(value)}, \{\text{ratee}\}] (\text{JOIN}[\text{rater}=\text{id}] (\text{TOP}[10, \text{pagerank}] (\text{CENTRALITY}[\text{pagerank, id, rater, ratee, value, PageRank}] (\text{Entity}))))) \]

\[
\text{SELECTION}[\text{context}=c, \text{facet}=f_c](\text{Feedback}))
\]

\[
\text{SELECTION}[\text{context}=c, \text{facet}=f_c](\text{Feedback})
\]

### 2.3 Trust Management in Healthcare Domain

In this section we describe how and why trust relations in the healthcare domain are substantially important for healthcare providers, institutions and patients that consume the services.

Trust in healthcare setting is an important aspect to be addressed. Trust has long been a foundation of an effective patient-doctor relationships in traditional healthcare interactions. It is important since the trust in this particular context relates to the information asymmetries arising from the specialist nature of medical knowledge and the uncertainty and risk regarding the competence and intentions of the medical service providers on whom the patient is dependent [RC06].

As the modern healthcare infrastructures move towards electronic healthcare system, trust, either interpersonal or institutional, is gaining more importance level in healthcare context and in consequence spurs the research in this area. Researchers in
this domain have been worked for various purposes [JIB07, vDKP08], such as en-
hancing ubiquitous healthcare [YGL08], and implementing access control for healthcare
infrastructure (see Section 2.6.2).

The following section will address the Trust Management Architecture of TAS\textsuperscript{3}
that works to address both structural and behavioral approaches, and also provides
a means of adding novel trust managements as proposed by [TAS]. The following
section will explain this approach.

\section{The Trust Management Architecture of TAS\textsuperscript{3} Project}

The Trusted Architecture for Securely Shared Services or TAS\textsuperscript{3} Trust Management
Architecture\textsuperscript{3} is an EU funded project that specifies a trusted services network which
tries to advance the current state of the art of the existing solutions. The overall
objective of TAS\textsuperscript{3} Architecture itself is to demonstrate that its architecture can deal
with the following challenges in a generic and scalable way:

1. User & service provider authentication & credential management.
2. Establishing trust between users, information repositories and service providers.
3. Data protection policies.
5. Demonstrator challenges.

Moreover, the Trust Management Architecture [TAS] strives to built a dependable
Trust & Security Architecture which offers safe and dependable business processes
environment for exchanging personal identifiable information, which is typically gen-
erated over a human lifetime, and collected, stored at distributed locations and used
in a multitude of business domain, in particular employability and electronic health.

The following two sections describe the trust policy architecture of TAS\textsuperscript{3} project
and the trust policy language proposed by [TAS].

\subsection{Trust Policy Architecture}

In this section we illustrate the trust policy architecture, as shown in [TAS] along
with the role of different components and their relations as follows:

\textsuperscript{3}TAS\textsuperscript{3} Project, available at http://www.tas3.eu/
The Policy Enforcement Point (PEP) uses a Master Policy Decision Point (Master PDP) to make its policy decisions which in turn calls sub PDPs, including a sub PDP specialized in answering trust policy queries. This Trust PDP provides the main interfaces to the credential and reputation components described in this chapter. The Trust PDP is responsible for evaluating the TAS\(^3\) trust policies. The Trust PDP itself combines the results of different trust services, similar to Master PDP combining its decisions. While the Trust PDP is the main interface between the trust policy management architecture and the rest of the TAS\(^3\) infrastructure, the individual trust services do rely on other (TAS\(^3\)) components. These dependencies are given in Section 2.2.1 and 2.2.2 which describes the trust services and their interfaces. The Trust PDP accepts trust evaluation requests from the Master PDP encoded in the form of an XACML request context. Finally, the Trust PDP returns a standard XACML response context conveying a yes, no, indeterminate or not applicable answer to the trust evaluation request [TAS]. [TAS] provides detail information on trust policy framework for overall design.

Beside the CTM and the RTM trust service, the Key Performance Indicator Trust Management (KPI TM) trust service is also introduced in [TAS] as a trusted factor provider which provides trust level evaluation based on predefined indicators. [TAS] describes the details on how KPITM trust level is computed for a given setting. The advantage of this service is that the correctness of ratings can be verified by others, or automatically generated based on automated monitoring of events. Since ratings in this trust service is based on objective criteria, it would be much simpler to achieve high system robustness. However, the KPITM would also face a challenge if, in the healthcare setting, the patients who use this service, are sceptical about the reality of performance figures in light of evidence of managers *gaming* the system to meet targets [RC06].

As described in [TAS], TAS\(^3\) will support passing policies from Master PDP to other specialized PDPs using the XACML elements. Including policies in the request is not included in but also not precluded by the XACML standard [OAS05a]. Several approaches to address this issue also applicable in [TAS]. In this setting, the policies contained in the request to the Trust PDP are not XACML policies, instead they encode a TAS\(^3\) trust policy in the combined policy language described in the section below.

### 2.4.2 The Trust Policy Language

The trust policy language is aimed to expressively capture users trust preferences, with metrics described in Section 2.2.1 and 2.2.2, to combine the decision result and to illustrate required interactions and restrictions between trust services. In order to see whether the increased expressiveness is applicable, user experiments and use case
analysis will be performed for a particular setting.

The trust policy statements as described in [TAS] consist of a required trust level for a given trust service and express the relation between trust sources. The following three sections described each trust policy statement specified in [TAS].

2.4.2.1 Type1 Trust Policy Statement

TAS³ defines **Type1** trust policy statement as follows:

**Definition 2.1.** Given a list of available trust services T₁, ..., Tₙ with corresponding trust metric languages L(T₁), ..., L(Tₙ), a Type1 trust policy statement (or single policy) is a triple <Tᵢ, m, l> where Tᵢ is (the identifier denoting) the service, m is trust metric for this service, i.e. m is in L(Tᵢ) and l is a number expressing a required trust level. A Type1 (full) trust policy or policy set is a set of Type1 trust policy statements along with a policy combining algorithm.

A Type1 trust policy basically the union of the trust service metric languages where the trust levels of different trust services can be combined with policy combination algorithms, for example the XACML standard deny-overrides implements logical conjunction, permit-overrides algorithms implements logical disjunction. The following example illustrates the definition:

**Example 2.5.** In the setting introduced in Section 2.3 the healthcare provider wants to evaluate the trustworthiness of a medical service provider in order to provide a medical assistance to the patients who register to a home telemonitoring program for patients who have certain diseases. We consider the following scenario: Bob, as a patient, has specified that the medical service providers must be a certified (accredited) nurse expressed by CTM metric m₁ and that their reputation according to RTM metric m₂ must be at least 0.8. The healthcare provider is only willing to work with an institution which perform sufficient (.75) on given business goals expressed by KPITM metric m₃. This results in the following policy:

\[(< \text{CTM, } m₁, \text{True, } < \text{RTM, m₂, .8, } < \text{KPITM, m₃, .75} >)\]

The policy combination algorithm is *Deny-overrides*, i.e. a failure to reach the required trust level on any of the metrics will cause the policy to evaluate to deny.

2.4.2.2 Type2 Trust Policy Statement

TAS³ defines **Type2** trust policy statement as follows:

**Definition 2.2.** A Type2 trust policy statement S₂ is a formula given by the following grammar: \( S₂ ::= S₁ | (S₂ \land S₂) | (S₂ \lor S₂) \) Where S₁ is a Type1 trust policy statement...
2.1. A Type2 (full) trust policy is a set of trust Type2 policy statements along with a policy combining algorithm.

A Type2 trust policy provides more intricate logical combination than Type1 policy language, therefore offering added flexibility. For example we can base trust on a sufficient trust level on metric 1 or a sufficient level on both metric 2 and metric 3.

Example 2.6. Consider the setting from Example 2.5, instead of requiring both certification and a good reputation, the user requires certification or a good reputation. With the same requirement as the previous example, the policy becomes:

\[(< \text{CTM}, m_1, \text{True} > \lor < \text{RTM}, m_2, .8 >) \land < \text{KPTM}, m_3, .75 >)\]

with no the policy combination algorithm to define. In the other hand, if the statement gets treated separately rather than integrated in the users policy statement:

\[(< \text{CTM}, m_1, \text{True} > \lor < \text{RTM}, m_2, .8 >), < \text{KPTM}, m_3, .75 >\]

with the Deny-overrides policy combination algorithm.

2.4.2.3 Type3 Trust Policy Statement

TAS³ defines Type3 trust policy statement as follows:

Definition 2.3. A Type 3 trust statement \( S_3 \) and nested metric \( M_T \) with top level \( T \) are given by:

\[
S_3 ::= < T, M_T, r > \mid (S_3 \land S_3) \mid (S_3 \lor S_3)
\]

\[
M_T ::= m \mid m(S_3, ..., S_3)
\]

Where \( T \) is a trust service, \( m \) a metric in \( L(T) \), \( r \) a minimum required trust level and \( m(S_3, ..., S_3) \) denotes a parameterized policy with the parameters instantiated with trust statements. If trust service \( T \) does not support parameterized policies then \( M_T \) will obviously be restricted to the ground term \( m \). A Type3 (full) trust policy is a set of trust Type3 policy statements along with a policy combining algorithm.

Example 2.7. Consider the setting from Example 2.6, if the user wants to make sure that the source of feedback used in computing a reputation is trusted (s)he could require that the reputation among (certified) members in his health program is sufficiently high (at least .8). With the same requirement as the previous one, the policy becomes:

\[< \text{RTM}, m_2(< \text{CTM}, m_1.\text{member}, \text{True} >), .8 > \land < \text{KPTM}, m_3, .75 >\]
As described in Section 2.4.1, TAS$^3$ trust policies [TAS] follow the approach of XACML policy or policy sets. Note that TAS$^3$ uses XACML semantics to implement the policy language.

Before explaining the XACML semantics in order to understand the settings where this policy language applies, the following section discusses various approaches to trust management integration.

### 2.5 Approaches to Trust Management Integration

During everyday activities, humans consider the mixture between two types of trust: structured trust and unstructured trust. Structured trust captures the trust relationships that exist between individuals and institutions, while unstructured trust represents the trust that can be inferred from the observations and opinions of others. Even though at fundamental level, both structured trust and unstructured trust involve the transfer of trust from one entity to another, they assume different sources of trust and accordingly propose different mechanisms to deal with trust.

Most research in trust management focuses only on one of these approaches (either structured or unstructured approach). However, each approach has its own advantages and limitations. For example, structured trust is benefit by the capability to express pre-condition of an entity, so that Bob, for example may specify that a service provider has to be accredited provider before he could use the service. Furthermore, structured trust results in only two condition, trusted or not. On the other hand, unstructured trust captures more flexible notion of trust, as they can reflect trust that inferred from an entity’s past behavior.

Several studies in trust management explore the combination of two type of trusts [JRSV07, LY09, CMM$^+$07, WAC$^+$09, HJS04, vDKP08, TAS] in their own settings and contexts, provide different solutions and mostly focus the discussion only from one type of trust (e.g. structured or unstructured). Most of these solutions try to provide a comprehensive solution towards trust management integration by providing the same policy language for both structured trust and unstructured trust as described in [JRSV07, LY09, CMM$^+$07, HJS04, WAC$^+$09, vDKP08].

For example, in hybrid trust model described in [JRSV07], the trust decision is a result of evaluating policies with all historical information which comprise three different components: (1) the experience information of the past interactions, (2) the recommendation from other entity and (3) the credentials of each entity participating in the evaluation process. This approach uses a policy language specified in [KNS05], a variant of LTL logic which incorporates simple disjunction and conjugation, and constraint to combine different sources of trust.

Another approach is CTM [LY09] which argued as the first solution towards developing a comprehensive approach to composite trust management for distributed
CTM provides a flexible policy language that allows arbitrary composition of structured and unstructured trust metrics. It supports sequential policy composition (nested policy) for both structured and unstructured trust metrics based on the filtering mechanism provided by the extended version of RT family language [LY09].

Beside providing the language for reputation-based and policy-based TM, [BDOS05] proposes an approach in which both of them can be integrated, based on the policy language PROTUNE [BO05]. [BDOS05] uses the notion of trust services by suggesting a modular approach, via delegation to suitable external packages. In Grid environment, [CMM+07] proposes fine grained access control for Globus which combines trust and reputation management based on RTML language [MMV05].

While the other TM integration solutions focus on the development of both structured and unstructured TM, other solutions use the decision manager approach towards trust management integration by providing one additional component which acts as a decision manager. For example, QuanTM [WAC+09] uses the decision manager component to make an optimal decision with respect to the current context, structured and unstructured values, and a predefined meta-policy. The decision manager’s meta-policy is defined during setup and is application dependent.

Beside the above solutions, there are also several reputation systems that incorporate credentials as part of their trust level evaluation [vDKP08, HJS04]. These approaches, however, could not be categorized as a composite approach since the intention of these works are limited only to enhance one aspect of TM (e.g. structured or unstructured).

After discussing the context where trust management integration solutions exist, the following section described the XACML language that is used to implement the trust policy language as a part of trust policy architecture mentioned in Section 2.4.1.

## 2.6 XACML

The eXtensible Access Control Markup Language or XACML is a popular OASIS standard XML-based language for access control policies. As an expressive language, XACML describes both an access control policy language and a request/response language for distributed resources. The core of XACML are attributes values of known type and name that is referred to a subject, a resource, an action and an environment in which the access request is made. When a request is sent to a PDP, that request is formed in terms of attributes, and these attributes will be compared to attribute values in a policy to render an access decision.

Beside defining the policy and request/response language, XACML standard also specifies a processing environment model to manage the policies, attributes and to render the access decisions. In this model, the XACML context handler encodes decision request into XACML canonical form and subsequently encodes the decision.
response into its native response format so that the application environment can communicate with the decision point. This process illustrates in Figure 2.2.

Beside rendering decision response in the form of negative or positive authorization, XACML also standardizes several other profiles, namely hierarchical Role Based Access Control (RBAC) [OAS05d] and privacy profile [OAS05b]. In addition, XACML has also been used for various purposes when defining security policies for distributed resources. The security policy itself may consist of access control policies [OAS05a], privacy policies [And06, OAS05b], and trust policies as proposed by [TAS] and mentioned in detail in Section 2.4.2. For a complete list of profiles and recent XACML OASIS standards, we suggest to visit the XACML OASIS website [OAS05a].

For clarity, we will briefly discuss the main components of XACML policy and request language in the following paragraph.

The root of all XACML policies is a policy set (PS) which contains other policy sets and policies (POLICYSET). Policy (POL) contains a set of rules (RULESET).
Each policy set and policy contains policy and rule combining algorithms for resolving result conflict. For policy set, the policy combining algorithms (PCA) are: first-applicable, deny-overrides, permit-overrides, and only-one-applicable. While for policy, the rule combining algorithms (RCA) does not include only-one-applicable algorithm. Beside combining algorithms, policy set, policy and rules comprise a target (TARGET) which is necessary to be specified in order to find the corresponding policy for a given request. Each target component, namely subjects (SUBJECTS), resources (RESOURCES) and actions (ACTIONS) is described in term of matching function (MATCH) which is used to match attribute value (AV) with attributes referenced by attribute designator (AD) or attribute selector (AS) finder. In addition, the rules also consist of permit or deny decision (EFFECT) and condition (COND). The condition is a placeholder to express the policy logic using functions (FUNC) describe in standard XACML specification [OAS05a]. For example, the function string-one-and-only is used to ensure that a particular attribute in the request context contains exactly one value.

For XACML request, each request contains zero or more typed attribute with attribute value(s) which correspond to subjects, resources and actions in the previous paragraph.

In the end the XACML processing environment will return the following decision result based on the possible policy combiners algorithms: (1) permit, if the subject is permitted to perform an action to the corresponding resource, (2) deny, if the subject is denied to perform an action to the corresponding resource, (3) not applicable, if the request cannot be evaluated, (4) indeterminate, if some error occurs during evaluation such that the policy engine is unable to determine a valid result for evaluating the policy or rule.

Formalizing the overall XACML specification is not the intention of this thesis, therefore for illustrative purpose, we use a subset of XACML to specify the previous illustration as shown in Listing 2.1 and 2.2, which we called Trust XACML (TXACML) with a lisp-like syntax, adapted from [TK06, Kol08].

