

e4wLGyTU6fHa
jWe3ZPU9pCLT
6769glYHBcOCT
XfohQ5umi / Js
z5zR / o / +KcN3W
iaVQbb9YAKR8V
KImO9IPnFE11U
2He4cZ0Q9bNH
4Rpj1x2IxZEKA
NVED1aIR9tdSg
gfGQy5nQc7Qrh
dX6gM8BOZEvR
IEk3Zmc7qttH7
IqptVm4SeEVL
ruldIze739o01
hsgFPzpS42EB5
Sa03N7bxBur7D
e0quxCSRgKop7
tZ5Q3UQoGUq
4HAwt6h8aChy
D1W9mRPd6RJ
sl0A7+Y2qOCnA
:CqONWvynkcPt
0xV8svh7blePn
MSKZ47RC9I19j
ieV3JDICvSW0D
3jhpgpWWTpqv
dRsVIVTbOUSE
2x2LRB9CKoYNa
UefvYQN4NM2C
hgYex / UpES / n+
r3DBeHxZE957F
ckXlInnnQzp7dY
GsnLw13yvd / s
2NhQf31vE7cDv
oVL30wFBfknV
h ptc Lfki
FFM3 Kw
Jpb KA+
r r1QP7I
SwQmww177

Privacy Rights Markup Language Specification

Version 0.9

June 2001

Copyright © 2001 Zero-Knowledge Systems Inc. All rights reserved.

Zero-Knowledge Confidential.



Table of contents

1. Introduction	4
1.1. Goals and Capabilities	4
1.1.1. Expressiveness	4
1.1.2. Multiple PRML Files	4
1.2. Examples	4
1.3. Terminology	5
1.4. Notation used in the Document	5
2. Document Structure	6
2.1. Sections of a PRML Document	7
2.2. Elements Common to all PRML Objects	7
3. Object Dictionary	9
3.1. Static Objects (Roles, Operations, and Purposes)	9
3.1.1. Roles	10
3.1.2. Operations	10
3.1.3. Purposes	11
3.2. Active Objects (Constraints, Transformations, and Actions)	11
3.2.1. Constraints	12
3.2.2. Transformations	12
3.2.3. Actions	12
4. Data Schema	14
4.1. Data Fields	15
4.2. Data Groups	15
4.3. Data Attributes	15
4.4. Subjects	16
5. Declarations	17
5.1.1. Explicit and Implicit Declarations	19
6. Example Privacy Policies Expressed in PRML	20
6.1. Sample PRML Document A	20
6.2. Sample PRML Document B	23
7. Future work	25
7.1. Commonly Used Objects and Declarations	25

7.2. Transformations	25
7.3. Assigning Declarations on Query Results	25
7.4. Declarations on Level of Generalization	25
7.5. Integrating RDF with PRML.....	26
7.6. Extending the Model for Roles	26
7.7. Reconsidering the <extends> Element.....	26
8. Appendices	27
8.1. Acknowledgements	27
8.2. UML to XML Mapping.....	27
8.3. PRML Static Diagram.....	28
8.4. PRML DTDs.....	29
8.4.1. prml-v1.0.dtd	29
8.4.2. prml-v1.0-rdf.dtd	30
8.4.3. prml-v1.0-sup.dtd	30
8.4.4. prml-v1.0-types.dtd	31
8.4.5. prml-v1.0-data.dtd	32
8.4.6. prml-v1.0-decl.dtd	33
8.4.7. prml-v1.0-roles.dtd.....	33
8.4.8. prml-v1.0-ops.dtd	34
8.4.9. prml-v1.0-purpose.dtd	34
8.4.10. prml-v1.0-constraint.dtd	35
8.4.11. prml-v1.0-trans.dtd	36
8.4.12. prml-v1.0-action.dtd	36
8.4.13. prml-v1.0-attribute.dtd.....	37

1. Introduction

In today's corporate environment, there is often a gap between corporate privacy policies and actual data handling policies. By enabling an organization to formalize its privacy policies with respect to data handling, Privacy Rights Markup Language (PRML) helps solve this problem.

PRML is an XML-based language that allows for the definition of objects- *roles*, *operations*, *data groups*, *subjects*, *purposes*, *constraints*, *actions* and *transformations*- and a mechanism for linking these objects together to form PRML *privacy declarations*. A *Declaration* specifies that a *role* can do an *operation* on a *data group* belonging to a *subject* for a *purpose* if (optionally) certain *constraints* are satisfied. It can also optionally specify that an *action* should take place immediately after this occurs and that the *data element* should be subject to a *transformation* before the *operation* can occur. A privacy policy is a collection of such PRML *declarations*.

Some examples of policies that can be encoded in PRML are:

- A practicing physician can read and update a patient's medical file only if she is the patient's doctor. The doctor can share the file with the lab for analysis. When the file is shared, the age of the patient must be generalized. When the file is shared, the patient must be notified.
- When a PRML document is modified (and thus a privacy policy changed), all current customers must be notified.

1.1. Goals and Capabilities

1.1.1. Expressiveness

PRML's flexible design supports a wide-range of corporate

1.1.2. Multiple PRML Files

A mechanism for inter-PRML file referencing is specified. A PRML file may specify the data handling policies for objects which are defined in multiple PRML files. For example, one PRML file may contain objects associated with the activities of the marketing department while another contains objects associated with the activities of the customer care department. A third file may contain declarations that refer to the objects in the other two files through URI references (see chapter 1.3).

1.2. Examples

Examples of several of the concepts introduced thus far are provided below.

- An example of a *role* is: physician.
- An example of an *operation* is: view.
- An example of a *purpose* is: providing medical care.
- An example of a *data group* is: a medical record containing the patient's name, address and medical condition.
- An example of a *subject* is: a Canadian patient.
- An example of a *constraint* is: the physician is the patient's doctor.

- An example of an *action* is: the patient is notified via email when her record is accessed.
- An example of a *transformation* is: the patient's age is rounded to the nearest decade.
- An example of a *declaration* is: A physician can view medical records belonging to Canadian patients for the purpose of providing medical care if the physician is the patient's doctor. The patient is notified via email when this occurs. The patient's age is rounded to the nearest decade in this circumstance.

1.3. Terminology

PRML Declaration	A statement that establishes a permissible relationship between PRML objects.
PRML Object	A role, operation, data group, subject, data attribute, data field, purpose, constraint, action, or transformation. PRML declarations refer to these objects to specify PRML Declarations.
PRML File	An XML file containing a complete or incomplete set of PRML object definitions and/or declarations.
PRML Document	(Also referred as PRML policy): A PRML File or collection of PRML Files containing a complete set of declarations and objects definitions.
URI Reference	A URI Reference consists of a relative or absolute URI, followed by a hash sign (#) and a fragment identifier: {absolute URI relative URI}#fragment identifier. The URI refers to a PRML file and the fragment identifier refers to a PRML object within that file. Note that to refer to a fragment (PRML object) in the current file, the URI reference may start with the hash sign.
Cardinality	Cardinality defines how many instances of an element type are permitted. The minimum and maximum number of instances are separated by “..”. For example, zero or one instance of an element type is stated as “0..1”. Zero or more instances is stated as “0..*”.

