Offline Expansion of XACML Policies

C. Ardagna, E. Damiani, S. De Capitani di Vimercati, C. Fugazza, P. Samarati

Dipartimento di Tecnologie dell'Informazione
Università di Milano
26013 Crema - Italy
{ardagna,damiani,decapita,fugazza,samarati}@dti.unimi.it

Abstract. In the last few years XML-based access control languages like XACML have been increasingly used for specifying complex policies regulating access to network resources. Today, growing interest in Semantic-Web style metadata for describing resources and users is stimulating research on how to express access control policies based on advanced descriptions rather than on single attributes.
In this paper, we discuss how standard XACML policies can handle ontology-based resource and subject descriptions based on the standard P3P base data schema. We show that XACML conditions can be transparently expanded according to ontology-based models representing semantics. Our expansion technique greatly reduces the need for online reasoning, and decreases the system administrator's effort for producing consistent rules when users' descriptions comprise multiple credential with redundant attributes.

1 Introduction

Semantic-Web style ontologies are aimed at providing a common framework that allows data to be shared and reused by applications across enterprise, and community boundaries. While interest for ontology-based data representation is now growing in many application fields, access control techniques still do not take full advantage of Semantic Web metadata. Even recent proposals for specifying and exchanging access control policies adopt XML-based languages such as the *eXtensible Access Control Markup Language* (XACML) [3] whose flexibility is severely limited by the comparatively low expressive power of formalisms used to describe resources and users requesting them.

In this paper, we present a practical and efficient approach for incorporating into XACML policies ontology-based resource and subject descriptions based on the standard P3P base data schema [6]. In particular, we describe how XACML conditions can be expanded taking into account ontology-based models representing user profiles and resources semantics. Our expansion technique greatly reduces the need for on-line reasoning, relieving the (potentially heavy) computational burden of supporting resource and users' semantics in policy definition and evaluation. As far as user descriptions are concerned, we rely on P3P-based credentials, which are increasingly used to represent subject-related personal information in privacy policies. In our approach, the P3P standard data schema

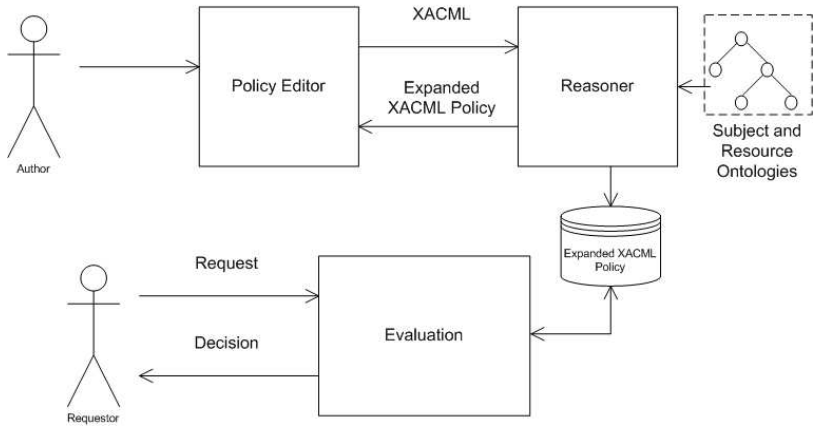

Fig. 1. Scenario

for credential is converted into semantic-Web style metadata that can be easily checked the framework of policy evaluation.[1] Figure 1 illustrates the scenario we consider. The XACML policies are created via the *Policy editor* component. The *Reasoner* takes such XACML policies together with the subject and resource ontologies and computes the expanded XACML policies including semantically equivalent additional conditions. These conditions, specified in disjunction with the original ones, allow for increasing the original policy's expressive power. Our semantically expanded XACML policies can be straightforwardly used as a replacement of the original ones or, more interestingly, can be evaluated side by side with them, flagging cases of inconsistency in the policies' semantics.

The remainder of this paper is organized as follows. Section 3 shows a simple example of XACML policy expansion, deriving possible alternatives obtained by straightforward reasoning over context information. Section 4 illustrates how the reasoning process, by taking into account an explicit representation of resources, derives the actual credential users should provide to be granted access to a given resource. Finally, Section 5 shows how a complex condition is modeled and how the ontology is used to deduce some semantically equivalent alternatives.