The syntax for policy is:

```
S ::= POLICYSET
POLICYSET ::= PS | POL
   PS ::= ( PolicySet PCA TARGET POLICYSET* )
   PCA ::= First-Applicable | Deny-Overrides | Permit-Overrides |
          Only-One-Applicable
   TARGET ::= ( (SUBJECTS* ) (RESOURCES* ) (ACTIONS* ) )
SUBJECTS ::= Any | MATCH*
RESOURCES ::= Any | MATCH*
ACTIONS ::= Any | MATCH*
POL ::= ( Policy RCA TARGET RULESET )
   RCA ::= First-Applicable | Deny-Overrides | Permit-Overrides
RULESET ::= R*
```
R ::= (Rule TARGET EFFECT COND)
EFFECT ::= Permit | Deny
MATCH ::= (Match AV AD MATCH–FCN) | (Match AV AS MATCH–FCN)
AD ::= (AttributeDesignator ATTR–ID TYPE)
AS ::= (AttributeSelector CONTEXT–PATH TYPE)
AV ::= (AttributeValue VAL TYPE)
MATCH–FCN ::= type–equal | type–greater–than |
                 type–greater–than–or–equal | type–less–than |
                 type–less–than–or–equal | type–regexp–match
TYPE ::= string | boolean | integer | ...
COND ::= (Condition EXPR∗)
EXPR ::= APPLY | AS | AV | FUNCTION | AD
APPLY ::= (Apply FUNCTION)
FUNCTION ::= (FCN–ID EXPR∗)
FCN–ID ::= any–of | all–of | regexp–match | ...

Listing 2.1: TXACML Policy Specification Syntax

Example 2.8. The following is a TXACML policy:

(Policy Deny–Overrides
  ((Any)
    ((Match (AttributeValue data1 string)(AttributeDesignator
      resource–id string) type–equal))
    (Any))
  (Rule
    (((Match (AttributeValue nurse string)(AttributeDesignator role
      string) type–equal))
    (Any)
    ((Match (AttributeValue read string)(AttributeDesignator
      action–id string) type–equal)))
    Permit)
  (Rule
    (((()))
    ((())) Deny))

In this example, this policy denies all requests for access to data1 except those made
by nurses with action read. In detail, the policy is composed of two rules using the
combinator deny-overrides and applies only to requests where the resource has the
name of data1. The first rule permits requests where the subject has the role of nurse
with an action read regardless of the resource. The second denies all requests.

The syntax for requests is:

RQ ::= (Request (AVC∗)) (AVC∗) (AVC∗))
AVC ::= (Attribute ATTR–ID TYPE AV∗)

Listing 2.2: TXACML Request Specification Syntax
Example 2.9. The following is a TXACML request:

\[
\text{(Request} \quad ((\text{Attribute subject-id string (amanda string)}) (\text{Attribute role string (AttributeValue nurse string)})) \\
((\text{Attribute resource-id string (AttributeValue data1 string)})) \\
((\text{Attribute action-id string (AttributeValue read string)})))
\]

It simply list the subject, resource, and action attributes. In detail, in this example, Amanda who has a nurse role would like to read data1 which belong to another entity.

In order to avoid a verbose XML representation of XACML, examples presented in this thesis will be described using the previous syntax as shown in Listing 2.1 and 2.2.

Before presenting the real-life settings where XACML is applicable and going to the detail of XACML implementation, the following section will discuss how XACML handles the results conflict as mentioned above and fits into TAS³ trust policy language as described in Section 2.4.2.

2.6.1 Combining Algorithms

The need for combining algorithms arises to address conflict resolution issues when different sub policies or rules have different decision results. In this context, a combining algorithm can be applied either in policy level or rule level. For example, a policy combining algorithm can briefly described as follows: in permit-overrides algorithm, permit takes precedence while in deny-overrides algorithm, deny takes precedence. In first-applicable algorithm, the decision reached by the first sub policy which has a determinate value of permit or deny and in the only-one-applicable algorithm, the decision is the result of evaluation the sub policy, if only one sub policy is considered applicable by evaluation of its target.

In detail, if one policy set applies the deny-overrides policy combination algorithm described above and mentioned in Type1 trust policy language in Section 2.4.2.1, the following mechanism based on algorithm 2.1 will apply.

If any policy evaluates to Deny, then the result will be Deny. If all policies are found to be Not Applicable to the decision request, then the policy set will return Not Applicable. If an error occurs while evaluating the target of a policy, or a reference to a policy is considered invalid or the policy evaluation results in Indeterminate, then the policy set will return Deny.

After discussing the XACML components, the following section presents the works of XACML in healthcare domain.
2.6.2 XACML in Healthcare Domain

Related works on access control policies in healthcare domain have been studied for various purposes. ARTEMIS and MEDIS [SS09] are two examples of using XACML in healthcare information system which are aimed to develop a semantic web service based on P2P interoperability infrastructure. The other, SAPHIRE [ND08], is built to implement an intelligent health care monitoring and decision support system using SAML and XACML in web service infrastructure. Another example is the Australian’s project NEHTA (National E-Health Transition Authority) which explicitly recommends XACML as a standard method of defining access policies for e-health services.

Since the works in Section 2.5 are aimed to create real-life solutions of the trust management integration approach, we need to consider implementing XACML language for this purpose. To date, several XACML implementations exist which address different approaches and solutions [TC08, Sun], however, for the purpose of implementing a proof-of-concept implementation, the Sun’s XACML prototype implementation is enough to serve this thesis objective. The following section will explains the nitty-gritty of Sun’s XACML implementation.

2.6.3 Sun’s XACML Implementation

To date, the stable version of Sun’s XACML implementation (version 1.2) has supported only XACML 1.0 and not XACML 2.0 yet. However, the current state of this implementation also provides several XACML 2.0 features, such as:

*com.sun.xacml.cond.VariableReference* - This object allows an expression to reference a variable definition. A reference can be included anywhere in an expression where the referenced expression would be valid.

*com.sun.xacml.cond.VariableManager* - This object is used by the parsing routines to handle the relationships between variable references and definitions.

*com.sun.xacml.cond.VariableDefinition* - This object allows a policy to pre-define any number of expression blocks for general use.

and also other features as shown in [Sun]. In order to work with the latest implementation features, we download the code from Code Version System (CVS) located in sourceforge.net and then compile them using the given build.xml file. In this way, we can obtain the last stable version of Sun’s XACML implementation.

---

4see http://203.110.153.105/index.php?option=com_standardscatalogue&m=resource&cid=17&id=113&Itemid=424
The Sun’s XACML implementation APIs [Sun] consists of several important packages as follows:

**com.sun.xacml** is the core package. It contains the core PDP class which provides starting point for most code, the logic for various type of target matching, policy and rule evaluation, policy and policy set handling, obligations (if exist), basic configuration store and other related features.

**com.sun.xacml.attr** is the package that supports all the standard XACML attribute data types (see [OAS05a]) and attribute type designators and selectors for referencing to the standard XACML attributes. This package also has the factories which is used to create new attribute values and, standard interfaces and abstract classes to define new attributes types.

**com.sun.xacml.combine** contains all of the standard XACML policy and rule combining algorithms as well as the factory for accessing those algorithms. Standard interfaces are provided for creating new combining algorithms.

**com.sun.xacml.cond** provides all of the standard XACML conditions and functions logic (except the XPath functions which define in another module). Standard interfaces and classes are provided for defining new functions.

**com.sun.xacml.ctx** represents all of the types defined in the XACML context schema (the request and response context). All of these classes are both encodable and parsable, making it easy to build a PEP based on these classes.

**com.sun.xacml.finder** provides support for retrieving and referencing things that the PDP needs. This package is typically used for finding policies, retrieving attributes not provided in the request, resolving resource identifiers, or generating real-time values. A set of standard classes are used for defining different kinds of finder modules. Additional finder package **com.sun.xacml.finder.impl** includes to provide selector functionality using XPath mechanism, simple environment module and file-based access to a specific set of policies.

Also in the APIs are **com.sun.xacml.attr.proxy** and **com.sun.xacml.cond.cluster** which are the helper classes used by the configuration code. For more complete list of the APIs, the reader could visit Sun’s XACML JavaDocs\(^5\). Recent implementation of Sun’s XACML software also adds additional package to provide support for several XACML 2.0 features via **com.sun.xacml.support.finder** modules.

\(^5\)http://sunxacml.sourceforge.net/javadoc/index.html
Algorithm 2.1 Deny-overrides policy-combining algorithm [OAS05a]

**Input:** Set of policies

\[
\text{atLeastOnePermit} \leftarrow \text{false}; \\
\text{for } i = 0 \text{ to lengthOf(Policy policy[i]) do} \\
\quad \text{decision} \leftarrow \text{evaluate(policy[i])}; \\
\quad \text{if } \text{decision} = \text{Deny} \text{ then} \\
\quad \quad \text{Deny}; \\
\quad \text{end if} \\
\quad \text{if } \text{decision} = \text{Permit} \text{ then} \\
\quad \quad \text{atLeastOnePermit} \leftarrow \text{true}; \\
\quad \quad \text{continue}; \\
\quad \text{end if} \\
\quad \text{if } \text{decision} = \text{NotApplicable} \text{ then} \\
\quad \quad \text{continue}; \\
\quad \text{end if} \\
\quad \text{if } \text{decision} = \text{Indeterminate} \text{ then} \\
\quad \quad \text{return Deny}; \\
\quad \text{end if} \\
\quad \text{end if} \\
\text{i} \leftarrow i + 1 \\
\text{end for} \\
\text{if } \text{atLeastOnePermit} \text{ then} \\
\quad \text{return Permit}; \\
\text{end if} \\
\text{return NotApplicable; }
This chapter describes various healthcare services, applicable scenarios which show trust relations between each stake holder in the scenario, explains and concentrates on the particular scenario which will be used throughout this thesis.

3.1 Overview

Over the years, healthcare services have gradually evolved from traditional care (e.g. inpatient, outpatient, home) to telehealthcare which exploits the advances in communication and information technology in order to deliver the healthcare services outside the boundary of physical healthcare settings.

Traditional inpatient care involves patients visiting a hospital, clinic, or physician’s office for diagnosis or treatment and spending the night in these institutions. While outpatient care is carried out without spending the night in these places. Traditional home care typically involves periodic home visits by a physician, a nurse or other healthcare practitioners to treat patients with chronic health problems.

In telehealthcare settings, however, the necessity for direct physical contact is reduced and later it is offered during a follow-up responses when the patients need an urgent care. The standard telehealthcare service involves exchanging medical information between patients and their healthcare providers in the forms of digital images, audiovisual communication and data transfer. In addition, a telehealthcare service often utilizes unobtrusive equipments worn in the patients body to provide a detailed measurement of patients physical condition.

In [SEP06], telemedicine, home telecare and remote patient monitoring are defined as subset of telehealthcare services. These healthcare services are used in various medical domain using different approaches and methodologies. For example, telemedicine is used in remote consultative and diagnostic medicine domain, while home telecare is applied in remote homecare domain to provide care for the elderly and vulnerable
people to live in their own environments. Furthermore, remote patient monitoring or home telemonitoring extends the remote care service by involving a wide range of medical devices, softwares, and other optionally environment sensors to passively collect physiological and contextual data of patients and transmit the collected data to the remote care provider, either in real-time or intermittently, for review and intervention [SEP06].

Remote patient monitoring has been mostly deployed in the disease management domain [SEP06] which involves a systematic care management process of patients with specific diseases (e.g. chronic conditions). In disease management domain programs, the remote patient monitoring solutions are mainly used in the management of cardiac patients, diabetes patients, and pulmonary patients. To date, remote patient monitoring applications are being researched to be implemented in the domain of post-event rehabilitation (e.g. after cardiac surgery), diagnostic monitoring (e.g. of cardiac events), obesity and weight management, and elderly care [SEP06].

Many approaches to remote patient monitoring exist today. For example, as shown in Figure 3.1, a home telemonitoring solution from Phillips telemonitoring service is depicted and clearly illustrated to provide their own proprietary solutions to remote patient monitoring service. A typical scenario in this service includes patients who registered to the healthcare service, medical observation instruments to monitor patients vital signs, softwares and hardwares to transmit back and forth the medical information of the patients and finally the care team who reviews and follows up the information received by the care system. By implementing a monitoring program with follow-up scenarios (e.g.emergency response), the healthcare service could provide an efficient early warning mechanisms for the healthcare community as a whole to support patient behavior change and avoid costly hospitalization [SEP06, CPR+04].

we present our own remote patient monitoring scenario based on the previous illustration in the following section.

### 3.2 Home Telemonitoring Program for Patients with Chronic Heart Failure

In this section, we develop our own scenario for patients with chronic health failure that have subscribed to the home telemonitoring program of a healthcare provider. In addition to a typical home telemonitoring solution, we also provide trust relations between stake holders in this scenario.

As illustrated in Figure 3.2, the stake holders in the scenario are:

**MOP or Medical Observation Provider.** MOP is utilized to measure vital signs such as body weight, blood pressure, glucose or ECG rhythm. These devices
may be stationary, portable, or body-worn. As technology grows, these devices might be implemented in the form of low-cost unobtrusive monitoring devices, with pervasive connectivity, intelligent, using personalized applications and services. Many of these technology building blocks are crossing over from the consumer and entertainment sectors into healthcare and wellness [SEP06].

**MSP or Medical Service Provider.** MSP is an entity in the form of an institution or person who provides professional medical services to the patients in need of healthcare services. In our scenario, the list of applicable MSPs for a given healthcare unit is assumed to be available for each HCP after performing credential discovery process. A typical MSP officers comprise a list of physicians, nurses, medical interns, social workers or pharmacist who qualified and accredited to perform medical services.

**HCP or Health Care Provider.** HCP is an accredited healthcare institution which provides a systematic and professional medical care (e.g., in our scenario, home telemonitoring program for patients with chronic heart failure) and consists of the following subsection:
Monitoring Center acts as the back-end data center for all subscribed patients. The collection of data is being made available for review by physician, nurse, care manager or other care team members.

MERC or Medical Emergency Response Center provides a follow-up response based on alarm received by the care system or manually generated as a result of the care team reviews.

Reputation Service. The reputation service receives feedbacks from the patients who give different opinions related to different aspects of the quality of service delivered. The patients have an ability to provide ratings for different contexts.

MSP List is maintained in the HCP for the purpose of performing follow-up responses after conducting trust and privacy negotiation process to ensure only trustworthy MSPs are on the list.

For clarity, the following example illustrates how the scenario works for a particular
case.

After Bob diagnosed having a chronic heart problem, he subscribes in the home telemonitoring program in one of the HCP in the city hospital which provides medical services related to Bob’s health problem; as suggested and referenced by his personal physician. Bob registers to this program in order to prevent worsening of the disease and deal with urgent conditions. Right after the installation of required software and hardware, Bob starts using several trusted stationary and body-worn measurement devices to capture his vital signs (step 1) and automatically collects these information in his home system within a predetermined period (step 2). His home system will then send the collected data along with each device identity to the HCP system. The HCP system sends feedback (step 3) either based on an automated indication or as a result of care team reviews. After a period of time, Bob’s home system captures an indication of health problem requiring urgent treatment based on predetermined indicator and sends an alarm (step 4) to the monitoring unit in HCP. Subsequently, the HCP system will send alerts (step 5) to a trusted MSP officers in the nearest location where Bob lives who oblige to give a response (step 6) as soon as possible. A selected MSP officers will finally receive Bob’s medical information and Bob’s location (step 7) in order to perform the medical services in time. Eventually, Bob could send feedback (step 8) related to different aspects of the services (e.g. punctuality) given by the MSP.

Based on the above example, we could see various interactions performed by different trusted providers which play important part in the scenario. In order to provide a detailed description of the trust management architecture scenario shown in Figure 3.2, I will illustrate the scenario, adapted from [TAS] in the following section.

### 3.3 Emergency Scenario

Although various settings would be applicable in this context, such as urgent medicine assistance, data collection from measurement devices and prior credential initialization for MSPs as presented previously in the example, we choose to concentrate on the emergency scenario dealt with a patient who has an emergency case (e.g. heart attack) and MSP officers who have to perform an emergency care (e.g. artificial respiration, external chest compression) as quickly as possible on scene.