1.4. Notation used in the Document

UML Notation	The objects and attributes of a PRML policy document are described informally in this specification with Unified Modeling Language (UML) static object model diagrams. The UML object diagrams capture the information and relationships, which are then represented in an XML format according to Document Type Definition (DTD) files. The UML class diagrams capture the object types (classes), their attributes, the attribute types, and the relationships between classes.
XML Notation:	<code><Text in Courier></code> is used to represent portions of an XML file.

2. Document Structure

A PRML document is composed of four sections: the RDF Header, the Object Dictionary, the Data Schema and the Declaration Set.

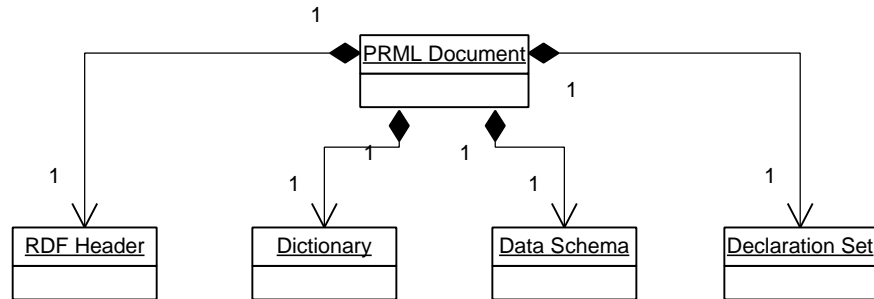


Figure 1 – PRML Document Static Diagram

The following is a sample outline for a PRML document.

Example:

```
<?xml version="1.0" ?>
<!DOCTYPE prml SYSTEM "PRML DTD/prml-v1.0.dtd">
<prml>

<RDF>
  ...
</RDF>

<dictionary>
  <role-set>
    ...
  </role-set>
  <operation-set>
    ...
  </operation-set>
  <purpose-set>
    ...
  </purpose-set>
  <constraint-set>
    ...
  </constraint-set>
  <transformation-set>
    ...
  </transformation-set>
  <action-set>
    ...
  </action-set>
</dictionary>

<data-schema>
  <data-field-set>
    ...
  </data-field-set>
  <data-group-set>
    ...
  </data-group-set>
  <attribute-set>
```

```
    ...
  </attribute-set>
  <subject-set>
    ...
  </subject-set>
</data-schema>

<declaration-set>
  ...
</declaration-set>

</prml>
```

2.1. Sections of a PRML Document

Header

The Header section of the document encapsulates the meta-data about the document, including the author, title, and creation date. The Resource Definition Framework (RDF) (<http://www.w3c.org/RDF>) XML schema is used to represent this information.

Dictionary

The Dictionary section is a collection of definitions of objects that may be used in a policy declaration. This includes Roles, Operations, Purposes, Constraints, Transformations, and Actions. Together with the Subject and Data Group objects defined in the Data Schema, these are the only objects a PRML policy may refer to. For further details, refer to chapter 3.

Data Schema

The Data Schema section of the document is an inventory of data elements, and their useful aggregates which are used to define policies. For further details, refer to chapter 4.

Declaration Set

A privacy policy consists of a collection of declarations. Each declaration describes a relationship between the objects declared in the dictionary and a set of data elements. For further details, refer to chapter 5.

2.2. Elements Common to all PRML Objects

PRML Objects have several attributes, represented as XML elements. While different PRML Objects have different elements, certain elements are common to all Objects. These common elements include <name>, <description>, <implementation>, and unique object identifier (<oid>) elements. Refer to Table 1 for further details.

Table 1 - Elements Common to all PRML Objects

Element	Description	Datatype	Cardinality
oid	A unique object identifier- must be unique among all PRML Objects.	String	1
name	Human readable name.	String	0 .. 1
description	Human readable description.	String	0 .. 1
implementation	An opaque implementation string. Further explanation is found in the descriptions of each object.	String	0 .. 1
properties	A set of name/value pairs. These pairs can be used to extend the language for specific applications.	Property Set (set of name/value pairs)	0 .. 1

The <implementation> element requires further explanation. This string is used by an application to enforce the privacy policy as specified by the PRML Declarations. For Roles, this string can identify a database role. For Operations, this string can specify a database operation, such as read, update, delete, or insert. For Purposes, this string can identify a database application that has this purpose. For Constraints, Actions, Transformations, and Data Attributes, this string can specify their implementation. For Data Fields, this string can specify their actual location, such as a column in a database or a filing cabinet in an office). The format of this string is not specified; it is up to the application to determine the format.

3. Object Dictionary

The Object Dictionary is a collect of definitions of objects that may be referred to by a PRML Declaration. Figure 2 shows the data model of the PRML Object Dictionary.