2 Representing Heterogeneous Credential Information

We start from a simple RDFS representation of P3P standard data schema. Fig. 3 shows the first level of RDF Schema [8] definitions (henceforth the *language layer*) for our Semantic-Web style representation of the P3P standard

[1] More specifically, the ontology used in this paper relies on the revised P3P base data schema introduced in our previous work [2, 5].

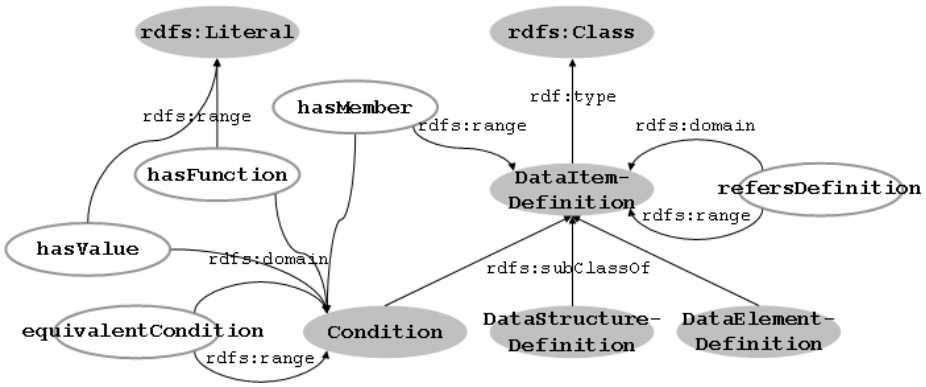

Fig. 2. The language layer defining the building blocks of our RDFS-based representation of P3P data schema.

base data schema [6]: here classes have a grey background color while properties have borders and no background color. Root class `DataItemDefinition` is sub-classed by `DataStructureDefinition` and `DataElementDefinition` in order to model P3P *data structures* and *data elements*. It is also sub-classed by a class `Condition`, used to model policy conditions by means of the associated properties `hasFunction`, `hasMember`, and `hasValue` (representing respectively the evaluation function, the variables and the literal values used in the condition). `Conditions` can feature alternatives by means of property `equivalentCondition`. We denote the inclusion between `DataItems` by means of a property called `refersDefinition`; in RDFS terms, `DataItemDefinition` is both the `rdfs:domain` and `rdfs:range` of property `refersDefinition` used to model the data schema. This property, together with the `rdfs:subClassOf` property, will be used throughout the paper to query the knowledge base for alternatives to data items.

Below the language layer, the *ontology layer* depicted in Fig. 3 comprises data elements and structures represented as classes and linked with each other by means of the `refersDefinition` property. This layer models the semi-lattice structure of the P3P base data schema[5]. At the bottom of the model, the *instance layer* contains the actual negotiable user credential expressed as instances of the classes defined by the ontology layer. Credentials are connected by the `refersInstance` property, modeling the tree structure linking literal values to nodes representing credential instances as a whole. For the sake of simplicity, rather than bending the built-in inference rules associated to ontology languages like OWL we decided to rely on plain RDFS[8] and define from scratch a rule set representing the reasoning patterns required by our application. The examples in the paper make use only of the implications in the ontology layer, hence the instance layer is not further described.

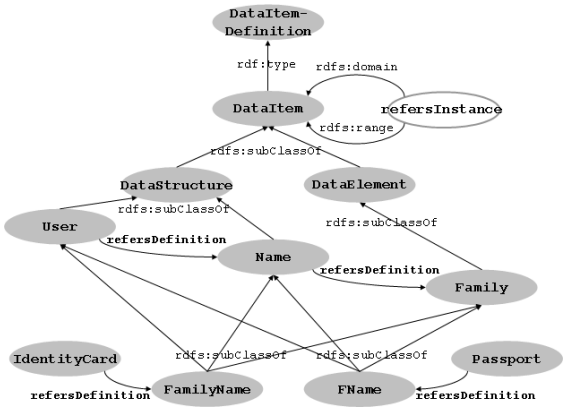

Fig. 3. The ontology layer integrating the P3P base data schema with credential definitions.