As shown in Figure 3.3 the emergency scenario involves the following steps:

1. A patient who suffers from CHF contacts his personal physician to give a reference to be admitted to a home telemonitoring program for patients with CHF in a local hospital. Using this reference, a patient registers for the program and directly contacts the care manager.
2. The patient is then referred to a Cardiologist who is responsible to assess the current health status of the patient, resulting in a Personal Health Record Profile (PHRP) of the patient.

3. Upon finishing the assessment the patient, assisted by the care manager, determines his preferences on MOPs and MSPs based on his personal preferences and medical situation.

4. Based on this information, the care manager selects suitable MOPs to be used in collecting and monitoring health indicators.

5. The selected MOPs then initialize a local PHRP of the patient and starts to monitor the patient health indicators.

6. With a specific time interval, the local PHRP is forwarded to the monitoring center in HCP and analyzed by an MERC officer in charge. During this period, if an urgent situation occurs (i.e. heart attack, unusual high blood pressure), the alarm will be generated either automatically from the patient home or manually initiated by an MERC officer in charge as a result of a medical analysis of the patient.

7. After receiving the urgent notification, the MERC officer selects suitable MSPs based on the location of the patient to perform an early response on scene.

8. Based on this selection, the alert will be sent directly to the selected MSP officers who upon receiving the message are obligated to send a response back to the monitoring center. The monitoring center then sends the actual data related to the patient (i.e. personal information, specific information on the PHRP of the patient and patient location) to the MSP officer who gives a positive response.

9. Eventually, after receiving the medical assistance from an MSP officer, the patient gives a feedback to the reputation service whom he already registered.

In the emergency scenario shown in Figure 3.3, it can be seen that there are two steps which show trust and privacy negotiation. However, in this thesis, we will focus only on the activities involving MSP officers which is illustrated as follows.

Since a preference regarding the MSPs has been set by the patient (step 3), the MERC officer will start and check whether an alert message can be sent to P4 (step 7). As a part of this check the Master PDP will be ask for an authorization which consist of the request purpose and the requester role. Since the policy also including a trust policy, the Master PDP forwards the evaluation request to the Trust PDP and asks if P4 is trusted to received Bob’s blood pressure information and his current medical status. The Trust PDP then evaluates the request based on Bob’s trust preferences.
Figure 3.3: Emergency scenario for home telemonitoring program for patients with Chronic Heart Failure (CHF)

As a part of the request evaluation, the role of Trust PDP in this scenario can be described in the following situation.

The reputation service runs by a public city hospital where Bob lives allows anyone from different home telemonitoring programs (e.g. heart failure and stroke) provides feedback on their experience with an MSP officer. In addition, a local chapter of Medical Professional Association in the public city hospital also runs a reputation service which allows medical professionals who have been trained to get a first-aid certificate or received a medical degree provide feedback on their experience with the Training (i.e. Red Cross) or Education Provider (i.e. Medical University). The Medical Professional Association is endorsing Bob to access the reputation service, since the health telemonitoring program where bob is admitted is able to determine a preference of MSP based on which organization or university does the MSP has been trained or educated from.

As suggested by his care manager, Bob has decided to trust only the MSPs who has a first-aid certificate from a reputable organization or medical degree from a
Figure 3.4: Trust evaluation flow for Bob’s trust policy

good university which has a good reputation from all patients who are registered to
different home telemonitoring programs in the public city hospital. This situation is
shown in Figure 3.4 and captured in a policy which combines two different type of
trust services.

The MSP officer credential is captured on a nested trust metric policy while the
reputation of the MSP officer is captured in a single reputation trust metric policy. To
evaluate Bob’s policy the Trust PDP will perform the following steps:

1. After receiving a request from the Master PDP,

2. the Trust PDP queries the Structural Trust Service to determine which are
   the trusted users (i.e. the MSP officer). To get the credential of the trusted
   users, the Structural Trust Service needs to calculate the reputation of the
   training provider or educational institution where the trusted users are trained
   or educated. To find this information,

3. the Structural Trust Service will call the Trust PDP through the call-back mech-
   anism,

4. then the Trust PDP will forward the request to the Behavioral Trust Service
   to find the reputation of the training provider or educational institution where
   the trusted users are trained or educated,

5. and return this to the Trust PDP. Subsequently,
6. the Trust PDP sends the result to the Structural Trust Service as a response to a call-back mechanism. The Structural Trust Service now has a complete information to get the credential of the trusted users,

7. and returns a *Permit* decision to the Trust PDP. However to sufficiently returns a trust decision response to the Master PDP,

8. the Trust PDP needs to call the Behavioral Trust Service to find the reputation of P4. The Behavioral Trust Service calculates the reputation value of P4, which turns out to be sufficiently high,

9. and returns a *Permit* decision to the the Trust PDP. Finally,

10. after combining the decisions, the Trust PDP is able to return a *Permit* response to the Master PDP.
Chapter 4

Requirements

This chapter defines requirements for the Trust PDP and trust services along with their related components based on applicable scenarios presented in the previous chapter. The requirements presented in this chapter are used as a baseline for assessing the Trust PDP implementation in later chapter.

4.1 Overview

The scenario given in the previous chapter has led us to identify several requirements to implement the trust evaluation model depicted in the scenario. Explicitly, the scenario also requires the necessity of protecting the resources using policies and provides finding a match between requester and service provider through trust and privacy negotiation.

As illustrated in the aforementioned trust evaluation model and specified in [TAS], all requests for access to a protected resource go through an abstract component called a TAS$^3$ Master Policy Decision Point or Master PDP as described in Section 2.4.1. Based on Master PDP meta policy, the formulated request that includes a description of the access request (e.g., the requester, the requested resource, the action performed to the requested resource, the statement of purpose and other relevant information) will be forwarded to subsequent abstract component called a Trust Policy Decision Point or Trust PDP. The forwarding mechanism implies that the information specify in the Master PDP includes references to trust policies.

Subsequently, the Trust PDP retrieves related policies from the policy store that are applicable to the request, along with any additional information required to evaluate those policies in the form of attributes accompanying the request. Then these attributes will be supplied to the trust services as mentioned in Section 2.2.1 and 2.2.2. Based on the request and its applicable policies, the Trust PDP determines the
following actions: (1) which trust services it needs to contact and (2) how to combine their responses with methods presented in detail in Section 2.4.1.

The illustrated trust evaluation model also indicates the usage of sequential composition policies where the trust level computed by one trust service depends on the trust level provided by another trust service. As mentioned in 2.4.1, in order to address this problem, the Trust PDP will implement a call-back mechanism to facilitate interaction between the trust services and Trust PDP. Based on this mechanism, the response received as a result of forwarding the request to another trust service by the Trust PDP will be forwarded back to the first trust service, so that the initial trust service could compute the intended trust level. In the case where the requestor is looking for trusted entities which are not available in local environment, [TAS] specifies that this discovery of trusted entities should be delegated to the trust services based on the available trust information. In addition, when the Trust PDP deals with single type or logical combination policy language as described in Section 2.4.2, the Trust PDP would not need to implement the call-back mechanism.

As described in the previous explanation, the trust evaluation model entails that we need to specify two different interfaces for TAS³ trust policy language; the Trust PDP interface and the trust service interface. In the next section, we will describe the requirements for the Trust PDP interface using the terminology adopted from [TAS] and afterwards describe the requirements for a TAS³ trust service in a later section. In addition, the requirements presented in this chapter are requirements to be fulfilled by a Trust PDP implementation.

Note that in order to understand the meaning and intention of which the requirements for the Trust PDP interface and trust service interface are specified, the key words must, must not, required, shall, shall not, should, should not, recommended, may, and optional in this document are to be interpreted as described in IETF RFC 2119 [Bra97].

4.2 The Trust PDP Interface

The following requirements briefly describe the behavior of the Trust PDP:

**Unified Representation.** As mentioned in [TAS], the Trust PDP MUST provide a standard interface to the rest of the TAS³ framework and MUST also accept trust evaluation requests formatted as XACML request contexts and respond with a XACML decision context. An interface satisfying this requirement would allow trust metric to be included into the policy without introducing new semantics to XACML specification.

**Flexible Policy Composition.** The Trust PDP MUST support various TAS³ trust policy language as mentioned in Section 2.4.1. An arbitrary flexible composition
of policies imply the use of independent, simple disjunction and conjunction, and sequential composition (levelled) inside the policies. These policies will then evaluate on each trust service. As specified in [TAS], supporting the policy language implies:

1. the capability of selecting applicable policies from the policy store,
2. extracting the trust service information, the trust metric and required trust level specified by any applicable policy,
3. calling the trust service to evaluate the trust metric and acquire a trust level,
4. comparing the trust level acquired from the trust service to the required trust level extracted from the applicable policy,
5. combining the result of the applicable policies according to the policy combination rules.

Flexible Algorithm. The Trust PDP SHOULD support an arbitrary set of algorithm for dealing with the TAS trust policy language. [TAS] specifies that the Trust PDP SHOULD support a call-back mechanism between the trust services and the Trust PDP. This mechanism is required to support sequential policy composition. The call-back function in this particular settings will accept a trust metric, a minimum required level and optionally a set of entities for which the trust metric needs to be evaluated. The call-back function should return a sorted list of (entity, trust level) pairs containing all entities (in the set) which reach the predetermined trust level [TAS]. A subset of this mechanism which provides a similar support for sequential policy composition could also be applied in this context. This approach uses filtering mechanism to limit the input set given to another trust service.

Trusted Entities Discovery. As decried earlier, the Trust PDP SHOULD support discovery of trusted entities. However, to be able to perform trusted entities discovery, the (top level) trust service specified in the trust policy to be evaluated also needs to support trusted entities discovery.

4.3 The Trust Service Interface

A trust service should satisfy the following requirements for implementing the trust policy architecture specified in [TAS]. Note that TAS also defines several trust services which are describe in Section 2.4. This specification implies the existence of additional or novel trust services which could be easily added to the Trust PDP.
Unified Representation of Trust Metrics. [TAS] specifies that a trust service MUST supply a TAS³ compatible trust metric language. This requirement determines the representation of trust metrics (e.g. how the service should compute its level of trust). However, this unifying representation does not prohibit the expressiveness of the trust metric language. In addition, this requirement also requires the trust services to encode each trust metric definition in an XACML policy attributes which is used to pass trust policies and metrics.

Parameterized Trust Metrics. If applicable, a trust service SHOULD supply a parameterized metric language as mentioned in Section 2.2.1 and 2.2.2 which allows embedding (other) TAS³ trust metrics [TAS]. This is used to build a sequential policy composition. The embedded trust metrics will be given as a parameters and will evaluate to a sorted list of (entity, trust level) pairs. If the support for embedding the trust metric is not applicable, the trust service can still be used but only in independent policies context or as the lowest level in a levelled policy.

Single Entity Evaluation. A trust service MUST supply a single entity trust level evaluation interface as a basic functionality of the trust service [TAS]. The trust service will need to provide interfaces and routines for the Trust PDP to call to evaluate a trust metric and obtain a corresponding trust level. The returned trust level must be a standard numeric value or boolean. The trust service must also provide a routine to compare trust levels.

Group Entity Evaluation. A trust service SHOULD support a group of entities trust level evaluation interface [TAS]. This evaluation function is needed to support sequential composition (levelled) of policies. Without this feature, the trust service can still be used but only in independent policies or as the top level in a levelled policy.

Trusted Entities Discovery Evaluation. A trust service SHOULD support a trust level evaluation interface for discovering entities with a given minimum trust level [TAS]. This feature corresponds to the support of trusted entities discovery by the Trust PDP.
Chapter 5

Design

This chapter illustrates the high-level design of TAS\textsuperscript{3} Trust Policy Architecture and explains how the Trust PDP and trust services fits into this architecture. The design implements separation of layers into different parts of components and explains their relations.

5.1 Overview

A key goal of the TAS\textsuperscript{3} Trust Policy Architecture as mentioned in Section 2.4.1 is to enable the independent attachment or development of trust services by different trust providers. To address this objective, we considered a multi-layered architecture so that the trust services can effectively utilize the Trust PDP interface and vice versa. Layering and components are also important for a clean architecture, thus making this architecture more flexible and helping to increase its extensibility feature.

As previously mentioned, a multi-layered Trust Policy Architecture is briefly described as follows:

**The PEP Layer.** The PEP is a system entity that performs policy decisions by performing decision requests and enforcing the decisions response.

**The Master PDP Layer.** In trust policy architecture settings as mentioned in Section 2.4.1, the PEP uses a Master PDP to make its policy decisions. The Master PDP forwards the decision requests into the subsequent layer which consists of specialized PDPs.

**The Policy Decision Layer.** It consists several specialized PDPs, including a PDP specialized in answering trust policy queries (e.g. Trust PDP). This system entity evaluates the decision requests based on applicable policies and finally
renders the decision result. This Trust PDP includes the main interfaces to the trust service component described in Section 2.2.1.1 and 2.2.2.1.

In Trust PDP settings, the design also involves the following layers.

**The Trust Services Layer.** This layer consists of several trust service components where their interactions govern by the top level policy definition.

**The Utilities Layer.** If applicable, this layer presents external components to provide specific functions (e.g. logging mechanism, data store interfaces) that are used by the system entities.

### 5.2 Architectural Requirements

Based on the goal specified in the previous section, we identify the following requirements for the Trust Policy Architecture.

**Independent.** A system independency is achieved by implementing a loose attachment between system components. A multi-layered approach presented previously can be seen a way to implement this requirement. In addition, this requirement also implies the following conditions:

1. Each layer MAY use the layers below,
2. but none SHOULD use the layers above, and
3. component subsystems within a layer SHOULD NOT refer to each other in order to reduce dependencies.

**Flexible.** In order to have extensibility feature, the trust policy architecture MUST allow easy integration of additional trust services. Since [TAS] also envisions novel trust services to be added into the architecture, it is reasonable to include this requirement into the architectural design. In this settings, the existing trust services do not need to be modified or replaced when adding a new service since only the Trust PDP is aware of the different trust services available in the system.

### 5.3 Architectural Design

The trust policy architecture implements a Trust PDP which is called by the Master PDP and manages the list of dedicated trust services and facilitates their interaction.
as described above (see Section 2.4.1). The Trust PDP also functions to integrate
the decision of different trust services. In this settings, we also identify and describe
these services as mentioned in Section 2.2.1 and 2.2.2. The POLIPO CTM trust
service falls in the structural trust service component which specifies the requested
trust level based on trust rules and delegation process. The Centrality RTM trust
service falls in the behavioral trust service component which builds the requested trust
level computed as a reputation value from feedbacks and recommendations which is
collected over time. In the mean time, the KPITM trust service which also falls in the
same category as the Centrality RTM trust service provides the trust level based on
a more objective feedbacks and recommendation captured in dynamically obtained
factors.

Since the use of XACML is mandated as decision requests and responses as spec-
ified in Section 4 and acts as language (syntax) to facilitate interaction between
the Trust PDP and trust services, we implement the architecture based on XACML
specification [OAS05a]. However the necessity towards development of fine-grained
components has bring us to extend the architecture based on [OAS05e] and [OAS05c]
as presented in Figure 5.1.

Before explaining in more detail, we introduce terminologies used in this chapter
and found throughout this document.

**PEP**  - Policy Enforcement Point. See Section 5.1.

**AA**  - Attribute Authority. A system entity trusted by one or more entities to create
and sign attribute certificates. See [OAS05e, OAS05c].

**PDP**  - Policy Decision Point. A system entity that evaluates an access request
against one or more applicable policies to produce an access decision.

**Master PDP**  - Master Policy Decision Point. See Section 5.1.

**Sub PDP**  - A collection of various specialized Policy Decision Point, called by Mas-
ter PDP.

**Authorization PDP**  - Authorization Policy Decision Point. Similar to PDP defi-
nition as presented in [OAS05a].

**Trust PDP**  - Trust Policy Decision Point. See Section 5.1.

**Privacy PDP**  - Privacy Policy Decision Point. The system entity that specialized
in answering privacy policy queries.