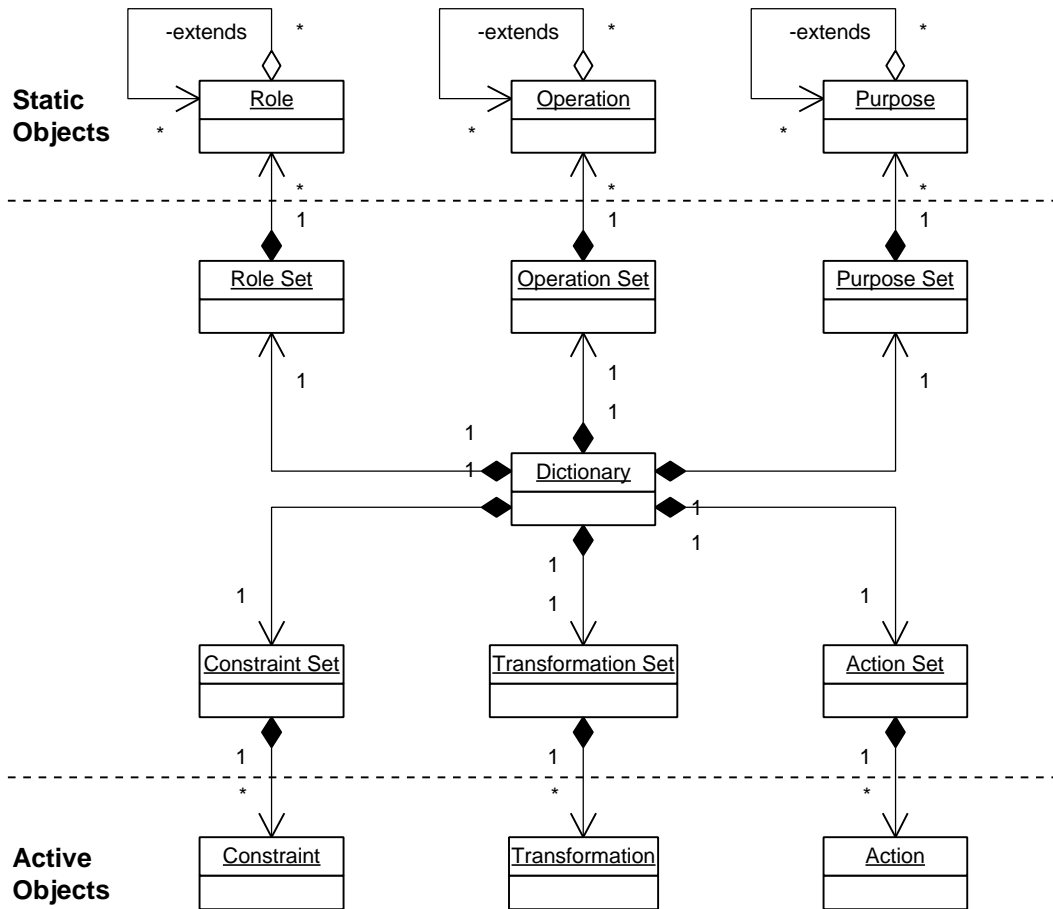


Figure 2 – PRML Dictionary Static Diagram

The Object Dictionary consists of Static Objects- Roles, Operations, and Purpose- and Active Objects- constraints, transformations, and actions. A description of the attributes specific to each object, and corresponding examples for each object type follow.

3.1. Static Objects (Roles, Operations, and Purposes)

PRML Static object types include Roles, Operations, and Purposes. Each of these object types represents a static hierarchy (in the form of a directed acyclic graph) for the corresponding logical entities. In addition to the elements common to all PRML Objects (see Table 1), Table 2 describes an additional element common to all Static Objects.

Table 2 – Additional Attributes Common to all PRML Static Objects

Element	Description	Datatype	Cardinality
extends	URI reference(s) to the base object(s)	URI Reference	0 .. *

3.1.1. Roles

Examples of roles include ‘customer care representative’, ‘customer care supervisor, and ‘database administrator.’ PRML does not include the concept of ‘user’. Instead, all users play a particular role or roles at any given moment. PRML distinguishes users by their roles because privacy policies are created in terms of roles. A policy grants or, by not explicitly granting, prohibits access to a certain role, regardless of which specific user happens to be playing that role.

One role can extend multiple other roles. If a role X extends a role Y, then the existence of a declaration containing a reference to role Y implies that the declaration also applies to role X. For example, if the customer care supervisor role extends the customer care representative role, and a declaration exists granting permission for a customer care representative to perform an operation, a customer care supervisor is implicitly granted this same permission.

Example:

```

<role-set>
  <role>
    <oid>ROLE-001</oid>
    <name>CUSTOMER_CARE_REP</name>
    <description>Customer care representatives are users in the Customer
    Care Department.
    </description>
    <implementation>DB2CUSTOMERDB.ROLES.CUST_CARE_ROLE</implementation>
  </role>
  <role>
    <oid>ROLE-002</oid>
    <name>CUSTOMER_CARE_SUPERVISOR</name>
    <description>Customer care supervisors are extensions of customer care
    representatives.</description>
    <extends>#ROLE-001</extends>
  </role>
</role-set>

```

3.1.2. Operations

Examples of operations include ‘read’, ‘update’, ‘delete’, ‘send email,’ and ‘send surfing profile to ad network.’

Similar to roles, operations may form a hierarchy using the <extends> element. However, the relationship is slightly different. If an operation X extends an operation Y, then the existence of a declaration containing a reference to X implies the existence of a declaration with a reference to Y. (Note that in the case of roles, the existence of a declaration with a reference to Y implies the existence of a declaration with a reference to X.) For example, if sending an email extends reading (an email address from a database) and a declaration exists granting permission for a role to send an email, the same role is implicitly granted permission to read (an email address from a database).

Example:

```

<operation-set>
  <operation>
    <oid>OPERATION-001</oid>
    <name>Read</name>
    <implementation>SELECT</implementation>
  </operation>
  <operation>
    <oid>OPERATION-002</oid>
    <name>Send an email</name>
    <extends>#OPERATION-001</extends>
  </operation>
</operation-set>

```

3.1.3. Purposes

A Purpose object provides the context for performing an operation on some data. A declaration can indicate that an operation is permitted only if the operation is executed for a specific purpose.

Similar to roles and operations, purposes may form a hierarchy using the <extends> element. The direction of the implied relationship is the same as for roles. If a purpose X extends a purpose Y, then the existence of a declaration containing a reference to Y implies the existence of a declaration with a reference to X. For example, if raising awareness about a certain product ABC extends marketing, and a declaration exists granting permission for a role to perform an operation for the generic purpose of marketing, the same role is implicitly granted permission to do the same operation for the purpose of raising awareness about product ABC.

Example:

```

<purpose-set>
  <purpose>
    <oid>PURPOSE-001</oid>
    <name>Marketing</name>
  </purpose>
  <purpose>
    <oid>PURPOSE-002</oid>
    <name>Raise awareness about product ABC</name>
    <extends>#PURPOSE-001</extends>
    <implementation>Siebel_CRM_System</implementation>
  </purpose>
</purpose-set>

```

3.2. Active Objects (Constraints, Transformations, and Actions)

PRML Active Objects are Constraints, Transformations, and Actions. Each of these object types represents run-time procedures for specifying, respectively, when a particular policy statement is in effect, what filters should first be applied to the data, and what obligatory procedure should be initiated immediately after the transaction. Active Objects contain only the elements in Table 1.

3.2.1. Constraints

All constraints referred to by a declaration must be satisfied for the operation to be considered permissible. A constraint in a PRML document contains a label. This label should refer to a function returning a Boolean value. The binding mechanism between a PRML document and the runtime environment is out of the scope of this specification.

Example:

```
<constraint-set>
  <constraint>
    <oid>CONSTRAINT-001</oid>
    <name>Consent has been granted by the user referred to by the data
    element</name>
    <implementation>OracleDB.StoredProcs.IsConsent</implementation>
  </constraint>
</constraint-set>
```

A declaration can contain a reference to CONSTRAINT-001, to indicate that, for example, a role can perform an operation on a data element only if consent has been granted by the user referred to by the data element. The `<implementation>` element 'OracleDB.StoredProcs.IsConsent' refers to a function that returns true if the declaration should be permissible and false otherwise.

The inclusion of Constraints allows PRML to be very expressive. A Constraint can be used to make a declaration dynamic, changing permissiveness based on the data itself. For example, using a Constraint, an Operation as referred to in a Declaration may be permissible for one customer's data, but not another's. One such example is when consent is required to perform an operation.

3.2.2. Transformations

When a declaration refers to a data-access operation, an optional transformation object can be used to determine how the data should be changed prior to delivering it to the requesting user. For example, a declaration can request the age be rounded to the nearest decade before a particular role can read the data. As for constraints, the `<implementation>` element refers to a function that implements the transformation.