3 Referencing Credential Information in Policies

We are now ready to describe some worked out examples of ontology-based policy expansion. In the first example, we show how using an ontology to link independent credential to the P3P base data schema allows for easily specifying alternatives. Fig. 4 shows a XACML condition that references data items of the extended context provided by the ontology via the `urn:...:BaseDataSchema:User:Name:Family` URN, matching it with the literal "Rossi". From the syntax we can gather that the referenced data items:

- belong to the underlying P3P base data schema and can feature alternatives (corresponding to different credentials);
- have type `Family`, which is part of the `Name` context associated to the generic `User` requiring a resource or service.

From the URN provided by the `AttributeId` we can derive a RDQL [9] query extracting such items from P3P data (class `SubjectAttribute` allows for distinguishing subject credentials from resources types):

```
SELECT ?DataElement
WHERE
(<http://.../BaseDataSchema#Family><http://.../BaseDataSchema#refersDefinition>?DataElemen t)
(?DataElement rdfs:type <http://.../BaseDataSchema#SubjectAttribute>)
```

Let us now examine in some more detail the (hopefully rather self-explanatory) RDQL syntax used above. First of all, in the selection clause we collect all ontology nodes referenced by the `Family` node. Since `Family` is a leaf and `refersDefinition` is a reflexive property, the only result is:

```
DataElement
=========================================
<http://.../BaseDataSchema#Family>
```

```
<Rule RuleId="urn:example:ruleid:1" Effect="Permit">
  <Description/>
  <Target/>
  <Condition>
    <Apply FunctionId="urn:oasis:...:function:string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
        Rossi
      </AttributeValue>
      <SubjectAttributeDesignator AttributeId="urn:...:BaseDataSchema:User:Name:Family"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </Apply>
  </Condition>
</Rule>
```

Fig. 4. A sample XACML condition referencing the extended context.

Then, we query the information base for data items semantically equivalent to `BaseDataSchema#Family`; note that the other elements from the URN in the original condition (`User` and `Name`) act as constraints:

```
SELECT ?DataElement
WHERE
(?DataElement rdfs:subClassOf <http://.../BaseDataSchema#User>)
(?DataElement rdfs:subClassOf <http://.../BaseDataSchema#Name>)
(?DataElement rdfs:subClassOf <http://.../BaseDataSchema#Family>)
(?DataElement rdfs:type <http://.../BaseDataSchema#SubjectAttribute>)
```

The RDFS reasoning engine gives us the following results:

```
DataElement
=========================================
<http://.../Passport#FName>
<http://.../IdentityCard#FamilyName>
```

The entities are then mapped to the following URNs:

```
urn:...:IdentityCard:FamilyName
urn:...:Passport:FName
```

We can now expand the original XACML condition to take into account the additional attributes obtained by our expansion procedure, as shown in Fig. 5.

Note that, as long as the two parties agree on the extended context, our expansion procedure does not affect the asymptotic complexity of evaluation, since the expanded condition can still be evaluated by applying XACML standard functions to literal values. Our policy expansion process is hiding the complexity of the semantics-aware information base.

4 Referencing Proprietary Representations of Resources

P3P base data schema is normally used for describing user-related personal information. However, the scope of our technique can be easily enlarged to encompass resources descriptions. The availability of a common data schema allows for rooting an arbitrary representation model for describing resources metadata; actual

```
<Condition>
  <Apply FunctionId="urn:oasis:...:function:or">
    <Apply FunctionId="urn:oasis:...:function:string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
        Rossi
      </AttributeValue>
      <SubjectAttributeDesignator AttributeId="urn:...:BaseDataSchema:User:Name:Family"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </Apply>
    <Apply FunctionId="urn:oasis:...:function:string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
        Rossi
      </AttributeValue>
      <SubjectAttributeDesignator AttributeId="urn:...:IdentityCard:FamilyName"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </Apply>
    <Apply FunctionId="urn:oasis:...:function:string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
        Rossi
      </AttributeValue>
      <SubjectAttributeDesignator AttributeId="urn:...:Passport:FName"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </Apply>
  </Apply>
</Condition>
```