**Trust Service**  . See Section 5.1.
Behavioral Trust Service. Trust service called by Trust PDP or other service providers, evaluating behavioral trust metrics.

Structural Trust Service. Trust service called by Trust PDP or other service providers, evaluating structural trust metrics.

KPITM Trust Service. Trust service called by Trust PDP or other service providers, evaluating a variant of behavioral trust metrics.

PAP - Policy Administration Point. The system entity that manages the policies. The availability of PAP system is crucial in the policy-based trust management since the decision responses will be based on the applicable policies fetched from the PAP. Different PDPs might implement their own PAP system.

As mentioned in Section 2.6, a standard XACML profile specifies four main components to handle access decisions: (1) PEP as an interface the outside world. It receives the access requests and evaluates them with the help of the other components, makes decision requests and enforces obligation statements. (2) PAP which acts as the repository for the policies and presents the policies to the PDP. (3) PDP is the decision point interface for the access requests. It collects the necessary information from other components to produce decision requests. (4) Policy Information Point (PIP) as the point where the necessary attributes for the policy evaluation are retrieved from several external or internal components.

In order to extend the above specification, we need to set the protocol and transport mechanisms for exchanging messages between XACML components described above. SAML by its nature is designed for carrying the security and authorization related information and have the bindings to basic transportation mechanisms. Therefore, the architecture should utilize OASIS SAML profile for the XACML [OAS05c] to carry the XACML messages between the XACML components. This profile defines the usage of SAML 2.0 [OAS05e] to protect, transport, and request XACML instances and other information. It specifies Attribute Request to the Attribute Authorities or Attribute Repositories, Attribute Statement and Assertion as a response from the Attribute Request, Policy Request to query policy from the PAP and Policy Statement or Assertion which carries the policies requested from the PAP.

Figure 5.1 illustrates the interaction depicted above, along with its information flow. As can be seen in the figure, the PEP is the component where the request is received. In this section, the attributes in the request is in the format of SAML attribute assertion. The PEP will map the request and attributes to different contexts (e.g. the XACML Request context in Trust PDP). Then the PEP sends this request to the Master PDP. Subsequently the Master PDP will forward the request to each specialized PDP based on the references specified in the meta-policies. During evaluation of the request, each specialized PDP will fetch the policies from the PAP and
if the request sent by the Master PDP is intended to be evaluated by the Trust PDP, the Trust PDP will then contact the associated trust services based on the applicable policies. Beside forwarding and encoding the request to the related trust services, The Trust PDP also provides the interaction between each trust services. After evaluation, the Trust PDP sends the XACML Response to the Master PDP, which in turn processes the response and sends the result via the Context Handler. The PEP fulfills the obligations if they exist and applies any policy decision that Master PDP concludes.

For the purpose of this thesis, the previous architecture holds the following assumptions:

**Policies are publicly disclosable.** The Trust PDP has all the information needed to make the decision responses since there is no reason in this settings why the Trust PDP could not disclose its relevant applicable policies based on the decision requests.

**Trust PDP has a list of trusted trust services.** In order to facilitate interaction between trust services, the Trust PDP assumes to manage the list of dedicated trust services.
Figure 5.1: Extended XACML architecture [OAS05a] in the TAS³ trust policy architecture settings
Chapter 6

Implementation

This chapter provides low-level specification and the implementation of the Trust PDP in Java routines which accepts XACML request context objects, evaluates trust policies encoded and wrapped in XACML policy objects and returns standard XACML responses. The implementation of the Trust PDP acts as a glue between the implementation of two different trust services: the POLIPO credential based service which utilizes SAML attribute query for querying the credentials and SAML assertions to encode trust credentials, and the Centrality reputation trust management to compute the reputation value using approaches describe in Section 2.2.2.1. The objective of this chapter is to enable the authentication and authorization framework to incorporate trustworthiness of the requester into its decision responses.

6.1 Design Specification

As mentioned above, the main objective of designing and implementing the Trust PDP is to enable trust evaluation based on different sources of trust information. Therefore the Trust PDP routines which includes functions to call trust services in a standardized XACML style interface has been implemented to address this challenge. However, before delving into details, in order to provide clarity the following two sections will describe the notation and terminology used throughout this chapter and will be followed-up by describing the XACML profile to be used by Master PDP and Trust PDP. These profiles requires no changes or modification to the standard XACML version 1.0, 1.1. or 2.0. as well as their additional profiles (e.g. privacy policy profile of XACML, XACML profile for RBAC).

The specification defines for XACML profile is built to meet the requirements for the Trust PDP and trust services in the Trust Management Architecture Design within TAS³ framework as specified in [TAS]. After an informal explanation of the building blocks that constitute the Trust PDP and trust services implementation,
we will continue to formally describe a full example which illustrates this building blocks with the TXACML semantics mentioned in Section 2.6. This specification also discusses how these building blocks are used to implement the various elements and requirements of the Trust Management Architecture Design within TAS³ framework as specified in [TAS].

6.1.1 Notation

In order to improve the readability, similar to OASIS standard notation which can be found in [OAS05a, OAS05c], the examples given in this document assume the use of the following XML Internal Entity declarations:

```
^lt;!ENTITY xacml "urn:oasis:names:tc:xacml:1.0:">
^lt;!ENTITY xml "http://www.w3.org/2001/XMLSchema#">
^lt;!ENTITY policy-combine
"urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:">
^lt;!ENTITY function "urn:oasis:names:tc:xacml:1.0:function:">
^lt;!ENTITY research-function
"http://example.org:8080/trustpdp/names/function#">
^lt;!ENTITY subject "urn:oasis:names:tc:xacml:1.0:subject:">
^lt;!ENTITY resource "urn:oasis:names:tc:xacml:1.0:resource:">
^lt;!ENTITY action "urn:oasis:names:tc:xacml:1.0:action:">
```

For example, &xml;string is equivalent to http://www.w3.org/2001/XMLSchema#string.

6.1.2 Terminology

As described in the various OASIS standards and profiles related to this specification [OAS05a, OAS05e, OAS05c, Opeb, OAS05d, OAS05b], the following terminology is used in addition to the semantics coined by this specification.

attribute - In this document, the term "attribute" refers to an XACML <Attribute>. An XACML <Attribute> is an element of an XACML Request and Policy which consist of an attribute name identifier, a data type identifier, an issuer (optional) and an attribute value. Each <Attribute> is associated either with one of the subjects (Subject Attribute), the resource (Resource Attribute), the action (Action Attribute), or the environment of the Request (Environment Attribute). Attributes are referenced in a policy by using an <AttributeSelector> (an XPath expression) or one of the following Attribute Designator: <SubjectAttributeDesignator>, <ResourceAttributeDesignator>, <ActionAttributeDesignator>, or <EnvironmentAttributeDesignator>.
permission - the right to perform some action on some resource, possibly only under certain specified conditions and purpose of which the resources are requested.

role - A job function within the context of an organization that has associated semantics regarding the right and responsibility which formally attached to the user and assigned to the role.

policy - A set of rules indicating which subjects are permitted to access which resources using which actions under which conditions. In Trust PDP settings, the policy could also be used to assess the trustworthiness of the subjects or other entities by specifying a set of trust metrics inside the rule tags.

discovery - A set of policies indicating which subjects are permitted to access which resources using which actions under specified policies. Starting from XACML 2.0 specification, the policy and policyset can be implemented using static references to another policy or policyset by introducing the new semantics to specification (e.g. <PolicySetIdReference>, <PolicyIdReference>).

MPS - Master PDP <PolicySet>. See Section 6.1.3.1.

Auth - Authorization PDP <PolicySet>. See Section 6.1.3.1.

Trust - Trust PDP <PolicySet>. See Section 6.1.3.1.

Priv - Privacy PDP <PolicySet>. See Section 6.1.3.1.

TSP - Trust Service <Policy>. See Section 6.1.4.2.

6.1.3 XACML Profile for Multiple PDPs

Although the specification of the Master PDP is not part of the requirements mentioned in Chapter 4, we still try to describe this specification in order to clarify the relation between Master PDP and Trust PDP. For explanatory purpose, the examples presented in this chapter limited to two different roles in MSP as mentioned in the scenario: (1) nurse, and (2) social worker. The roles presented here are not aimed to implement a full-fledged core and hierarchical role based access control as specified in [OAS05d].

6.1.3.1 Policies

This section specifies two types of policies and their semantics.
1. Master PDP `<PolicySet>` or MPS: a `<PolicySet>` that contains references to a different Sub PDPs indicating the role which subject requester applied, the requested resource and the action to be performed. MPS is a *meta policy* designed specifically to perform a *redirection* of a request to the appropriate Sub PDPs and to forward the decision response to the PEP. Each MPS references a single or multiple corresponding Sub PDP `<PolicySet>` and does not contain any `<Policy>` elements. It is out of the scope of this specification to determine whether or not the *meta policy* will provide conflict resolution feature in its architecture.

2. Sub PDP `<PolicySet>`: a `<PolicySet>` that contains the actual permission policies, trust preferences or privacy policies for different Sub PDPs associated with a given role. Sub PDP `<PolicySet>` instances must be reachable through the corresponding MPS. In addition, each Sub PDP specifies an identity information which indicates a unique identification of a specific Sub PDP. The identification of the Sub PDP `<PolicySet>` instances are described as follows:

(a) Authorization PDP `<PolicySet>` or Auth: a `<PolicySet>` that contains `<Policy>` elements and `<Rules>` which describes the resources and actions that subjects are permitted to access, along with any further conditions on that access.

(b) Trust PDP `<PolicySet>` or Trust: a `<PolicySet>` that contains `<Policy>` elements and `<Rules>` which describes the trust preferences on resources and actions that subjects are obligated to adhere.

(c) Privacy PDP `<PolicySet>` or Priv: a `<PolicySet>` that contains `<Policy>` elements and `<Rules>` which describes the privacy preferences on resources and actions that subjects are obligated to adhere.

In order to illustrate the specification, the following section will provide an example of the Master PDP `<PolicySet>`.

6.1.3.2 Example of Master PDP `<PolicySet>` for the Nurse Role

The following MPS contains references to different Sub PDPs associated with the *nurse* role which can be seen as references to different PolicySetId in the associated trust services.

```
<?xml version="1.0" encoding="UTF-8"?>
<PolicySet PolicySetId="MPS:nurse:role"
         PolicyCombiningAlgId="#policy-combine;=deny-overrides"
         xmlns="#&xacml:policy"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```
In this example, the policy set is composed of two references: Auth:A01:nurse:role which refers to an authorization PDP and Trust:T02:nurse:role which refers to a Trust PDP, using the combinator deny-overrides and applies only to requests where the resource has the name of Electronic Health Record, where the subject has the role of Nurse with an action Read.
6.1.4 XACML Profile of Trust PDP

This section determines an XACML Profile of Trust PDP which specifies the Trust PDP as an independent entity and expresses the details involving on how to receive a request and forward the response to the Master PDP.

6.1.4.1 Subject Attribute Roles

In contrary to the XACML Profile for RBAC [OAS05d], this specification solely expresses the roles as XACML Subject Attributes. This role attributes is expressed as a one "role attribute" where the value of this attribute indicates the role held by the subjects. For example, the identifier urn:example:attributes:role, might have the following possible role values: physician, nurse, intern, social worker and pharmacist. This is feasible since the XACML policy specified in this profile contains generic XACML specification and thereby can be attributed to a more expressive policy specification.

Since the application might also generates an access request initiated either from a human user or from the code bases on behalf of the user, this attribute may be associated with both initiators. For example, the human user has a SubjectCategory of &subject-category;access-subject; (this is the default category); the application that generates the access request has a SubjectCategory of &subject-category;codebase; and so on.

In addition, the role attribute might also be associated with an attribute assertion encoded in SAML contexts objects from Attribute Authority (AA).

6.1.4.2 Policies

Since the language supported by the Trust PDP (see Section 2.4.1) allows different policy expressions, it will lead to an increasing level of complexity. Thus, in order to manage complex policies, it will be useful to allow one policy to be made up of multiple sub-policies, which are evaluated, authored and maintained separately. Sub-policies also allows different entities or the stake holders to manage their own policies, therefore ensuring all relevant policies are taken into account before performing the trust decision. In the centralized context, the use of sub-policies allows policies to be grouped in useful ways into smaller units, making it easier and more efficient to maintain.

This section specifies two types of policies and their semantics which corresponds to the necessity of sub-policies as described above.

1. Trust PDP <PolicySet> or Trust: as mentioned in 6.1.3.1, Trust is a <PolicySet> that contains <Policy> elements and <Rules> which describes the trust preferences on resources and actions that subjects are obligated to adhere. The
<Policy> elements is a reference to the Trust Services <Policy>. Each Trust references a single or multiple corresponding Trust Services <Policy> and does not contain any <Rules> elements.

2. Trust Service <Policy> or TSP: a <Policy> that contains <Rules> elements which comprise trust preferences or metrics for specific Trust Services. In addition, each trust service specifies an identity information which indicates a unique identification of a specific trust service. The identification of the Trust Service <Policy> instances are described as follows:

(a) Structural Trust Service <Policy> or TSP:Rule: a <Policy> that contains <Rules> elements which describes the structural trust preferences.
(b) Behavioral Trust Service <Policy> or TSP:Rep: a <Policy> that contains <Rules> elements which describes the behavioral trust preferences.
(c) KPITM Trust Service <Policy> or TSP:Kpi: a <Policy> that contains <Rules> elements which describes the KPITM trust preferences.
(d) Logical Combination Trust Service <Policy> or TSP:Combine: a <Policy> that contains <Rules> elements which describes a logical combination of different trust preferences on resources and actions that subjects are obligated to adhere.
(e) Nested Trust Service <Policy> or TSP:Nested: a <Policy> that contains <Rules> elements which describes the nested trust preferences as specify in TAS requirements [TAS].

In order to illustrate this specification, the following example provides an example of the Trust PDP <PolicySet>. This example contains references to different TSPs associated with the nurse role.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<PolicySet PolicySetId="Trust:T02:nurse:role"
  PolicyCombiningAlgId=&policy-combine;deny-overrides" xmlns="&xacml;policy"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Description>
    Trust PDP policy set and references.
  </Description>
  <Target>
    <Subjects>
      <SubjectMatch MatchId=&function;string-equal">
        <AttributeValue DataType=&xml;string>Nurse</AttributeValue>
        <SubjectAttributeDesignator DataType=&xml;string>
          AttributeId=
            "urn:example:subject:role"/>
      </SubjectMatch>
    </Subjects>
  </Target>
</PolicySet>
```
In this example, the policy set is composed of two references: TSP:Rule:nurse:role which refers to a CTM trust service and TSP:Rep:nurse:role which refers to an RTM trust service, using the combinator deny-overrides and applies only to requests where the resource has the name of *Electronic Health Record*, where the subject has the role of *Nurse* with an action *Read*.

6.1.4.3 Rules and Functions

This section describes rules and functions based on the scenario given in Chapter 3 and their corresponding trust services.

1. Structural trust metric.

   (a) “function:credential”. This function provides a Ground type of credential release query which takes three arguments describing the trust metric, i.e. the required credential (issuer (string or list), role (string), subject (string)) and a number with the required trust score i.e. 1 (optional, assumed to be
1 if missing. Recall that the CTM trust metric is binary, only returning 0 or 1).

(b) 

   "&function;credential-subject-list". This function provides a Subject-Var type of credential release query which is similar to 

   "&function;credential". This function checks if the requester has a certain credential this function finds all subjects that have a given credential. This function takes two arguments; the credential issuer and role.

(c) 

   "&function;credential-attribute-list". This function provides an Attribute-Var type of credential release query which is similar to "&function;credential-subject-list". This function finds all credentials in the form of roles that belongs to a given subject. This function takes two arguments; the credential issuer and the subject.

2. Behavioral trust metric.

   (a) 

   "&function;reputation". This function takes three arguments; a string given the centrality measure to use, a string giving the entity that needs to be evaluated and a number(double) that indicates the trust score required of this entity.

   (b) 

   "&function;reputation-by-accredited-ratee". This function, which is similar to "&function;reputation", takes into account the feedback from trusted accredited ratees. Takes four arguments; a list of trusted raters (typically obtained from another trust function), the ratee, the centrality measure to use and the required score on this metric.