Example:

```
<transformation-set>
  <transformation>
    <oid>TRANSFORMATION-001</oid>
    <name>Round Age</name>
    <implementation>OracleDB.StoredProcs.RoundAge</implementation>
  </transformation>
</transformation-set>
```

3.2.3. Actions

Actions are procedures which are executed whenever a declaration is determined to be *in effect*. A specific declaration is *in effect* it makes the current transaction permissible. More than one declaration (but never zero) may be *in effect* for one transaction. If more than one declaration is *in effect*, all actions referred to by each declaration that is *in effect* must be executed. In the case of multiple Actions, the order of execution is not specified. As for Constraints and Transformations, the `<implementation>` element refers to an abstract function that implements the action.

Example:

```
<action-set>
  <action>
    <oid>ACTION-001</oid>
    <name>Notify user</name>
    <implementation>OracleDB.StoredProcs.SendEmail</implementation>
  </action>
</action-set>
```

4. Data Schema

The data representation model used in PRML provides a layer of abstraction between the logical structure of the data and its physical representation in a data store. Figure 3 demonstrates the logical relationship between the data schema elements.

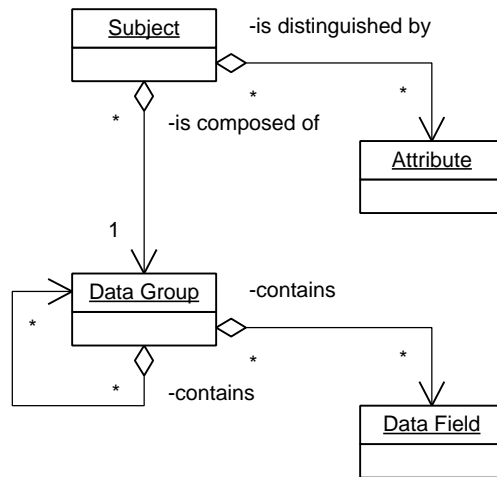


Figure 3 - PRML Data Schema Static Diagram

Four objects exist in the Data Schema: Data Fields, Data Groups, Data Attributes, and Subjects. Data Groups are aggregations of Data Fields and other Data Groups, while Subjects associate a Data Group with a Data Attribute. For example, a Data Group named 'customer contact information' may consist of the Data Fields 'customer name,' 'email address,' and 'country.' A Subject 'Canadian customer' may link the Data Group 'customer contact information' with a Data Attribute 'country is Canada.'

Example:

```
<data-schema>
  <data-field-set>
    <data-field>
      <oid>DATA-FIELD-EMAIL</oid>
      <implementation>DB2.CUSTOMER_TABLE.EMAIL_COLUMN</implementation>
    </data-field>
    <data-field>
      <oid>DATA-FIELD-NAME</oid>
      <implementation>DB2.CUSTOMER_TABLE.NAME_COLUMN</implementation>
    </data-field>
    <data-field>
      <oid>DATA-FIELD-COUNTRY</oid>
      <implementation>DB2.CUSTOMER_TABLE.COUNTRY_COLUMN</implementation>
    </data-field>
  </data-field-set>

  <data-group-set>
    <data-group>
      <oid>DATA-GROUP-CUST-CONTACT-INFO</oid>
      <elements>
        <data-field-idref>#DATA-FIELD-NAME</data-field-idref>
        <data-field-idref>#DATA-FIELD-EMAIL</data-field-idref>
        <data-field-idref>#DATA-FIELD-COUNTRY</data-field-idref>
      </elements>
    </data-group>
  </data-group-set>
```

```

    </data-group>
  </data-group-set>

  <attribute-set>
    <attribute>
      <oid>ATTRIBUTE-CUST-IS-CANADIAN</oid>
      <implementation>STORED_PROCS.CustIsCanadian</implementation>
    </attribute>
  </attribute-set>

  <subject-set>
    <subject>
      <oid>DATA-SUBJECT-CANADIAN-CUSTOMER</oid>
      <data-group-idref>#DATA-GROUP-CUST-CONTACT-INFO</data-group-idref>
      <attribute-specifier-set>
        <attribute-specifier>
          <attribute-idref>#ATTRIBUTE-CUST-IS-CANADIAN
          </attribute-idref>
        </attribute-specifier>
      </attribute-specifier-set>
    </subject>
  </subject-set>
</data-schema>

```

4.1. Data Fields

Data Fields represent atomic elements of data, such as columns in a database or hardcopies of resumes in a filing cabinet. Data Fields consist only of the elements in Table 1.

4.2. Data Groups

Data Groups represent aggregations of data. A Data Group may contain an `<elements>` element that contains URI References to other Data Groups and/or Data Fields.

Table 3 – Additional Attributes of PRML Data Group Objects

Element	Description	Datatype	Cardinality
Elements	A collection of Data Group and Data Field references.	Element Set	1

4.3. Data Attributes

Data Attributes are similar to Constraints in that they are evaluated at run-time. They are used to specify which data in a Data Group belongs to a Subject. For example, consider a company that wishes to express different declarations for the data handling policies of customer information from different countries. One Data Group could identify customer information. Data Attributes could be used to specify which data in the customer information Data Group is Canadian, which is Czech, and so on. In this case, there would be multiple Attributes- ‘Country is Canada’ and ‘Country is Czech Republic’- used by multiple Subjects- ‘Canadian customer,’ and ‘Czech customer.’

4.4. Subjects

Subjects are a modeling abstraction which fosters the association of a Data Group with a particular business view of that data group. This business view is implemented by zero or more Data Attributes. For example, an organization may maintain customer related data within a database. Some policies and procedures may only be applicable to a subset of the customers represented by the entire data set. Subjects can represent such subsets.

Table 4 – Additional Attributes of PRML Subject Objects

Element	Description	Datatype	Cardinality
data-group-idref	The set of data-elements which represent this subject	Data Group	0 .. 1
attribute-specifier-set	The set of data attributes which distinguish this subject	Attribute Set	0 ... 1

Note that the absence of an <attribute-specifier-set> element is equivalent to a single attribute in the set that always returns true. In this case, all data in the referenced Data Group belongs to that Subject. Parameters can be passed to an Attribute using the attribute-specifier-set.

5. Declarations

A PRML Document contains multiple PRML declarations. Each declaration creates a permissible relationship between PRML objects as defined in the Object Dictionary and Data Schema. Figure 4 depicts the relationships between the elements.