Fig. 5. The condition of Fig. 4 after the expansion.

```
<Condition>
  <Apply FunctionId="urn:example:functions:semantic-match">
    <ResourceAttributeDesignator AttributeId="urn:...:Document:Creator"
      DataType="urn:example:datatypes:structured-type"/>
    <SubjectAttributeDesignator AttributeId="urn:...:BaseDataSchema:User"
      DataType="urn:example:datatypes:structured-type"/>
  </Apply>
</Condition>
```

Fig. 6. XACML condition referencing an arbitrary categorization of resources.

credentials can then be associated with this information. Following the standard XACML approach, our second example introduces a custom matching function called `semantic-match` that will be applied to structured data items, indicated by the custom data type `structured-type`. With reference to Fig. 6, policy conditions will be satisfied if the `Document:Creator` data structure describing a resource matches the User data structure from the base data schema. The task of finding the right credential against which to match the resource's `Creator` can thus be left to the reasoning engine.

In this example, the behavior of our `semantic-match` function is the following:

– First, it retrieves data items identifying a document's creator (for the sake of conciseness, we assume that authors are uniquely defined by their first name, last name, and e-mail). Here, class `ResourceAttribute` allows for distinguishing resources types from subject credential:

```
SELECT ?DataElement
WHERE
(<http://.../Documents#Creator> <http://.../BaseDataSchema#refersDefinition> ?DataElement)
(?DataElement rdfs:type <http://.../BaseDataSchema#ResourceAttribute>)

DataElement
=======================================
<http://.../Documents#FirstName>
<http://.../Documents#FamilyName>
<http://.../Documents#EmailAddress>
```

– Since resources descriptions are not credential, their definitions need not be shared. Thus, for each of the data items retrieved, the corresponding super-classes in the base data schema are identified[2]. The RDQL code shown below selects all data items equivalent to the `FirstName` data element:

```
SELECT ?DataElement
WHERE
(<http://.../Documents#FirstName> rdfs:subClassOf ?DataElement)
(?DataElement rdfs:type <http://.../BaseDataSchema#SubjectAttribute>)

DataElement
=======================================
<http://.../BaseDataSchema#Given>
```

At this point, all data items can be retrieved as in section 3, taking into account the `User` constraint in the `SubjectAttributeDesignator`:

```
SELECT ?DataElement
WHERE
(?DataElement rdfs:subClassOf <http://.../BaseDataSchema#User>)
(?DataElement rdfs:subClassOf <http://.../BaseDataSchema#Given>)
(?DataElement rdfs:type <http://.../BaseDataSchema#SubjectAttribute>)

DataElement
=======================================
<http://.../Passport#GName>
<http://.../IdentityCard#FirstName>
```

After translating all the alternatives into URNs, the condition is expanded according to the translation's results. Once again, we remark that the expanded policy is fully compliant with the XACML standard schema defined in [3]. For the sake of conciseness, however, the (rather verbose) result of the expansion is shown in the Appendix.

5 Expressing advanced semantics-aware conditions

In this Section, our expansion technique is extended to take into account not only the metadata context being referenced by a policy, but also how data items are combined or evaluated in conditions. As we shall see, complex translations schemes can be defined, leading to equivalent conditions in terms of the attributes being compared, the function being applied, and also the right-end value of the

[2] Note that these super-classes are themselves leaves induced in the P3P base data schema by the `refersDescription` property.

```
<Condition>
  <Apply FunctionId="urn:oasis:...:function:string-match">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      Milano
    </AttributeValue>
    <SubjectAttributeDesignator AttributeId="urn:...:IdentityCard:CityOfBirth"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </Apply>
</Condition>
```

Fig. 7. The condition of Section 5

comparison. In other words, our example will take into account not only the context composed of attributes associated to subjects and resources, but also the operational semantics of the policy language describing the rules. Here, metadata represents not only the data items being exchanged, but also the conditions applied to them.