   (c) 

   "&function;reputation-of-issuer". This function, which is similar to "&function;reputation", finds potential issuers (e.g. of a credential) with a required reputation level. Typically used nested in another function. This function takes two arguments a string given the metric to use and a number (double) indicating the required score on this metric.

Appendix B illustrates the previous functions in a verbose XML format. As shown in Appendix B, several functions are specifically meant to be nested in others (e.g. reputation-of-issuer) or have others embedded (e.g. reputation-by-accredited-ratee). However, other may also be nested. For example, if we wish to accept credentials issued by an issuer with a good reputation we use the credential function where the first argument is a call to the reputation function. The nested functions need not be trust specific, any available function object can be used. A nested function call is achieved by the Apply Function attribute, for example
gets the identity of the subject (i.e. requester) as a string, which is useful for nearly all trust metrics.

The following section will illustrate how the informal specification mentioned in this chapter described formally with TXACML semantics described in Section 2.6. However, since additional functions have been defined as a result of specifying a new XACML profile, the semantics of the XACML policy specification described in Listing 2.1 needs to be extended.

```
S ::= POLICYSET

FCN-ID ::= reputation | reputation-by-accredited-ratee |
          reputation-of-issuer | credential |
          credential-attribute-list | credential-subject-list
```

Listing 6.1: TXACML Policy Specification Functions Extension Syntax

### 6.2 Examples

This section illustrates a complete example of various type of policies specified in this chapter with TXACML semantics describe in Section 2.6 and Listing 6.1. In addition, Appendix B illustrates the following scenario in a verbose XML format.

Assume that an MSP employs two type of roles, nurse and social worker. In this example, they are expressed as two separate values for a single XACML Attribute with AttributeId ”urn:example:attributes:role”, referred to from here on as the role Attribute.

Based on Bob’s trust preference, a nurse requires to have:

1. a medical degree and,

2. fairly good reputation among patients admitted to different home telemonitoring program in the public city hospital.

To finally come into a trust decision, Bob could use deny-overrides policy combining algorithm to capture the requirements since both requirements have to be true. This particular example captures Type1 trust policy statement.

In the other hand, a social worker requires to have:

1. a first-aid certificate or,
2. good reputation among patients admitted to different home telemonitoring program in the public city hospital.

To come into a trust decision, Bob uses a logical OR function to capture the requirements. This particular example captures Type2 trust policy statement.

As an alternative, a social worker might also require to have:

1. a first-aid certificate from a reputable organization and,

2. good reputation among patients admitted to different home telemonitoring program in the public city hospital.

To finally come into a trust decision, Bob could use deny-overrides policy combining algorithm to capture the requirements since both requirements have to be true. Since one of the requirement employs nested trust metric, this particular example captures Type3 trust policy statement.

### 6.2.1 Requests

The following examples formalized the XACML request contexts which are used to formulate decision request for two different roles, namely nurse and social worker, to a given resources called electronic health record with action read. In addition, Appendix A illustrates the following requests in a verbose XML format.

(Request

```
((Attribute subject-id string (AttributeValue alice string))(Attribute role string (AttributeValue nurse string)))
((Attribute resource-id string (AttributeValue electronicHealthRecord string)))
((Attribute action-id string (AttributeValue read string))))
```

(Request

```
((Attribute subject-id string (AttributeValue bob string))(Attribute role string (AttributeValue socialworker string)))
((Attribute resource-id string (AttributeValue electronicHealthRecord string)))
((Attribute action-id string (AttributeValue read string))))
```

### 6.2.2 Trust PDP <PolicySet>s and <Policy>s

The following examples formalized the XACML policy based on policy language described in Section 2.4.2 for two different roles, namely nurse and social worker, to a given resources called electronic health record with action read. In order to avoid the not-applicable responses as mentioned in Section 2.6.1, the following policies use additional rule which specifies that if there are not any applicable policy applies to a given request, it will render a deny decision result.
6.2.2.1 Structural Trust Service <Policy> for the Nurse Role

The following TSP contains a rule which specifies a structural trust metric associated with the nurse role based on the scenario described in Section 6.2.

(Policy Permit–Overrides
  ((Any)
   ((Match (AttributeValue electronicHealthRecord string)(
     AttributeDesignator resource-id string) type-equal))
   (Any))
  (Rule
   (((Match (AttributeValue nurse string)(AttributeDesignator role string) type-equal))
    (Any)
    ((Match (AttributeValue read string)(AttributeDesignator action-id string) type-equal)))
   Permit
   (Condition credential (AttributeValue greenMedicalService string )
    (AttributeValue nurse string)
    (Apply type-one-and-only
     (AttributeDesignator subject-id string))
    (AttributeValue 1.0 double)))
  (Rule
   (((())
     (())
     (()) Deny))

In this example, the policy is composed of two rules using the combinator permit-overrides and applies only to requests where the resource has the name of electronicHealthRecord. The first rule permits requests where the subject has the role of nurse with an action read regardless of the resource. The second denies all requests. This example specifies a condition that the subject has a credential nurse from greenMedicalService.

6.2.2.2 Behavioral Trust Service <Policy> for the Nurse Role

The following TSP contains a rule which specifies a behavioral trust metric associated with the nurse role based on the scenario described in Section 6.2.

(Policy Permit– Overrides
  ((Any)
   ((Match (AttributeValue electronicHealthRecord string)(
     AttributeDesignator resource-id string) type-equal))
   (Any))
  (Rule
   (((Match (AttributeValue nurse string)(AttributeDesignator role string) type-equal))

In this example, the policy is composed of two rules using the combinator permit-overrides and applies only to requests where the resource has the name of electronicHealthRecord. The first rule permits requests where the subject has the role of nurse with an action read regardless of the resource. The second denies all requests. This example specifies a condition that the subject has a reputation value equal or larger than 0.8, computed with pagerank.

6.2.2.3 Single Type Trust PDP <PolicySet> for the Nurse Role

The following Trust PDP <PolicySet> contains a single type representation of TAS policy language associated with the nurse role based on the scenario described in Section 6.2. The Trust PDP <PolicySet> combines the decision result using Deny-Overrides algorithm as described in Section 2.6.1.

(PolicySet Deny- Overrides
(Any)
(((Match (AttributeValue electronicHealthRecord string)(
    AttributeDesignator resource-id string) type-equal))
(Any))
(Policy Permit- Overrides
((Any)
(((Match (AttributeValue electronicHealthRecord string)(
    AttributeDesignator resource-id string) type-equal))
(Any))
(Rule
(((Match (AttributeValue nurse string)(AttributeDesignator role
    string) type-equal))
(Any)
(((Match (AttributeValue read string)(AttributeDesignator
    action-id string) type-equal)))
Permit
(Condition credential (AttributeValue greenMedicalService string)
    (AttributeValue nurse string)
    (Apply type-one-and-only
In this example, the policy set is composed of two policies using the combinator deny-overrides. The first policy is composed of two rules using the combinator permit-overrides and applies only to requests where the resource has the name of electronicHealthRecord. The first rule permits requests where the subject has the role of nurse with an action read regardless of the resource. The second denies all requests. This policy specifies a condition that the subject has a credential nurse from greenMedicalService. The second policy is composed of two rules using the combinator permit-overrides and applies only to requests where the resource has the name of electronicHealthRecord. The first rule permits requests where the subject has the role of nurse with an action read regardless of the resource. The second denies all requests. This policy specifies a condition that the subject has a reputation value equal or larger than 0.8, computed with pagerank.

### 6.2.2.4 Trust Service Combination <Policy> for the Social Worker Role

The following TSP contains a rule which specifies a logical combination trust metric associated with the social worker role.
In this example, the policy is composed of two rules using the combinator permit-overrides and applies only to requests where the resource has the name of electronicHealthRecord. The first rule permits requests where the subject has the role of socialworker with an action read regardless of the resource. The second denies all requests. This policy specifies a condition that the subject has a credential socialworker from greenMedicalService and has a reputation value equal or larger than 0.8, computed with pagerank.

6.2.2.5 Trust Service Nested <Policy> for the Social Worker Role

The following TSP contains a rule which specifies a nested trust metric associated with the social worker role.

(Policy Permit–Overrides
((Any)
  ((Match (AttributeValue electronicHealthRecord string)
      (AttributeDesignator resource–id string) type–equal))
  (Any))
(Rule
  (((Match (AttributeValue socialworker string)
      (AttributeDesignator role string) type–equal))
    (Any)
    ((Match (AttributeValue read string)
      (AttributeDesignator action–id string) type–equal)))
  Permit
  (Condition and
   (Apply credential (AttributeValue greenMedicalService string)
    (AttributeValue socialworker string)
    (Apply type–one–and–only
     (AttributeDesignator subject–id string))
    (AttributeValue 1.0 double))
   (Apply reputation (Apply type–one–and–only
     (AttributeDesignator subject–id string))
    (AttributeValue pagerank string)
    (AttributeValue 0.8 double))))
  (Rule
   (((()))
    (())
    ((()) Deny)))

In this example, the policy is composed of two rules using the combinator permit-overrides and applies only to requests where the resource has the name of electronicHealthRecord. The first rule permits requests where the subject has the role of socialworker with an action read regardless of the resource. The second denies all requests. This policy specifies a condition that the subject has a credential socialworker from greenMedicalService and has a reputation value equal or larger than 0.8, computed with pagerank.
In this example, the policy is composed of two rules using the combinator permit-overrides and applies only to requests where the resource has the name of electronicHealthRecord. The first rule permits requests where the subject has the role of socialworker with an action read regardless of the resource. The second denies all requests. This policy specifies a condition that the subject has a credential socialworker from greenMedicalService which has a reputation value equal or larger than 0.8, computed with pagerank.

6.2.2.6 Behavioral Trust Service <Policy> for the Social Worker Role

The following TSP contains a rule which specifies a behavioral trust metric associated with the social worker role.

(Policy Permit-Overrides
  ((Any)
    ((Match (AttributeValue electronicHealthRecord string)(
        AttributeDesignator resource-id string) type-equal))
    (Any))
  (Rule
    (((Match (AttributeValue socialworker string)(AttributeDesignator role string) type-equal))
        (Any)
      ((Match (AttributeValue read string)(AttributeDesignator action-id string) type-equal)))
    Permit
    (Condition reputation (Apply type-one-and-only
        (AttributeDesignator subject-id string))
      (AttributeValue pagerank string)
      (AttributeValue 0.8 double)))
  (Rule
In this example, the policy is composed of two rules using the combinator permit-overrides and applies only to requests where the resource has the name of electronicHealthRecord. The first rule permits requests where the subject has the role of socialworker with an action read regardless of the resource. The second denies all requests. This example specifies a condition that the subject has a reputation value equal or larger than 0.8, computed with pagerank.

### 6.3 The Trust PDP Implementation

The Trust PDP implementation extends Sun’s (prototype) XACML 2.0 PDP implementation [Sun] and runs on Java (runtime environment 6 or later). This implementation provides an integration of and interfaces to a collection trust services featured by the TAS\(^3\) Trust Policy Management architecture [TAS]. However, the implementation of the trust services in this thesis limited to two different trust services, a RTM service and a CTM service, and supporting tools. It is intended to be introduced as a proof-of-concept for the trust management integration approach. This section focuses on the Trust PDP, CTM service and its corresponding components, and the RTM service.

In a component view, the components of the Trust PDP implementation are depicted in Figure 6.1. Generally, this figure consists of two distinct interfaces, the Trust PDP interface and the Trust Service interface along with the helper components (e.g. logger, configuration store, exception handlers).

This implementation supports the TAS\(^3\) policy language (see section 2.4.2) which allows expressing Type 1, 2 or 3 policies. Evaluation of these metrics is supported for 1, 2 and limited Type 3 policies. The Trust PDP callback option, needed for more intricate level 3 policies is not yet supported in this implementation. Furthermore, as mentioned in Section 2.4.1 the overall trust policy is expressed using an XACML style policy set where conditions are used to express the trust requirements. The service call itself is provided by a Function object which exists for each supported trust service. AttributeValue elements are used to add the arguments that need to be passed to the Trust Service, such as the trust metric to use. The Function object itself indicates the corresponding trust services and provides a way to contact these services.

In more details, the following sub sections describe the Trust PDP and trust services implementation mentioned above.
6.3.1 The Trust PDP Interface

As shown in the Figure 6.1, extending the `com.sun.xacml.pdp` object provides a TrustPDP object with a PDP subtype which becomes useable in the context specified by the requirements presented in Chapter 4 and where a Sun XACML PDP can be used. To build a Trust PDP trust policy engine, we start by instantiating this object.

In a detailed class view, the collection of class diagrams of the Trust PDP implementation is depicted in Figure 6.2. This figure shows that the Trust PDP has an interface that is implemented by TrustPDPImpl object using evaluate method. Moreover, this figure also shows that the trust service interface is implemented by both trust services (e.g. CTMEngine object and RTMEngine Object).

6.3.2 TAS³ Trust Services Interfaces

An important part in the trust services implementation is the Function object which provides a wrapper around the trust service call. It extends the `com.sun.xacml.condition.FunctionBase` and `com.sun.xacml.condition.Function` interface to enable trust metric evaluation to become part of the standard XACML policy evaluation framework. As a trust service may offer different functions (e.g. for different types of metrics with different sets of arguments) a FunctionCluster object is used to bundle the functions supported by a trust service. The currently supported functions which
Figure 6.2: Trust PDP class diagram
described in detail in Section 6.1.4.3 are presented as follows.

1. reputation function - provided by eu.tas3.trustpdp. centralityRTM.

2. credential function - provided by eu.tas3.trustpdp.polipoctm.PolipoCTM.

3. reputation-of-issuer function - provided by eu.tas3.trustpdp.centralityRTM.

4. reputation-by-accredited-ratee function - provided by eu.tas3.trustpdp. centralityRTM.

5. credential-subject-list - provided by eu.tas3.trustpdp.polipoctm.PolipoCTM.

6. credential-attribute-list - provided by eu.tas3.trustpdp.polipoctm.PolipoCTM.

A detailed use of the function is specified in Section 6.1.4.3 which can be read in the Appendix B.

As mentioned in the previous section, this implementation does not yet include the trust service callback mechanism. Still, tough not as general/generic, the nesting of policies using specialized functions offers an efficient method of evaluating several classes of multilevel policies. The approach shows in this implementation use a filtering mechanism that has an ability to filter a set of entity as mentioned in Section 4.2.

To support additional trust services the functions for the different supported metrics should be created. The function implementation (multiple functions may be implemented by the same object) is responsible for constructing the query from the arguments supplied, calling the service and comparing its returned value to the required trust level. By adding a wrapper class which implements the com.sun.xacml.cond.FunctionCluster interface to the trust service configuration file the new functions become available for use in the Trust PDP. Much of this is straightforward adaption from the code for the existing supported trust services.

This implementation uses simple mechanism to add the RTM service to the Trust PDP interface by simply add the server object to the trust service configuration file

<trustService class="eu.tas3.trustpdp. centralityRTM"/>

Moreover, adding the CTM service is done by simply add the server object to the trust service configuration file as well

<trustService class="eu.tas3.trustpdp. polipoctm.PolipoCTM"/>

Detailed information related to Trust PDP configuration for different scenario of policy specification can been seen in Appendix C.
6.3.2.1 The POLIPO CTM Trust Service Interfaces

The POLIPO CTM trust service is aimed to verify the trustworthiness of entity by gathering and combining credentials which express trust and trust delegation statements by different authorities. Beside POLIPO, multiple options exist for the CTM service as mentioned in Section 2.2.2.1 and [TAS] (e.g. TuliP, RT0). This section describes the interfaces and main components of the POLIPO CTM trust service.

The POLIPO CTM Trust Service contains the following components (see Section 6.3): the CTM engine, the SAML2 engine and the Fedora Repository engine. Beside the core CTM engine, each of this engine is built to address credential release query processes.

To date, POLIPO CTM service prototype implementation provides a Java interface which corresponds to SAML attribute queries. The trust predicates are provided as SAML assertions. The answer is returned in the form of an SAML response containing one or more assertions, each of which corresponds to the credential atom. The POLIPO CTM service prototype uses version 2.0 of the SAML standard [OAS05e].