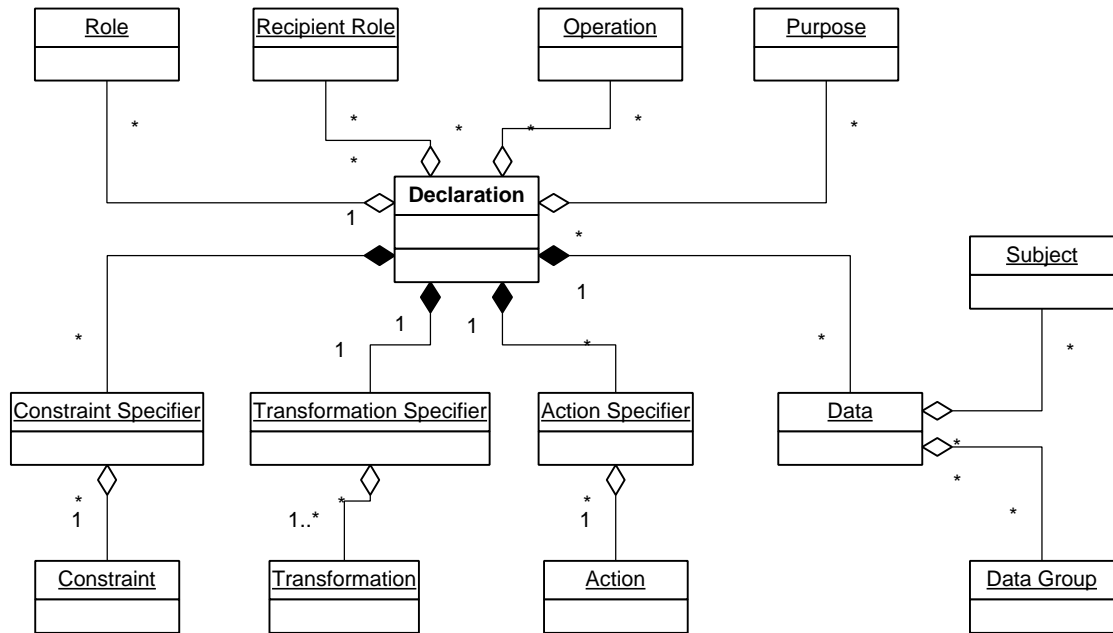


Figure 4 – PRML Declaration Static Diagram

To build a set of all permissible relationships, the following algorithm is used:

- By default, no relationships are permissible. No Roles can do any Operations on any Data Groups for any Purpose.
- If a declaration exists for a relationship, then that relationship is permissible.

A Declaration Object consists of the following elements:

Table 5 – Attributes of PRML Declarations

Element	Description	Default if Cardinality = 0	Datatype	Cardinality
oid	An identifier which must be unique across all declarations	n/a	String	1
name	A human-readable name	n/a	String	0 .. 1
description	A human- readable description	n/a	String	0 .. 1
role-idref	Reference by oid to the role which this declaration is permitting to interact with some data	All Roles	URI Reference	0 .. *
operation-idref	Reference by oid to the operation which the specified roles are to be permitted to perform	All Operations	URI Reference	0 .. *

purpose-idref	Reference by oid to the purpose for which the specified roles can perform the specified operations	All Purposes	URI Reference	0 .. *
data-set	A set of subject/data pairs to which the declarations applies	All Data and All Subjects	URI Reference	0 .. 1
transformation-specifier	A transformation and (optionally) corresponding parameters. The parameters are passed to the function that implements the transformation.	No Transformation on the Data (equivalent to the Identity Transformation)	Class (See diagram below)	0 .. 1
constraint-specifier-set	A set of constraints and (optionally) corresponding parameters. The parameters are passed to the function that implements the constraint.	No Constraints (equivalent to a single Constraint that always returns true)	Class (See diagram below)	0 .. 1
action-specifier-set	A set of actions and (optionally) corresponding parameters. The parameters are passed to the function that implements the action.	No Action is to occur.	Class (See diagram below)	0 .. 1
role-recipient-idref	Reference by oid to the role which receives the data in this declaration.	The Role(s) referred to by role-idref is the recipient.	URI Reference	0 .. *
parent-declaration	Reference by oid to the parent declaration (used if this is an implicit declaration)	This declaration is explicit.	URI Reference	0 .. 1
properties	A set of name value pairs	There are no properties.	Property Set	0 .. 1

The <role-idref>, <operation-idref>, <purpose-idref>, and <role-recipient-idref> elements are URI references (refer to chapter 1.3) to Roles, Operations, and Purposes Objects. If multiple <*-idref> elements exist for a single declaration, the implied relationship is OR. For example, if two <role-idref> elements exist in one declaration, either role can do the specified operation on the specified data for the specified purpose. In contrast, the implied relationship between multiple constraint or action objects in a single declaration is AND. Actions, Constraints, and Transformations can be provided parameters using the <parameter-set> element, which is a sub-element of a <*-specifier> element. Implementations of Transformations, Constraints, and Actions can use the parameters to allow re-use by multiple declarations. The format of such parameters is not defined by this specification.

The <role-recipient-idref> element refers to the recipient role. For example, if a declaration specifies that a marketing agent may send customer information to online advertising companies, <role-recipient-idref> would refer to the 'online advertising company' role, while <role-idref> would refer to the 'marketing' role.

The <data-set> is a set of Data Group/Subject pairs for which a Declaration applies. For example, a Declaration may apply to a Canadian customer's (Subject) email address (Data Group). Note if no subject is specified for a Data Group/Subject pair, the Declaration applies to the entire Data Group. A PRML file is inconsistent and invalid if a Data Group/ Subject pair refers to a Data Group that does not belong to the specified Subject, as defined in the Data Schema.

Example Declaration:

```
<declaration-set>
  <declaration>
    <oid>DECLARATION-001</oid>
    <name>purge-email</name>
    <description>This declaration allows marketing and customer support
      agents to enter Canadian and American customers' email addresses. It
      specifies that the email address must be deleted in 30 days.
    </description>
    <role-idref>#ROLE-MARKETING</role-idref>
    <role-idref>#ROLE-CUSTOMER-SUPPORT</role-idref>
    <operation-idref>#OPERATION-CREATE</operation-idref>
    <purpose-idref>#PURPOSE-DATA-MINIMIZATION</purpose-idref>
    <data-set>
      <data>
        <subject-idref>#SUBJECT-CANADIAN-CUST</subject-idref>
        <data-idref>#DATA-CUST-EMAIL</data-idref>
      </data>
      <data>
        <subject-idref>#SUBJECT-AMERICAN-CUST </subject-idref>
        <data-idref>#DATA-CUST-EMAIL</data-idref>
      </data>
    </data/set>
    <action-specifier-set>
      <action-specifier>
        <operation-idref>#ACTION-PURGE</operation-idref>
        <property-set>
          <property>
            <name>NumberOfDaysUntilDeletion</name>
            <value type="retention-calendar">30</value >
          </property>
        </property-set>
      </action-specifier>
    </action-specifier-set>
  </declaration>
</declaration-set>
```

5.1.1. Explicit and Implicit Declarations

In the Object Dictionary, roles, operations, and purposes each form a static hierarchy using the `<extends>` relationship (see chapter 3.1.1). When a declaration contains a reference to an object X that has an `<extends>` relationship (the object X extends an object Y, for example), implicit declarations exist. The implicit declarations may or may not explicitly appear in a PRML file. However, if they do appear, the `<parent-declaration>` element must appear and reference the original declaration. If the original declaration is subsequently deleted from the PRML file, the implied declaration must also be deleted.

For example, a PRML declaration 'customer-care-agent can send email to customer' uses operation 'send email'. This operation may extend operation 'read'. In order to send email a user will need permission to read the customer's email address. So the explicit declaration above leads to the implicit declaration 'customer-care-agent can read email address.'