The following triples model the condition of Fig. 7:

```
COND-001 rdf:type dom:Condition
COND-001 dom:hasFunction 'urn:oasis:...:function:string-match'
COND-001 dom:hasValue 'Milano'
COND-001 dom:hasMember IdentityCard:PlaceOfBirth
```

The expansion can derive the following semantic equivalent condition:

```
COND-001 dom:equivalentCondition COND-002
```

The new condition has different values on both the left and right sides, and also uses a different evaluation function:

```
COND-002 rdf:type dom:Condition
COND-002 dom:hasFunction 'urn:oasis:...:function:string-regexp-match'
COND-002 dom:hasValue '/11(\w)F205(\w)'
COND-002 dom:hasMember CodiceFiscale
```

Here the Italian tax code called *codice fiscale* (a 16 digits alphanumeric code uniquely defined by the first name, last name, gender, date and city of birth) is matched against a regular expression requiring the city code 'F205' (indicating people born in Milan) to appear before the control character at the end of the string. The expanded condition is represented in Fig. 8:

Computationally, the equivalence between the two conditions can be checked by direct mapping with tabled values, such as the city codes appearing in the `CodiceFiscale`, or else provided by means of numeric or string conversion functions.

6 Conclusions and future work

In this paper we have shown how we can expand XACML conditions expressing a predicate (e.g. equality) between an attribute and a literal or between two

```
<Condition>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">
    <Apply FunctionId="urn:oasis:...:function:string-match">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
        Milano
      </AttributeValue>
      <SubjectAttributeDesignator AttributeId="urn:example:...:IdentityCard:PlaceOfBirth"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </Apply>
    <Apply FunctionId="urn:oasis:...:function:string-regexp-match">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
        /11(\w)F205(\w)
      </AttributeValue>
      <SubjectAttributeDesignator AttributeId="urn:example:...:CodiceFiscale"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </Apply>
  </Apply>
</Condition>
```

Fig. 8. The condition of Fig. 7 after the expansion.

attributes by means of an ontology based on the standard P3P data schema. One step further toward increasing the policy language expressive power beyond plain XACML would require dealing with more complex logic conditions including variables and quantifiers. For example, a complex condition like "User X can see document Y if exists at least another document Z where =(Y.creation, Z.creation)" cannot be expressed in plain XACML. Also, evaluating this kind of conditions is known to be a difficult problem [1]. For this, we need to define a different XML-syntax, based on a BNF grammar like the one shown below:

```
Q varList booleanExprPred.
booleanExprPred ← pred, booleanExprPred
                ← pred; booleanExpression.
                ← pred.
                ← ¬ pred.
pred ← predName(varList).
varList ← varName, varList.
        ← varName.
Q ← ∃, ∀.
```

an ontological expression through the use of RDQL

Our formal grammar can now be translated into XML as shown below:

```
<resourceList>
  <resource id="001"
    AttributeId="urn:...:Document"
    DataType="urn:...:datatypes:structured-type"/>
</resourceList>
<condition>
  <quantifier id="urn:...:quantifiers:exist-at-least-one">
    <varList>
      <variable id="002"
        AttributeId="urn:...:Document"
        DataType="urn:...:datatypes:structured-type"/>
    </varList>
    <pred function="urn:...:and">
      <pred function="urn:...:predicates:equal">
        <resourceReference ref="001"
```

```
          attributeReference="urn:...:CreationDate"
          DataType="urn:...:datatypes:structured-type"/>
        <variableReference ref="002"
          attributeReference="urn:...:CreationDate"
          DataType="urn:...:datatypes:structured-type"/>
      </pred>
      <pred function="urn:...:predicates:different-individual">
        <resourceReference ref="001"/>
        <variableReference ref="002"/>
      </pred>
    </pred>
  </quantifier>
</condition>
```

Note that with this approach the evaluation mechanism of the new policy language will need to perform ontology-based reasoning as an integral part of the policy evaluation mechanism rather than using it to explicitly expand policies. The evaluation of complex conditions requires a component (currently being developed [**?**]) traslating XML-based logic conditions into RDQL queries to be submitted to an ontology-based reasoner during the evaluation phase. Online reasoning about conditions, of course, will require careful design in order to keep the computational burden of policy evaluation under control; also, it may need to unexpected results, as the effects on policy evaluation of the semantic information stored in the ontology are not available for inspection. For this reason it might be necessary to publish, for transparency, also the inference rules using for the evaluation. We plan to deal with this subject in a future paper.