Since query to credential release differs in how to formulate the query parameters, Figure 6.4 and 6.5 illustrate how the interaction happens between each components in the POLIPO CTM trust service implementation. A detailed examples of credential release queries, POLIPO policy and assertion can be seen in Appendix D.

As shown in Figure 6.4 and 6.5, beside directly fetches the credentials in POLIPO CTM service, this interface also provides a cash for credentials using a fedora repository. In this implementation, the REST API interface as mentioned in Section 2.2.1.3 is used to fetch the credential from the fedora repository server using query specified
Figure 6.4: Credential release ground query sequence diagram in POLIPO CTM client
Figure 6.5: Credential release subject-var and attribute-var sequence diagram in POLIPO CTM client
in the REST API format. However, before the trust service implementation is able to get the credentials from the fedora repository server, the credential has to be ingested to the repository using FOXML Digital Object Format as described in Section 2.2.1.3. To facilitate this interaction, the SAML elements needs to be mapped to the FOXML DCMES elements. The following table illustrates the mapping schema.

Table 6.1: Mapping SAML elements to FOXML DC Metadata Element

<table>
<thead>
<tr>
<th>SAML Element</th>
<th>FOXML DC Metadata Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issuer</td>
<td>Publisher</td>
</tr>
<tr>
<td>Subject</td>
<td>Subject</td>
</tr>
<tr>
<td>Attribute</td>
<td>Relation</td>
</tr>
</tbody>
</table>

In addition, Appendix E provides a detail explanation of objects and queries use in this setting.

6.3.2.2 The Centrality RTM Trust Service Interfaces

The aim of the Centrality RTM trust service is to calculate reputation values based on feedback data that is gathered over time. This implementation allows the user to choose their own method and describes how feedback is combined to calculate a reputation via trust policy Function objects. In this thesis, an external trusted party runs a feedback and reputation service which gathers feedback on a various aspect of services.

As shown in Figure 6.6, the Centrality RTM trust service interface has two distinct interfaces: one to the Trust PDP and one to the trust information collection point, which has an interface to the Authentication Service. A service request to the reputation trust service is formulated in the trust metric language defined in Section 2.2.2.1. In the end, the service request will result a trust level value computed in the trust information collection point based on the behavioral information. The Centrality RTM service provides an interface for users giving feedback on service providers. All feedback data is stored in a relational database management system such as Oracle Database\(^1\) or PostgreSQL\(^2\).

Figure 6.7 is used to illustrate the above interaction between the Centrality RTM trust service and the trust information collection point when user is requesting to evaluate the reputation of an entity.

In addition, Appendix F provides a detail examples on how to formulate reputation feedbacks and queries.

\(^1\)http://www.oracle.com/
\(^2\)http://www.postgresql.org/
Figure 6.6: Centrality RTM component diagram

Figure 6.7: Reputation sequence diagram in Centrality RTM client
Lastly, in order to demonstrate the behaviors of the Trust PDP and the trust service interfaces, we develop a simple prototype of the Trust PDP to simulate the interaction between decision requests and their applicable trust policies which in turn render a decision response.

Note that in order to be able to run the Trust PDP implementation, we have to consider the hardware and software prerequisites of this system as described in Appendix G.

Finally, additional work has been done to add the functionality of the Trust PDP by deploying Trust PDP web service. A web-service interface to the (java based) Trust PDP is provided by integration in the PERMIS XACML code, guaranteeing an equal WSDL support/interpretation of the SOAP messages. This implementation uses the PERMIS\(^3\) stand-alone server adapted to call the Trust PDP rather than the Sun's XACML PDP.

\(^3\)PERMIS authorization system, available at http://sec.cs.kent.ac.uk/permis/
6.4 Additional Scenarios

To illustrate a different scenario of specifying the trust policy statement and its metrics, the following illustration depicts the a more complex situation where Bob could express his policy by adding more information in the Behavioral Trust Service metric.

The reputation service runs by a public city hospital where Bob lives allows anyone from different home telemonitoring programs (e.g. heart failure and stroke) provides feedback on their experience with a medical service provider. As suggested by his care manager, Bob has decided to trust only the medical service providers who has a first-aid certificate with a good reputation amongst member of his home telemonitoring program. As shown in Figure 6.9, this situation is captured in a policy which combines two different trust services.

The MSP officer credential is capture on a single structural trust metric policy while the reputation of the MSP officer is capture a nested trust metric policy. To evaluate Bob’s policy the Trust PDP will perform the following steps:

1. After receiving a request from the Master PDP,

2. the Trust PDP queries the Structural Trust Service to determine which are the trusted users (P4). After the Structural Trust Service gets the appropriate credential,

3. it returns a permit decision to the Trust PDP. However to sufficiently returns a trust decision response to the Master PDP,
4. the Trust PDP needs to call the Behavioral Trust Service to find the reputation of P4. To calculate this reputation the Behavioral Trust Service needs to know which of its feedback items on P1 are from trusted users (i.e. home telemonitoring program for patients with CHF). To find this,

5. the Behavioral Trust Service will call the Trust PDP through the call-back mechanism,

6. then the Trust PDP will forward the request to the Structural Trust Service to determine which are the trusted users (i.e. the home telemonitoring program for patients with CHF members),

7. and return this to the Trust PDP. Subsequently,

8. the Trust PDP forwards the result to the Behavioral Trust Service as a response to a call-back mechanism. The Behavioral Trust Service now has a complete information to compute the reputation of P4, which turns out to be sufficiently high,

9. and returns a Permit decision to the Trust PDP. Finally,

10. after combining the decisions, the Trust PDP is able to return a Permit response to the Master PDP.

6.4.1 Additional Trust Service Nested <Policy> for the Social Worker Role

Based on Bob’s trust preference, a social worker requires to have:

1. a first-aid certificate and,

2. good reputation among patients admitted to the same program as Bob’s home telemonitoring program in the public city hospital.

To finally come into a trust decision, Bob could use deny-overrides policy combining algorithm to capture the requirements since both requirements have to be true. Since one of the requirement employs nested trust metric, this particular example also captures Type3 trust policy statement.

The following TSP contains a rule which specifies a structural trust metric associated with the social worker role.

(Policy Permit−Overrides
  ((Any)
   ((Match (AttributeValue electronicHealthRecord string))(AttributeDesignator resource−id string) type−equal)))
In this example, the policy is composed of two rules using the combinator permit-overrides and applies only to requests where the resource has the name of electronicHealthRecord. The first rule permits requests where the subject has the role of socialworker with an action read regardless of the resource. The second denies all requests. This policy specifies a condition that the subject has a credential socialworker from greenMedicalService.

In addition, the following TSP contains a rule which specifies a nested trust metric associated with the social worker role.

(Policy Permit–Overrides

(Any)

((Match (AttributeValue electronicHealthRecord string) (AttributeDesignator resource–id string) type–equal))

(Any))

Rule

(((Match (AttributeValue socialworker string) (AttributeDesignator role string) type–equal))

(Any)

((Match (AttributeValue read string) (AttributeDesignator action–id string) type–equal)))

Permit

(Condition reputation–by–accredited–ratee (Apply credential–subject–list

(AttributeValue greenHospitalHealthcareProgram string)

(AttributeValue member string))

(Apply type–one–and–only

(AttributeDesignator subject–id string))

(AttributeValue pagerank string)

(AttributeValue 0.8 double)))

(Rule
In this example, the policy is composed of two rules using the combinator permit-overrides and applies only to requests where the resource has the name of electronicHealthRecord. The first rule permits requests where the subject has the role of socialworker with an action read regardless of the resource. The second denies all requests. This policy specifies a condition that the subject has a reputation value equal or larger than 0.8, computed with pagerank by a list of ratee who has as a credential member from greenHospitalHealthcareProgram.
Chapter 7
Validation

This chapter reviews the design and implementation of the Trust PDP system based on requirements specified in Chapter 4. Additional evaluation is performed to analyze the Trust PDP system compared to other trust management integration solutions, and finally this chapter addresses the security issues correspond to the Trust PDP implementation.

7.1 Requirement fulfillment of the Trust PDP system

In this section, we review the Trust PDP implementation based on relevant requirements presented in Chapter 4. Beside discussing the requirement fulfillment, the following explanation also provides the means of describing the gaps between the following application prototypes: (1) Trust PDP, (2) POLIPO CTM trust service, and (3) Centrality RTM trust service with the TAS\textsuperscript{3} trust policy architecture.

The following explanations discuss the requirement fulfillment of the Trust PDP system.

Unified Representation. The implementation of the Trust PDP has shown to be able to accept trust evaluation request in the form of XACML request contexts and respond with XACML response contexts. In addition, the trust service client interfaces for both POLIPO CTM and Centrality RTM have been developed to allow the trust metrics associated with each trust service to be embedded in the in XACML policy without modifying or extending the XACML specification (see Section 6.1)

Flexible Policy Composition. The design of new XACML functions has been adapted to satisfy various policy composition based on the language shown
in Section 2.4.2. Section 6.1.4.3 shows additional XACML functions to cope with this requirement and corresponds to the scenario presented in Section.

**Flexible Algorithm.** The current build of the Trust PDP system supports only filtering mechanism using standard XACML functions and exploits the XACML policy structures, which inherently allows nested calls via sub policies and functions. Although a callback mechanism is not yet supported, both POLIPO CTM trust service and Centrality RTM trust service are able to encode XACML policy statement into each of their own languages. This is achieved by utilizing their own query statements, for example the POLIPO CTM could encode the XACML policy statement (in the form of attributes) into SAML attribute value, while the Centrality RTM could also encode the XACML policy statement into its SQL statements.

**Trusted Entities Discovery.** This feature relates to the trusted entities discovery features implemented in the (top level) trust service. The design of the Trust PDP interface does not restrict any feature supported by the trust services, since the Trust PDP design is aimed to support the discovery of the trust services.

The following explanations discuss the requirement fulfillment of the trust services attached to the current prototype of the Trust PDP system, namely POLIPO CTM trust service and Centrality RTM trust service.

**Unified Representation of Trust Metrics.** The current trust service clients implementations support both the POLIPO CTM trust metric and the Centrality RTM trust metric as described in Section 2.2.1.1 and 2.2.2.1. The representation of both trust service metrics is provided in XACML policy attributes. This XACML attributes provides a generic and robust representation of the trust service metrics since XACML standard already supports a wide variety of data types and functions. Nevertheless, there are already standards groups working on extensions and profiles that will increase the number of ways that XACML can be used.

**Parameterized Trust Metrics.** The current build of the trust service interface is designed to provide different trust metrics and to support different type of trust policy language (see Section 2.4.2). This is achieved by encoding XACML policy statement into both POLIPO CTM trust service and Centrality RTM trust service languages.

**Single and Group Entity Evaluation.** Both the POLIPO CTM trust service interface and Centrality RTM trust service interface provide supports for single and multiple entity trust level evaluation (see Section 6.3.2. The returned trust
level values are provided in an XACML standard numeric value for the single entity evaluation and an XACML bag value for the multiple entity evaluation.

**Trusted Entities Discovery Evaluation.** The last release of the trust services which were used in this thesis did not yet support the trusted entity discovery evaluation. However, the support for this feature is already addressed in the POLIPO CTM trust service future releases. Beside depends on the top level trust service feature, a subset of this discovery feature is already implemented in the POLIPO CTM trust service client. The POLIPO CTM trust service client provides an internal support for querying the trusted entity stored in the repository system managed by the POLIPO CTM trust service client.

### 7.2 The Trust PDP Evaluation

The following aspects are discussed to evaluate the Trust PDP design and implementation presented in this thesis, based on the various approaches to trust management integration as described in Section 2.5.

**Supported features** - Most TM integration solution provides comprehensive approach to combine structured and unstructured trust management by specifying the same language for both structured and unstructured trust metrics. QuanTM and Trust PDP, however, have specified different languages for both structured and unstructured trust management and introduces the concept of decision manager which has an objective to address result conflicts. When handling results conflict, Trust PDP has more complex features of combining algorithms since TAS3 trust policy language utilizes the OASIS XACML combining algorithms. In addition, Trust PDP solution also specifically addresses to incorporate novel trust services into its architecture.

**Policy language** - As described in Section 2.5, each TM integration solution implements their own policy language. However, QuanTM and especially Trust PDP use different languages for both structured and unstructured trust management. In addition each trust service in the Trust PDP independently implements their own policy language. The interaction between Trust PDP and trust services are done via a unifying interface (see Chapter 4).