6. Example Privacy Policies Expressed in PRML

The following examples are based on hypothetical privacy policies. Note that each privacy policy and corresponding PRML document would be portions of a comprehensive set of policies.

6.1. Sample PRML Document A

The following policy is encoded in the PRML document below:

A customer care representative (role) may create (operation) a customer's (subject) e-mail address (data group) in order to create an account (purpose). This email address must be deleted from the system no later than thirty days after its creation (action). A customer care representative (role) may send email (operation) using a Canadian customer's (subject) email address (data group) in order to provide customer care (purpose) only if there is a pending transaction (constraint). A software module called an operations agent (role) may delete (purpose) a customer's (subject) email address (data group) in order to satisfy the company's minimal retention policy (purpose).

```
<?xml version="1.0" ?>
<!DOCTYPE prml SYSTEM "PRML DTD/prml-v1.0.dtd">
<prml>
<RDF>
<Description />
</RDF>
<dictionary>
  <role-set>
    <role>
      <oid>ROLE-CUSTOMER-CARE-REP</oid>
      <name>Customer care representative</name>
      <description>Customer care representative. This role is assigned to
all users of CRM system</description>
    </role>
    <role>
      <oid>ROLE-OPERATIONS-AGENT</oid>
      <name>OPERATIONS-AGENT</name>
      <description>A software module that assures that all data held in the
customer database respects minimal retention guidelines.</description>
    </role>
  </role-set>

  <operation-set>
    <operation>
      <oid>OPERATION-DELETE</oid>
      <name>Delete</name>
    </operation>
    <operation>
      <oid>OPERATION-CREATE</oid>
      <name>Create</name>
    </operation>
    <operation>
      <oid>OPERATION-SEND-EMAIL</oid>
      <name>Send Email</name>
      <description>email correspondence</description>
    </operation>
  </operation-set>
</dictionary>
</prml>
```

```

<purpose-set>
  <purpose>
    <oid>PURPOSE-MINIMAL-DATA-RETENTION</oid>
    <name>Minimal data retention</name>
  </purpose>
  <purpose>
    <oid>PURPOSE-PROVIDE-CUST-CARE</oid>
    <name>providing customer care</name>
  </purpose>
  <purpose>
    <oid>PURPOSE-CREATE-ACCOUNT</oid>
    <name>Create account</name>
  </purpose>
</purpose-set>

<constraint-set>
  <constraint>
    <oid>CONSTRAINT-PENDING-TRANSACTION</oid>
    <name>There is an pending transaction.</name>
    <implementation>pending-transaction-exists</implementation>
  </constraint>
</constraint-set>

<action-set>
  <action>
    <oid>ACTION-PURGE-EMAIL-ADDRESS-TIMER</oid>
    <name>purge-email-address-timer</name>
    <implementation>purge-email-address-timer</implementation>
  </action>
</action-set>
</dictionary>

<data-schema>
  <data-field-set>
    <data-field>
      <oid>DATA-FIELD-EMAIL</oid>
      <name>email</name>
      <implementation>OracleDB.schema.table.email</implementation>
    </data-field>
  </data-field-set>

  <data-group-set>
    <data-group>
      <oid>DATA-GROUP-EMAIL</oid>
      <name>Customer Email Address</name>
      <elements>
        <data-field-idref>#DATA_FIELD-EMAIL</data-field-idref>
      </elements>
    </data-group>
  </data-group-set>

  <attribute-set>
    <attribute>
      <oid>ATTRIBUTE-CUSTOMER-COUNTRY</oid>
      <description>This attribute takes in a parameter indicating a country
and returns true if this customer is from the specified
country.</description>
      <implementation>StoredProcs.CustomerCountry</implementation>
    </attribute>
  </attribute-set>

  <subject-set>
    <subject>

```

```

<oid>SUBJECT-CANADIAN-CUSTOMER</oid>
<data-group-idref>#DATA-GROUP-EMAIL</data-group-idref>
<attribute-specifier-set>
  <attribute-specifier>
    <attribute-idref>#ATTRIBUTE-CUSTOMER-COUNTRY</attribute-idref>
    <property-set>
      <property>
        <name>Country</name>
        <value>Canada</value>
      </property>
    </property-set>
  </attribute-specifier>
</attribute-specifier-set>
</subject>
<subject>
  <oid>SUBJECT-CUSTOMER</oid>
  <data-group-idref>#DATA-GROUP-EMAIL</data-group-idref>
</subject>
</subject-set>
</data-schema>

<declaration-set>
<declaration>
  <oid>DECL-1</oid>
  <name>purge-email</name>
  <role-idref>#ROLE-CUSTOMER-CARE-REP</role-idref>
  <operation-idref>#OPERATION-CREATE</operation-idref>
  <purpose-idref>#PURPOSE-CREATE-ACCOUNT</purpose-idref>
  <data-set>
    <data>
      <data-group-idref>#DATA-GROUP-EMAIL</data-group-idref>
      <subject-idref>#SUBJECT-CUSTOMER</subject-idref>
    </data>
  </data-set>
  <action-specifier-set>
    <action-specifier>
      <action-idref>#ACTION-PURGE-EMAIL-ADDRESS-TIMER</action-idref>
      <property-set>
        <property>
          <name>purge-email-address</name>
          <value type="retention-calendar">30</value >
        </property>
      </property-set>
    </action-specifier>
  </action-specifier-set>
</declaration>

<declaration>
  <oid>DECL-2</oid>
  <name>email-correspondance</name>
  <role-idref>#ROLE-CUSTOMER-CARE-REP</role-idref>
  <operation-idref>#OPERATION-SEND-EMAIL</operation-idref>
  <purpose-idref>#PURPOSE-PROVIDE-CUST-CARE</purpose-idref>
  <data-set>
    <data>
      <data-group-idref>#DATA-GROUP-EMAIL</data-group-idref>
      <subject-idref>#SUBJECT-CANADIAN-CUSTOMER</subject-idref>
    </data>
  </data-set>
  <constraint-specifier-set>
    <constraint-specifier>
      <constraint-idref>#CONSTRAINT-PENDING-TRANSACTION</constraint-
idref>
    <property-set>

```

```

    </property-set>
  </constraint-specifier>
</constraint-specifier-set>
</declaration>

<declaration>
  <oid>DECL-3</oid>
  <name>email-correspondance</name>
  <role-idref>#ROLE-OPERATIONS-AGENT</role-idref>
  <operation-idref>#OPERATION-DELETE</operation-idref>
  <purpose-idref>#PURPOSE-MINIMAL-DATA-RETENTION</purpose-idref>
  <data-set>
    <data>
      <data-group-idref>#DATA-GROUP-EMAIL</data-group-idref>
      <subject-idref>#SUBJECT-CUSTOMER</subject-idref>
    </data>
  </data-set>
</declaration>
</declaration-set>
</prml>

```

6.2. Sample PRML Document B

The following policy is encoded in the PRML document below:

A marketing representative (role) may send email (operation) in order to do targeted advertising (purpose).