Acknowledgments

This work was supported in part by the European Union within the PRIME Project in the FP6/IST Programme under contract IST-2002-507591.

References

1. P. A. Bonatti, P. Samarati - *A Uniform Framework for Regulating Service Access and Information Release on the Web* - Journal of Computer Security 10(3): 241-272 (2002)
2. E. Damiani, S. De Capitani di Vimercati, C. Fugazza, and P. Samarati - *Extending Policy Languages to the Semantic Web* -In Proc. of ICWE 2004, Munich, 2004, Lecture Notes in Computer Science 3140.
3. *eXtensible Access Control Markup Language* (XACML) - Organization for the Advancement of Structured Information Standards - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml
4. Jeff Z. Pan, Ian Horrocks - *Metamodeling Architecture of Web Ontology Languages* - In Proc. of the Semantic Web Working Symposium 2001
5. P. Ceravolo, E. Damiani, S. De Capitani di Vimercati, C. Fugazza, and P. Samarati - *Advanced Metadata for Privacy-Aware Representation of Credentials* - In Proc. of the ICDE2005 Workshops, Tokyo, 2005
6. *Platform for Privacy Preferences* (P3P) - W3C Recommendation, 16 April 2002 - http://www.w3.org/TR/P3P/

7. *Privacy and Identity Management for Europe* (PRIME) - European RTD Integrated Project - http://www.prime-project.eu.org/
8. *RDF Vocabulary Description Language* (RDFS) - W3C Recommendation, 10 February 2004 - http://www.w3.org/TR/rdf-schema/
9. *RDQL - A Query Language for RDF* - W3C Member Submission, 9 January 2004 - http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/

A The condition of Fig. 6 after the expansion

```
<Condition>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <ResourceAttributeDesignator AttributeId="urn:...:Document:Creator:FirstName"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
        <SubjectAttributeDesignator AttributeId="urn:...:User:Name:Given"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </Apply><Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <ResourceAttributeDesignator AttributeId="urn:...:Document:Creator:FirstName"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
        <SubjectAttributeDesignator AttributeId="urn:...:IdentityCard:GivenName"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <ResourceAttributeDesignator AttributeId="urn:...:Document:Creator:FirstName"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
        <SubjectAttributeDesignator AttributeId="urn:...:Passport:GName"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </Apply>
    </Apply>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <ResourceAttributeDesignator AttributeId="urn:...:Document:Creator:FamilyName"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
        <SubjectAttributeDesignator AttributeId="urn:...:User:Name:Family"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <ResourceAttributeDesignator AttributeId="urn:...:Document:Creator:FamilyName"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
        <SubjectAttributeDesignator AttributeId="urn:...:IdentityCard:FamilyName"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <ResourceAttributeDesignator AttributeId="urn:...:Document:Creator:FamilyName"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
        <SubjectAttributeDesignator AttributeId="urn:...:Passport:FName"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </Apply>
    </Apply>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-equal">
        <ResourceAttributeDesignator AttributeId="urn:...:Document:Creator:EmailAddress"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
        <SubjectAttributeDesignator AttributeId="urn:...:User:Online:Email"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </Apply>
  </Apply>
</Condition>
```

B RDQL Basics

RDQL queries have the following general form:

```
SELECT ?a
FROM <http://input-model.rdf>
```

```
WHERE (?a, <http://some-predicate>, ?b)
AND ?b < 5
```

Question marks indicate variables, each variable in the `SELECT` clause determines a column in the output. The `FROM` clause allows for selecting a specific file as the input model; this functionality is not used in the paper.

The `WHERE` clause simply defines triples that must be found in the knowledge base for a result to be selected: in the example, elements eligible for the `?a` placeholder must have property `some-predicate` linking to some element `?b`.

Finally, the `AND` clause allows for evaluating literal values according to a set of standard functions: in the example, the element ?b linked to a candidate result ?a must evaluate as < 5. Also this functionality is not used in the paper.