**Syntax** - Although most solution addresses various syntaxes, only CTM and TAS3 Trust PDP that address sequential policy composition. While CTM solution relies on the filtering mechanism, the Trust PDP solution goes beyond that feature by defining a call-back mechanism, which provides single trust metric for defining nested policy. However, the current build of the Trust PDP system
supports the same feature as CTM solution. In addition to the nested policy language presented in Chapter 6, the Trust PDP could also provide a more complex sequential policy composition. For example, the following policy easily implements this feature.

```
Policy Permit-Overrides
  ((Any)
   ((Match (AttributeValue electronicHealthRecord string)
       (AttributeDesignator resource-id string) type-equal))
   (Any))
  (Rule
   ((((Match (AttributeValue socialworker string)
       (AttributeDesignator role string) type-equal))
     (Any)
     (((Match (AttributeValue read string)
         (AttributeDesignator action-id string) type-equal)))
   Permit
   (Condition reputation-by-accredited-ratee (Apply credential-subject-list
     (Apply reputation-of-issuer
      (AttributeValue pagerank string)
      (AttributeValue greenHospitalHealthcareProgram string)
      (AttributeValue 0.8 double))
     (AttributeValue member string))
     (Apply type-one-and-only
      (AttributeValue subject-id string))
     (AttributeValue pagerank string)
     (AttributeValue 0.8 double)))
   (Rule
     (((() ))
     ((() ) Deny))

This policy specifies that, based on Bob’s trust preference, a medical service provider requires to have a good reputation among reputed patients (members) admitted to the same program as Bob’s home telemonitoring program in the public city hospital.

To finally come into a trust decision, Bob could use permit-overrides policy combining algorithm to capture the requirements since both requirements have to be true. Since one of the requirement employs nested trust metric, this particular example also captures Type3 trust policy statement.

Results conflict resolution - Most language in TM integration solution does not need conflict resolution due to the nature of the language. Nevertheless, QuanTM and Trust PDP use conflict resolution mechanism which is needed by their decision manager components. In this feature, the Trust PDP conflict resolution
mechanism provides more dynamic conflict resolution using trust policy language and XACML combining algorithm mentioned in Section 2.6.1.

In order to briefly explained the above discussions, Table 7.1 summarized the different trust management integration approaches as mentioned in Section 2.5 along with the Trust PDP solution.

<table>
<thead>
<tr>
<th>TM Integration Solution</th>
<th>Features</th>
<th>Policy Language</th>
<th>Conflict Resolution</th>
<th>Combining Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid Trust Model</td>
<td>Comprehensive</td>
<td>Variant of LTL logic [KNS05]</td>
<td>Simple disjunction, conjunction, constraint</td>
<td></td>
</tr>
<tr>
<td>QuanTM</td>
<td>Independent with decision manager</td>
<td>KeyNote, TDG [WAC+09]</td>
<td>Dynamic, simple</td>
<td></td>
</tr>
<tr>
<td>CTM</td>
<td>Comprehensive</td>
<td>RT family language with extension [LY09]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTM and CTM Integration</td>
<td>Comprehensive, supports external trust services</td>
<td>PROTUNE [BO05]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AC with RTM and CTM Management for Globus</td>
<td>Comprehensive</td>
<td>RTML with trust extension [MMV05]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trust PDP</td>
<td>Independent with decision manager, supports external and new trust services</td>
<td>TAS³ Trust Policy Language [TAS] (XACML [OAS05a]), POLIPO [TSZE09a], TAS³ Reputation System [TAS]</td>
<td>Dynamic, supports complex conflict resolution</td>
<td>Simple disjunction, conjunction, logical combination, sequential composition (with callback mechanism*), XACML combining algorithms</td>
</tr>
</tbody>
</table>

*not yet supported
7.3 Security Considerations

In general, the primary mechanism for building trust relation between system entities are by having a pre-existing trust relationship, typically involving a Public Key Infrastructure (PKI). Whilst use of a PKI is not mandated in this design, it is highly recommended. The following recommendations are an overview of what are needed to provide a predefined trust relations:

- HTTP over SSL 3.0 or TLS 1.0 is recommended where message integrity and message confidentiality are required.

- When assertions or credentials are fetched from the POLIPO CTM trust service then it is required to digitally signed the response message using the XML digital signature standard.

- Assertion or credential signature should be used to verify whether or not the credentials have been modified or tampered.
Chapter 8

Conclusions and Future Work

This chapter concludes and summarizes the design and implementation of the Trust PDP system and also suggests the future works to enhance the current Trust PDP system.

8.1 Results and Conclusions

We have presented a Trust PDP design and implementation that combines different notions of trust in a single framework. Although the overall trust policy architecture as mentioned in Section 2.4.1 is not yet implemented, the current implementation of Trust PDP still offers a flexible and extendable way of combining different sources of trust. Adoption of a broad notion of trust policy in one single application framework, encompassing not only CTM and/or RTM approaches in TM, is considered to be the potential contribution of this implementation. The evaluation based on the requirements shows that the Trust PDP implementation is able to meet (partially) the requirements specified by TAS\textsuperscript{3} [TAS].

The Trust PDP system also specifies and implements the required interactions to inquire a trust decision from different trust services. By conducting various experiments and use case analysis based on healthcare scenario illustrated in Chapter 3, The Trust PDP system has shown that the increased expressiveness through levelled policies is applicable to achieve for a particular setting (see Section 6.4).

In addition, the Trust PDP system is also able to implement different trust metrics from different trust management systems into trust policies using XACML policy language. The use of XACML in valuing trustworthiness of an entity (trust policy) has shown to be possible since XACML standard provides a generic and powerful language that supports a wide variety of data types, functions, and rules about combining the results of different policies.
8.2 Future Work

Following are some possibilities which can facilitate future work in enhancing the Trust PDP system and the trust policy architecture:

**Call-back Mechanism** - By implementing the call-back mechanism, the Trust PDP will fully conform the TAS\(^3\) trust policy architecture as described in Section 2.4.1. This mechanism allows the Trust PDP: (1) embedding different trust policies inside the XACML policy, (2) receiving and (3) forwarding this policy into its corresponding trust service.

**Results Caching** - Cache mechanism is used to allow the decisions to be made based upon previously computed and cached results. Especially in the system where an updated information is achieved in timely basis.

**Loop Detection Mechanisms** - In declarative policy specification, loops may easily occur and should not be considered as errors. If not handled accordingly, such loops may end up in nonterminating evaluation. In reality, policies are complex and large in number which increases the risk of loops and nontermination during dynamic policy evaluation.

**Policy Specification Tools** - Tools like the UMU XACML policy administration tool\(^1\) provides an easy-to-use application to help policy writers. This is important since errors in the policy specification and implementation will allow inappropriate access to the resources.

---

\(^1\)http://sourceforge.net/projects/umu-xacmleditor/
Bibliography


[MMV05] Fabio Martinelli, Paolo Mori, and Anna Vaccarelli. Towards continuous usage control on grid computational services. Autonomic and Autonomous Systems and International Conference on Networking and Services, Joint International Conference on, 0:82, 2005.


Appendix A

XACML Requests and Response

The following two sections illustrate the examples of XACML encoded requests and response context in a verbose XML format.

A.1 XACML Requests for Nurse and Social Worker

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:xacml:1.0:context"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Subject>
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
            xsi:nil="true"/>
        <AttributeValue>Alice</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:example:subject:role" xsi:nil="true"/>
    <AttributeValue>Nurse</AttributeValue>
</Subject>
<Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
        xsi:nil="true"/>
    <AttributeValue>Electronic health record</AttributeValue>
</Resource>
<Action>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
        xsi:nil="true"/>
    <AttributeValue>Read</AttributeValue>
</Attribute>
</Request>
```
A.2 XACML Response

```xml
<Response>
  <Result ResourceId="electronic health record">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
</Response>
```
Appendix B

XACML Functions, <PolicySet>s, <Policy>s and Rules

The following sections illustrate the examples of XACML functions and policies in a verbose XML format.

B.1 Structural Trust Metric Functions

```xml
<Rule RuleId="socialworker:role:rueltrustmetric" Effect="Permit">
  <Condition FunctionId="&research−function;credential">
    <AttributeValue DataType="&xml;string">Green Medical Service</AttributeValue>
    <AttributeValue DataType="&xml;string">Social Worker</AttributeValue>
    <Apply FunctionId="&function;string−one−and−only">
      <SubjectAttributeDesignator DataType="&xml;string" AttributeId="&subject;subject-id"/>
    </Apply>
  </Condition>
  <AttributeValue DataType="&xml;double">1</AttributeValue>
</Rule>

<Rule RuleId="nurse:role:nestedtrustmetric" Effect="Permit">
  <Condition FunctionId="&function;string−is−in">
    <AttributeValue DataType="&xml;string">Alice</AttributeValue>
    <Apply FunctionId="&research−function;credential−subject−list">
      <AttributeValue DataType="&xml;string">Green Hospital Healthcare Program</AttributeValue>
      <AttributeValue DataType="&xml;string">Nurse</AttributeValue>
    </Apply>
  </Condition>
</Rule>
```
B.2 Behavioral Trust Metric Functions
<Apply>
  <AttributeValue DataType="&xml; string">pagerank</AttributeValue>
</Apply>

<AttributeValue DataType="&xml; double">0.8</AttributeValue>

</Condition>
</Rule>

B.3 <PolicySet> and <Policy>s for Nurse

```xml
<?xml version="1.0" encoding="UTF-8"?>
<PolicySet PolicySetId="Trust:T02:nurse:role"
  PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides"
  xmlns="urn:oasis:names:tc:xacml:1.0:policy"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Description>
    Trust PDP policy set and references.
  </Description>
  <Target>
    <Subjects>
      <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Nurse</AttributeValue>
      </SubjectMatch>
      <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string">
        <AttributeValue DataType="urn:example:subject:role"/>
      </SubjectAttributeDesignator>
    </Subjects>
  </Target>
</PolicySet>
```
<SubjectMatch />
</Subjects>
<Resources>
<Resource>
<ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
</ResourceMatch>
</Resource>
</Resources>
<Actions>
<Action>
<ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>
</ActionMatch>
</Action>
</Actions>
</Target>
<!— Reference to Trust Service <Policy>s associated with the nurse role -->
<PolicyIdReference>TSP:Rule:nurse:role</PolicyIdReference>
<PolicyIdReference>TSP:Rep:nurse:role</PolicyIdReference>
</PolicySet>

<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    PolicyId="TSP:Rule:nurse:role"
    RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
    <Description>
        Nurse role Rule-based Trust Service rule.
    </Description>
    <Target>
        <Subjects>
            
            </Subjects>
        
    </Target>
</Policy>
<AnySubject/>
</Subjects>
<Resource>
<ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0
:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
string">electronic health record</AttributeValue>
<ResourceAttributeDesignator DataType="http://www.w3.org/2001/
XMLSchema#string">
<AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
</ResourceMatch>
</Resource>
</Resources>
<Actions>
<AnyAction/>
</Actions>
</Target>
<Rule RuleId="nurse:role:ruletrustmetric" Effect="Permit">
<Target>
<Subjects>
<SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0
:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
string">Nurse</AttributeValue>
<SubjectAttributeDesignator DataType="http://www.w3.org/2001/
XMLSchema#string">
<AttributeId="urn:example:subject:role"/>
</SubjectMatch>
</Subjects>
<Resources>
<AnyResource/>
</Resources>
<Actions>
<Action>
<ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0
:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#
string">read</AttributeValue>
<ActionAttributeDesignator DataType="http://www.w3.org/2001/
XMLSchema#string">
<AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>
</ActionMatch>
</Action>
</Rule>
<Action/>
</Actions>
</Target>
<Condition FunctionId="http://example.org:8080/trustpdp/names/function#credential">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
  Green Hospital</AttributeValue>
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
  Nurse</AttributeValue>
</Condition>
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string—one-and-only">
  <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string">
    AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id" />
</Apply>
<Condition/>
</Rule>
<Rule RuleId="DenyAllOthers" Effect="Deny"/>
</Policy>
<Resources>
<Actions>
<AnyAction/>
</Actions>
</Target>

<Rule RuleId="nurse:role:reptrustmetric" Effect="Permit">
<Target>
<Subjects>
<SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Nurse</AttributeValue>
<SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="urn:example:subject:role"/>
</SubjectMatch>
</Subjects>
<Resources>
<AnyResource/>
</Resources>
<Actions>
<Action>
<ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
<ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>
</ActionMatch>
</Action>
</Actions>
</Target>

<Condition FunctionId="http://localhost:8080/trustpdp/names/function#reputation">
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
<SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"/>
</Apply>
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">pagerank</AttributeValue>
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double">0.8</AttributeValue>
</Condition>
B.4 `<PolicySet>` and `<Policy>`s for Social Worker

```xml
<?xml version="1.0" encoding="UTF-8"?>
<PolicySet PolicySetId="Trust:T02:socialworker:role"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Description>
        Trust PDP policy set and references.
    </Description>
    <Target>
        <Subjects>
            <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Social Worker</AttributeValue>
                <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="urn:example:subject:role"/>
            </SubjectMatch>
        </Subjects>
        <Resources>
            <Resource>
                <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">electronic health record</AttributeValue>
                    <ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
                </ResourceMatch>
            </Resource>
        </Resources>
        <Actions>
            <Action>
                <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                </ActionMatch>
            </Action>
        </Actions>
    </Target>
</PolicySet>
```
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>

<ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string">
  AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>
</ActionMatch>
</Action>
</Actions>
</Target>

<!— Reference to Trust Service <Policy> associated with the social worker role —>
</PolicyIdReference>TSP:Combine:socialworker:role</PolicyIdReference>
</PolicySet>

<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  PolicyId="TSP:Combine:nurse:role"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permitoverrides">
  <Description>
    Social Worker role logical combination Trust Service rule.
  </Description>
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
  </Target>
  <Resources>
    <Resource>
      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Electronic Health Record</AttributeValue>
        <ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string">
          AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
      </ResourceMatch>
    </Resource>
  </Resources>
  <Actions>
    <AnyAction/>
  </Actions>
</Rule RuleId="socialworker:role:combinetrustmetric" Effect="Permit"/>
<Target>
<Subjects>
  <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Social Worker</AttributeValue>
    <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="urn:example:subject:role"/>
  </SubjectMatch>
</Subjects>
<Resources>
  <AnyResource/>
</Resources>
<Actions>
  <Action>
    <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Read</AttributeValue>
      <ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>
    </ActionMatch>
  </Action>
</Actions>
</Target>
<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
  <Apply FunctionId="http://example.org:8080/trustpdp/names/function#credential">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Green Medical Service</AttributeValue>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Social Worker</AttributeValue>
  </Apply>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
    <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"/>
  </Apply>
</Condition>
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double">1</AttributeValue>
</Apply>
<Apply FunctionId="http://example.org:8080/trustpdp/names/function#reputation">
B.5 Alternate <Policy>s for Social Worker

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
PolicyId="TSP:Nested:role:role"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
<Description>
Social worker role nested Trust Service rule.
</Description>
<Target>
<Subjects>
<AnySubject/>
</Subjects>
<Resources>
<Resource>
<ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue DataTypes="http://www.w3.org/2001/XMLSchema#string">Electronic Health Record</AttributeValue>
</ResourceMatch>
</Resource>
</Resources>
<Actions>
</Actions>
</Policy>
```
<AnyAction/>
</Actions>
</Target>
<Rule RuleId="socialworker:role:nested:trustmetric" Effect="Permit">
<Target>
<Subjects>
<SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:match:id" FunctionId="http://example.org:8080/trustpdp/names/function#credential">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Social Worker</AttributeValue>
</SubjectMatch>
</Subjects>
<Resources>
<AnyResource/>
</Resources>
<Actions>
<Action>
<ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:match:id" FunctionId="http://example.org:8080/trustpdp/names/function#reputation-of-issuer">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">pagerank</AttributeValue>
</ActionMatch>
</Action>
</Actions>
</Target>
<Condition FunctionId="http://example.org:8080/trustpdp/names/function#reputation-of-issuer">
<Apply FunctionId="http://example.org:8080/trustpdp/names/function#reputation-of-issuer">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Green Medical Service</AttributeValue>
</Apply>
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double">0.8</AttributeValue>
</Condition>
</Rule>
<Condition FunctionId="http://example.org:8080/trustpdp/names/function#reputation-of-issuer">
<Apply FunctionId="http://example.org:8080/trustpdp/names/function#reputation-of-issuer">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Social Worker</AttributeValue>
</Apply>
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Green Medical Service</AttributeValue>
</Condition>
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string—one-and-only">
  <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"/>
</Apply>

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double">1</AttributeValue>
</Condition>
</Rule>
</Policy>

<?xml version="1.0" encoding="UTF-8"?>
<!--
Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
PolicyId="TSP:Rep:socialworker:role"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
<Description>
Social worker role Reputation-based Trust Service rule.
</Description>
<Target>
<Subjects>
<AnySubject/>
</Subjects>
<Resources>
<Resource>
<ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Electronic Health Record</AttributeValue>
  <ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
</ResourceMatch>
</Resource>
</Resources>
<Actions>
<AnyAction/>
</Actions>
</Target>
<Rule RuleId="socialworker:role:retrustmetric" Effect="Permit">
<Target>
<Subjects>
<SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Social Worker</AttributeValue>
  <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="urn:example:subject:role"/>
</SubjectMatch>

<Resources>
  <AnyResource/>
</Resources>

<Actions>
  <Action>
    <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Read</AttributeValue>
      <ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>
    </ActionMatch>
  </Action>
</Actions>

<Target>
  <Condition FunctionId="http://example.org:8080/trustpdp/names/function#reputation">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
      <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"/>
    </Apply>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">pagerank</AttributeValue>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double">0.8</AttributeValue>
  </Condition>
</Target>

<Rule RuleId="DenyAllOthers" Effect="Deny"/>
</Policy>
B.6 Multi-Nested <Policy>s for Social Worker

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       PolicyId="TSP:Nested:socialworker:role"
       RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
                           algorithm:permit-overrides">
  <Description>
    Social worker role nested Trust Service rule.
  </Description>
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <Resource>
      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:
                          function:string-equal">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
          Electronic Health Record</AttributeValue>
        <ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
      </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <Rule RuleId="socialworker:role:nestedtrustmetric" Effect="Permit">
    <Target>
      <Subjects>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:
                         function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            Social Worker</AttributeValue>
          <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="urn:example:subject:role"/>
        </SubjectMatch>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
    </Target>
  </Rule>
</Policy>
```
```xml
"<Resources>

"<Actions>

"<Action>

"<ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0
    :function:string-equal">

"<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Read</AttributeValue>

"<ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string">

"AttributeValue AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>

"</ActionMatch>

"</Action>

"</Actions>

"<Target>

"<Condition FunctionId="http://example.org:8080/trustpdp/names/function#reputation-by-accredited-ratee">

"<Apply FunctionId="http://example.org:8080/trustpdp/names/function#credential-subject-list">

"<Apply FunctionId="http://example.org:8080/trustpdp/names/function#reputation-of-issuer">

"<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">pagerank</AttributeValue>

"<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Green Hospital Healthcare Program</AttributeValue>

"<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double">0.8</AttributeValue>

"</Apply>

"<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">

"Member</AttributeValue>

"<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">

"<SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string">

"AttributeValue AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"/>

"</Apply>

"<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double">

"pagerank</AttributeValue>

"<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#double">0.8</AttributeValue>

"</Apply>

"<Condition>

"</Rule>

"<Rule RuleId="DenyAllOthers" Effect="Deny"/>

"</Policy>
```
Appendix C

Trust PDP Configuration

The following section provides information related to Trust PDP configuration for different scenario of policy specification.