Note that the following implicit declaration also exists in the PRML document:

A marketing representative (role) may read (operation) a customer's (subject) email address (data group) in order to do targeted advertising (purpose).

```

<?xml version="1.0" ?>
<!DOCTYPE prml SYSTEM "PRML DTD/prml-v1.0.dtd">
<prml>
<RDF>
<Description />
</RDF>

<dictionary>
  <role-set>
    <role>
      <oid>ROLE-MARKETING</oid>
      <name>MARKETING-REPRESENTATIVE</name>
      <description>Marketing representative. This role is responsible for
communicating promotion campaigns</description>
    </role>
  </role-set>
  <operation-set>
    <operation>
      <oid>OPERATION-READ</oid>
      <name>Read</name>
    </operation>
    <operation>
      <oid>OPERATION-SEND-EMAIL</oid>
      <name>Send Email</name>
      <extends>#OPERATION-READ</extends>
    </operation>
  </operation-set>
  <purpose-set>
    <purpose>
      <oid>PURPOSE-TARGETING</oid>

```

```

        <name>Targeted advertising</name>
    </purpose-set>
</purpose-set>
<constraint-set />
<transformation-set />
<action-set />
</dictionary>

<data-schema>
  <data-field-set>
    <data-field>
      <oid>DATA-FIELD-EMAIL</oid>
      <name>customer's email address</name>
      <implementation>OracleDB.userschema.userdata.email_address
      </implementation>
    </data-field>
  </data-field-set>
  <data-group-set>
    <data-group>
      <oid>DATA-GROUP-EMAIL</oid>
      <name>Customer Email Address</name>
      <elements>
        <data-field-idref>#DATA_FIELD-EMAIL</data-field-idref>
      </elements>
    </data-group>
  </data-group-set>
  <attribute-set />
  <subject-set>
    <subject>
      <oid>SUBJECT-CUSTOMER</oid>
      <data-group-idref>#DATA_GROUP-EMAIL</data-group-idref>
    </subject>
  </subject-set>
</data-schema>

<declaration-set>
  <declaration>
    <oid>Declaration-1</oid>
    <name>Reading email address to provide targeted advertising</name>
    <role-idref>#ROLE-MARKETING</role-idref>
    <operation-idref>#OPERATION-READ</operation-idref>
    <purpose-idref>#PURPOSE-TARGETING</purpose-idref>
    <data-set>
      <data>
        <data-group-idref>#DATA-GROUP-EMAIL</data-group-idref>
        <subject-idref>#SUBJECT-CUSTOMER</subject-idref>
      </data>
    </data-set>
  </declaration>
</declaration-set>

</prml>

```

7. Future work

7.1. Commonly Used Objects and Declarations

A base set of commonly used objects and declarations will be provided to simplify the creation of PRML documents. For example, a certain number of declarations will be required in any privacy policy that follows the Fair Information Practices.

A language without usage guidelines is difficult to use. The base declarations along with base objects create a framework for development of more rich and customized declarations.

Some examples of base objects are:

- Role: Marketing role
- Data Group: Contact info
- Data Group: Email address
- Subject: Customer
- Purpose: Targeted marketing

An example of a base declaration is:

A privacy architect (role) can modify (operation) a PRML document (data group) for any reason (purpose). Immediately after this occurs, all current customers must be notified (action).

7.2. Transformations

The syntax for transformations requires completion. Should multiple transformation objects apply to a single declaration, where each transformation object transforms a specific data field, or should one transformation object apply to a single declaration and accept parameters for each data-field?

7.3. Assigning Declarations on Query Results

In order to formalize some policy declarations one must refer to the aggregated data structures. It is conceivable to specify a declaration that allows access to the aggregated data structure while prohibiting the access to the very data required to produce aggregated result.

Example: Frequent flier points are collected for every transaction completed by the customer. The total number of points is not stored in the database, but retrieved from the database as a sum of points in all transactions. The role 'customer representative' is allowed to read the total number of points. The same role isn't allowed to read details of individual transactions.

7.4. Declarations on Level of Generalization

Some policy declarations allow a role full access to personal information for the purpose of data mining with the condition that people do not run queries that identify an individual. Such declarations must constrain the minimum number of records returned by a query.

Example: An archiving agent's goal is to populate a data warehouse with historical data on individuals. This role has access to an entire database (with the exception of individual's names), but isn't permitted to run queries that will uniquely identify individuals.

7.5. Integrating RDF with PRML

PRML is currently using a sub-set of the Resource Definition Framework (RDF) (<http://www.w3c.org/RDF>). The use of RDF may be expanded to provide meta-data on the more granular level of PRML objects and declarations.

7.6. Extending the Model for Roles

The data model of PRML roles is not sufficiently expressive to allow formalizing a number of privacy policies. For example, no clean way to represent role delegation exists in the language.

7.7. Reconsidering the <extends> Element

The <extends> element is treated slightly differently for roles and purposes than it is for operations. In the case of roles and purposes, if an object X extends an object Y, then the existence of a declaration referring to Y implies the existence of a declaration referring to X. In the case of operations, if an object X extends an object Y, then the existence of a declaration referring to X implies the existence of a declaration referring to Y. Perhaps a different element should be used to highlight this difference.

8. Appendices

8.1. Acknowledgements

- The PRML Data Schema (see chapter 4.) was derived, in part, from the P3P specification (<http://www.w3.org/TR/P3P/>).
- The XML DTD is generated from the UML drawings according to a set of rules derived from the Customer Profile Exchange (CPExchange) specification and are described in the remainder of this chapter.

8.2. UML to XML Mapping

PRML is an XML application. Currently, the XML representation is defined in XML Document Type Definition (DTD) files.

Firstly, primitive data types are defined to indicate how #PCDATA values should be constrained to match the XML Schema data types. Some of these are the built-in data types defined by the XML Schema Datatypes standard. Others are PRML definitions of new XML Schema generated data types. The intent of the constraints imposed by each data type is either documented in this specification or referenced to in other standards. The XML 1.0 DTD cannot express the data type constraint; instead, the data type is represented with a parameter entity reference. For example:

```
<!-- Primitive Types: they match the XML Scheme Data Types -->
```

```
<!ENTITY % timeInstant "#PCDATA">
```

A class may be represented by two ENTITY definitions in the DTDs. One ENTITY expresses the content of the class (if any), while the other ENTITY expresses programmatic attributes of the class (if any). Subclass entities include superclass entities. Data and relationships, which are the core of the language concepts, are expressed as the content of the relevant class and are represented by element ENTITY definitions. XML attributes, on the other hand, are used to express meta-data about the construct, or instructions to the tools, which must process the construct. Where a class has member values, they are defined following the ENTITY definitions for the contents of that class. For example:

```
<!-- Identifiable Mixin Class -->
<!ENTITY % Identifiable " oid">
  <!-- properties -->
  <!ELEMENT oid (%key;)>
<!-- ExternalReference-Attr
  (describes classes with meta-data telling the tool to import data from
  an external resource -->
<!ENTITY % ExternalReference-Attrs " external-ref CDATA #IMPLIED">
<!-- Role Classes -->
<!ENTITY % Role-Set " role*">
<!ENTITY % Role-Set-Attrs " %ExternalReference-Attrs; , ...">
<!ELEMENT role-set (%Role-Set;)>
<!ATTLIST role-set (%Role-Set_Attrs)>
<!ENTITY % Role " %Identifiable; , ...">
<!ELEMENT role (%Role;)>
```