C.1 Trust PDP Configuration for Single and Logical Combination <Policy>s

```xml
<?xml version="1.0" encoding="UTF-8"?>
<config defaultPDP="TrustPDP" defaultAttributeFactory="attr"
    defaultCombiningAlgFactory="comb" defaultFunctionFactory="func">
    <pdp name="TrustPDP">
        <policyFinderModule class="com.sun.xacml.support.finder.StaticPolicyFinderModule">
            <list>
                <string>resources/policy/tpdp_policyset_nurse.xml</string>
                <string>resources/policy/cred_policy_nurse.xml</string>
                <string>resources/policy/rep_policy_nurse.xml</string>
                <string>resources/policy/tpdp_policyset_socialworker.xml</string>
                <string>resources/policy/logical_combination_policy_socialworker.xml</string>
            </list>
        </policyFinderModule>
    </pdp>
</config>
```
C.2 Trust PDP Configuration for Nested <Policy>s

With a similar approach presented in the previous section, the trust PDP configuration for other policy specifications define as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<config defaultTrustService="ExampleTrustServices"
  trusts serviceName="ExampleTrustServices"
  <list>
    <trustService class="eu.tas3.trustpdp.centralityrtm.CentralityRTM"/>
    <trustService class="eu.tas3.trustpdp.polipoctm.PolipoCTM"/>
  </list>
  </trustserviceName>
</config>
```
<policyFinderModule class="com.sun.xacml.support.finder.StaticRefPolicyFinderModule">
    <list>
        ...
        <string>resources/policy/tpdp_policyset_socialworker_2.xml</string>
        <string>resources/policy/nested_policy_socialworker_2.xml</string>
        <string>resources/policy/rep_policy_socialworker_2.xml</string>
    </list>
</policyFinderModule>
...
Appendix D

POLIPO Credential-based Trust Service

The following sections illustrate the verbose XML format of credential release query in a standard and non standard SAML attribute request, the simple POLIPO policy and a sample assertion.

D.1 Credential Release Queries

D.1.1 Ground Query

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soap11:Envelope xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
>
  <soap11:Body>
    <samlp:AttributeQuery xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" ID="1" IssueInstant="2009-11-10T15:10:03.906Z" Version="2.0">
      <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
        Trust PDP</saml:Issuer>
      <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified">Alice</saml:NameID>
    </samlp:AttributeQuery>
    <samlp:Attribute xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" Name="position">
      <samlp:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="xs:string">Nurse</samlp:AttributeValue>
    </samlp:Attribute>
  </soap11:Body>
</soap11:Envelope>
```
D.1.2 Subject-Var Query

```xml
<soap11:Envelope xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/
   ">
  <soap11:Body>
    <samlp:AttributeQuery xmlns:samlp="urn:oasis:names:tc:SAML:2.0
doctype="application/soap+xml; charset=UTF-8"
   :protocol" ID="3" IssueInstant="2009-11-10T15:10:03.921Z"
   Version="2.0">
      <saml:Attribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
         Name="position">
         <saml:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:type="xs:string">Nurse</saml:AttributeValue>
      </saml:Attribute>
    </samlp:AttributeQuery>
  </soap11:Body>
</soap11:Envelope>
```

D.1.3 Attribute-Var Query

```xml
<soap11:Envelope xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/
   ">
  <soap11:Body>
    <samlp:AttributeQuery xmlns:samlp="urn:oasis:names:tc:SAML:2.0
doctype="application/soap+xml; charset=UTF-8"
   :protocol" ID="2" IssueInstant="2009-11-10T15:10:03.921Z"
   Version="2.0">
      <saml:Subject xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
         <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
            format:unspecified">Alice</saml:NameID>
      </saml:Subject>
    </samlp:AttributeQuery>
  </soap11:Body>
</soap11:Envelope>
```
D.2 POLIPO Policy

% POLIPO POLICY

% Directorate−General (DG) POLICY
cred ( 'DG' , 'accreditedUNI' , 'TU/e' ).
cred ( 'DG' , 'accreditedUNI' , 'UT' ).
cred ( 'DG' , 'accreditedUNI' , 'UvA' ).
cred ( 'DG' , 'hospital' , 'YellowHosp' ).
cred ( 'DG' , 'hospital' , 'GreenHosp' ).
cred ( 'DG' , 'healthCareInst' , 'MSP1' ).
cred ( 'DG' , 'healthCareInst' , 'MSP2' ).
cred ( 'DG' , 'diseasePrevProgram' , 'HTP1' ).
cred ( 'DG' , 'diseasePrevProgram' , 'HTP2' ).

% GREEN HOSPITAL POLICY

% MSPs List
cred ( 'GreenHosp' , 'medicalServiceProv' , 'MSP1' ).
cred ( 'GreenHosp' , 'medicalServiceProv' , 'MSP2' ).
% Medical practitioners
cred ( 'GreenHosp' , 'doctor' , 'Julie' ).
cred ( 'GreenHosp' , 'nurse' , 'Alice' ).
% Patients List
cred ( 'GreenHosp' , 'patient' , 'Rob' ).
% Healthcare Programs List
cred ( 'GreenHosp' , 'homeTeleProgram' , 'HTP1' ).
cred ( 'GreenHosp' , 'homeTeleProgram' , 'HTP2' ).
% Healthcare staffs
cred ( 'HTP1' , 'careManager' , 'Jess' ).
cred ( 'HTP1' , 'cardiologist' , 'Lucy' ).
cred ( 'HTP1' , 'nurse' , 'Amber' ).
% Healthcare members
cred ( 'HTP1' , 'member' , 'David' ).
cred ( 'HTP1' , 'member' , 'John' ).
% Healthcare members
cred ( 'HTP2' , 'member' , 'Carl' ).

% TU/e POLICY
cred ( 'TU/e' , 'researcher' , 'Nicola' ).

% UT POLICY
cred ( 'UT' , 'phdDegree' , 'Nicola' ).
cred ('UT', 'phdMedicine', 'Jack').

% UvA POLICY
cred ('UvA', 'phdDegree', X) :- cred ('UvA', 'phdMedicine', X).
cred ('UvA', 'phdMedicine', 'Mark').
cred ('UvA', 'phdDegree', 'Fred').
cred ('UvA', 'firstAidCert', 'Bob').

% MSP1 POLICY
cred ('MSP1', 'firstAidOfficer', X) :- cred ('DG', 'hospital', Y), cred (Y, 'doctor', X).
cred ('MSP1', 'firstAidOfficer', X) :- cred ('DG', 'hospital', Y), cred (Y, 'nurse', X).
cred ('MSP1', 'firstAidOfficer', X) :- cred ('DG', 'accreditedUNI', Y), cred (Y, 'phdMedicine', X).
cred ('MSP1', 'firstAidOfficer', X) :- cred ('DG', 'accreditedUNI', Y), cred (Y, 'firstAidCert', X).

% MSP1 staffs
cred ('MSP1', 'surgeon', 'Phil').
cred ('MSP1', 'physician', 'Andy').
cred ('MSP1', 'socialWorker', 'Bob').
cred ('MSP1', 'nurse', 'Alice').

% MSP2 POLICY
cred ('MSP2', 'pediatrician', 'Betty').
cred ('MSP2', 'generalPractitioner', 'Laura').

D.3 POLIPO Assertion

<soap11:Envelope>
<soap11:Body>
<samlp:Response ID="_fc10830a615df2c3d017106ecd7f3c97" InResponseTo="1" IssueInstant="2009-11-10T15:27:09.281Z" Version="2.0">
<ds:Signature>
<ds:SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<ds:Reference URI="#_fc10830a615df2c3d017106ecd7f3c97"/>
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<saml:AttributeStatement>
  <saml:Attribute FriendlyName="eduPersonEntitlement" Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.7" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
    <saml:AttributeValue xsi:type="xs:string">Alice</saml:AttributeValue>
    <saml:AttributeValue xsi:type="xs:string">Nurse</saml:AttributeValue>
  </saml:Attribute>
</saml:AttributeStatement>
</samlp:Response>
</soap11:Body>
</soap11:Envelope>
Appendix E

Credential Repository

The following sections illustrate the object format and queries (in REST API encoded query) used in Fedora Repository server.

E.1 Fedora Repository FOXML Digital Object Format

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <foxml:objectProperties>
    <foxml:property NAME="info:fedora/fedora-system:def/model#state" VALUE="A"/>
  </foxml:objectProperties>
  <foxml:datastream ID="DC" CONTROL_GROUP="X" STATE="A">
    <foxml:datastreamVersion FORMAT_URI="http://www.openarchives.org/OAI/2.0/oai_dc/" ID="DC1.0" MIMETYPE="text/xml" LABEL="Dublin Core Record for this object">
      <oai_dc:dc xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
        <dc:title>Green Hospital Credentials for Alice</dc:title>
        <dc:creator>Trust PDP</dc:creator>
        <dc:subject>Alice</dc:subject>
        <dc:description>List of SAML assertions</dc:description>
      </oai_dc:dc>
    </foxml:datastreamVersion>
  </foxml:datastream>
</foxml:digitalObject>
```
E.2 Fedora Repository Query

```
http://example.org:8080/fedora/objects?query=pid%7E*Green%20Hospital%20publisher%7EGreen%20Hospital%20subject%7EAlice%20relation%7ENurse%20state%7EA&pid=true&publisher=true&subject=true&relation=true&state=true&mDate=true&resultFormat=xml
```
Appendix F

Reputation Trust Service

The following sections illustrate how to formulate reputation feedbacks and queries to the Centrality RTM trust service.

F.1 Reputation Feedbacks

```xml
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
               xmlns:fb="http://example.ns.org">
  <soap:Header/>
  <soap:Body>
    <fb:giveFeedback>
      <!-- Optional: -->
      <fb:userID>John</fb:userID>
      <!-- Optional: -->
      <fb:serviceEPR>Bob</fb:serviceEPR>
      <!-- Optional: -->
      <fb:serviceType>Emergency Care</fb:serviceType>
      <!-- Optional: -->
      <fb:feedbackFacet>Response Time</fb:feedbackFacet>
      <!-- Optional: -->
      <fb:rating>1.0</fb:rating>
    </fb:giveFeedback>
  </soap:Body>
</soap:Envelope>
```

```xml
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
               xmlns:fb="http://example.ns.org">
  <soap:Header/>
  <soap:Body>
    <fb:giveFeedback>
      <!-- Optional: -->
      <fb:userID>David</fb:userID>
    </fb:giveFeedback>
  </soap:Body>
</soap:Envelope>
```
<!-- Optional: -->
<fb:serviceEPR Alice</fb:serviceEPR>
<!-- Optional: -->
<fb:serviceType>Emergency Care</fb:serviceType>
<!-- Optional: -->
<fb:feedbackFacet>Care Quality</fb:feedbackFacet>
<!-- Optional: -->
<fb:rating>1.0</fb:rating>
</fb:giveFeedback>
</soap:Body>
</soap:Envelope>

F.2 Reputation Service Queries

SELECT score FROM (CENTRALITY score, id, rater, ratee, value, pagerank FROM 'Entity', 'Feedback') AS score JOIN users ON (score.id = users.id) WHERE user_id='Alice'

SELECT user_id, score FROM (CENTRALITY score, id, rater, ratee, value, pagerank FROM 'Entity', (SELECT * FROM feedback WHERE rater IN (SELECT id FROM users WHERE user_id IN (VALUES ('David'), ('John')))) AS Feedback') AS score JOIN users ON (score.id = users.id)
Appendix G

Hardware and Software Prerequisites

The Trust PDP implementation runs on Java (runtime environment 6 or later). Development and testing has been done on Windows XP SP3 though it does not use platform specific code and should work on any platform running the Java VM.

As listed in Table G.1, the Trust PDP requires the following external components:

<table>
<thead>
<tr>
<th>Component</th>
<th>Web Address</th>
<th>Tested Jar Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sun XACML</td>
<td><a href="http://sunxacml.sourceforge.net/">http://sunxacml.sourceforge.net/</a></td>
<td>build from unstable in SVN, 08-10-2009</td>
</tr>
<tr>
<td>Sun XACML-support</td>
<td><a href="http://sunxacml.sourceforge.net/">http://sunxacml.sourceforge.net/</a></td>
<td>build from unstable in SVN, 30-11-2009</td>
</tr>
<tr>
<td>Apache logging</td>
<td><a href="http://logging.apache.org/log4j/1.2/index.html">http://logging.apache.org/log4j/1.2/index.html</a></td>
<td>log4j-1.2.15.jar</td>
</tr>
</tbody>
</table>

In addition, the Centrality RTM trust service interface requires the external component listed in Table G.2. In the other hand, the POLIPO CTM trust service and its interface require the external components and network connectivity shown in Table G.3.
Table G.2: The Centrality RTM trust service required components

<table>
<thead>
<tr>
<th>Component</th>
<th>Web Address</th>
<th>Tested Jar Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>Web Address</td>
<td>Tested Jar Version</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>Apache commons</td>
<td><a href="http://commons.apache.org/">http://commons.apache.org/</a></td>
<td>commons-codec-1.3.jar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>commons-collections-3.1.jar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>commons-httpclient-3.1-beta1.jar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>commons-lang-2.1.jar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>commons-logging-1.0.4.jar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>commons-pool-1.2.jar</td>
</tr>
<tr>
<td>Fedora Commons-DB client</td>
<td><a href="http://sourceforge.net/projects/fedora-commons/">http://sourceforge.net/projects/fedora-commons/</a></td>
<td>fedora-client.jar</td>
</tr>
<tr>
<td>Java cryptography API</td>
<td><a href="http://www.bouncycastle.org/java.html">http://www.bouncycastle.org/java.html</a></td>
<td>bcprov-jdk14-1.38.jar</td>
</tr>
<tr>
<td>Time-Date handling</td>
<td><a href="http://joda-time.sourceforge.net/index.html">http://joda-time.sourceforge.net/index.html</a></td>
<td>joda-time-1.5.1.jar</td>
</tr>
<tr>
<td>Logging</td>
<td><a href="http://www.slf4j.org/">http://www.slf4j.org/</a></td>
<td>logback-classic-0.9.13.jar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>logback-core-0.9.13.jar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>slf4j-api-1.5.6.jar</td>
</tr>
<tr>
<td>SSL utilities</td>
<td><a href="http://juliusdavies.ca/commons-ssl/download.html">http://juliusdavies.ca/commons-ssl/download.html</a></td>
<td>not-yet-commons-ssl-0.3.9.jar</td>
</tr>
<tr>
<td>Open SAML Open WS</td>
<td><a href="https://spaces.internet2.edu/display/">https://spaces.internet2.edu/display/</a></td>
<td>opensaml-2.2.3.jar</td>
</tr>
<tr>
<td></td>
<td>OpenSAML/Home/</td>
<td>opensws-1.3.0.jar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>xmltooling-1.2.0.jar</td>
</tr>
<tr>
<td>Java Servlet</td>
<td><a href="http://tomcat.apache.org/">http://tomcat.apache.org/</a></td>
<td>servlet-api.jar (version 2.4)</td>
</tr>
<tr>
<td>Shibboleth</td>
<td><a href="http://shibboleth.internet2.edu/">http://shibboleth.internet2.edu/</a></td>
<td>shibboleth-common-1.1.2.jar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>shibboleth-identityprovider-2.1.2.jar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>shibboleth-jce-1.0.0.jar</td>
</tr>
<tr>
<td>Spring framework</td>
<td><a href="http://www.springsource.org/">http://www.springsource.org/</a></td>
<td>spring-beans-2.5.5.jar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>spring-context-2.5.5.jar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>spring-context-support-2.5.5.jar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>spring-core-2.5.5.jar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>spring-test.jar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>spring-web-2.5.5.jar</td>
</tr>
<tr>
<td>Apache Velocity</td>
<td><a href="http://velocity.apache.org/">http://velocity.apache.org/</a></td>
<td>velocity-1.5.jar</td>
</tr>
<tr>
<td>XML security</td>
<td><a href="http://santuario.apache.org/">http://santuario.apache.org/</a></td>
<td>xmlsec-1.4.1.jar</td>
</tr>
<tr>
<td>Graph visualization</td>
<td><a href="http://www.jgrapht.org/">http://www.jgrapht.org/</a></td>
<td>jgrapht-jdk1.5-0.7.3.jar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(dependency XML sec)</td>
</tr>
</tbody>
</table>