8.3. PRML Static Diagram

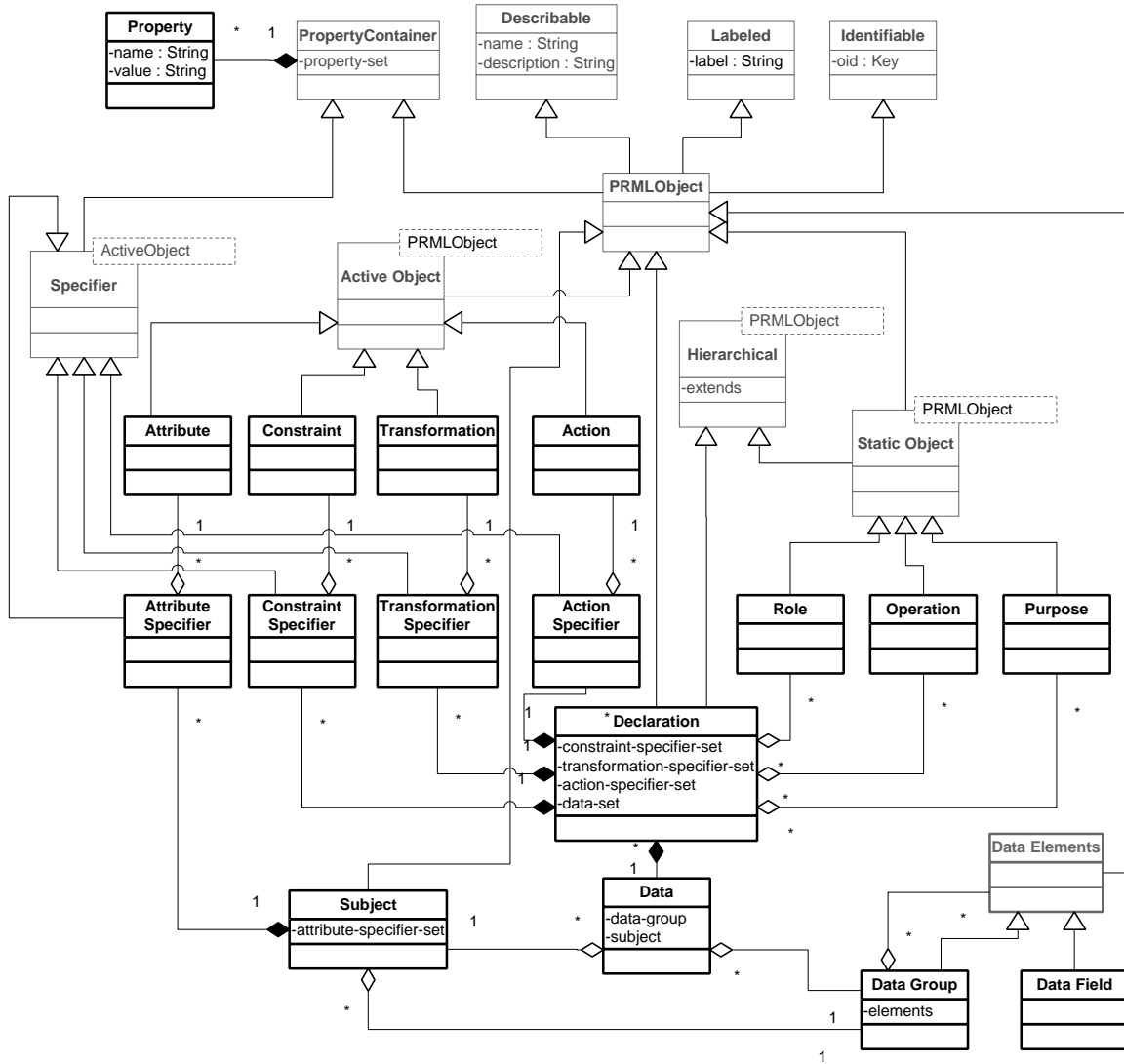


Figure 5 – PRML Object Relationship Diagram

8.4. PRML DTDs

8.4.1. prml-v1.0.dtd

```
<!--
=====

$Id: prml-v1.0.dtd,v 1.2 2001/06/16 18:37:04 roger Exp $

PRML v1.0 DTD

Complete Privacy Rights Markup Language DTD Version 1 with all elements
and entities defined in the standard.

=====
-->

<!-- include all the PRML definitions -->

<!ENTITY % type-defs SYSTEM " prml-v1.0-types.dtd">
%type-defs;

<!ENTITY % support-defs SYSTEM " prml-v1.0-sup.dtd">
%support-defs;

<!ENTITY % role-defs SYSTEM " prml-v1.0-roles.dtd">
%role-defs;

<!ENTITY % operation-defs SYSTEM " prml-v1.0-ops.dtd">
%operation-defs;

<!ENTITY % purpose-defs SYSTEM " prml-v1.0-purpose.dtd">
%purpose-defs;

<!ENTITY % rdf-defs SYSTEM " prml-v1.0-rdf.dtd">
%rdf-defs;

<!ENTITY % data-defs SYSTEM " prml-v1.0-data.dtd">
%data-defs;

<!ENTITY % constraint-defs SYSTEM " prml-v1.0-constraint.dtd">
%constraint-defs;

<!ENTITY % transformation-defs SYSTEM " prml-v1.0-trans.dtd">
%transformation-defs;

<!ENTITY % action-defs SYSTEM " prml-v1.0-action.dtd">
%action-defs;

<!ENTITY % declaration-defs SYSTEM " prml-v1.0-decl.dtd">
%declaration-defs;

<!-- define root document element -->

<!ELEMENT prml (RDF, prml-import*, dictionary, data-schema, declaration-
set)>

<!-- define dictionary element -->
<!ELEMENT dictionary (role-set, operation-set, purpose-set, constraint-
set?, transformation-set?, action-set?)>
```

```

<!ENTITY % PrmlImportReference " external-ref-location CDATA #REQUIRED">

<!ELEMENT prml-import EMPTY>
  <!ATTLIST prml-import %PrmlImportReference; >

```

8.4.2. prml-v1.0-rdf.dtd

```

<!--
=====

$Id: prml-v1.0-rdf.dtd,v 1.2 2001/06/16 18:37:04 roger Exp $

This file defines the header of PRML document. RDF syntax is used to
include document metadata. See http://www.w3.org/RDF/ for more
information.

=====
-->

<!ELEMENT RDF (Description)>
<!ELEMENT Description (title?, creator?, date?)>
<!ELEMENT title (%string;)>
<!ELEMENT creator (%string;)>
<!ELEMENT date (%string;)>

```

8.4.3. prml-v1.0-sup.dtd

```

<!--
=====

$Id: prml-v1.0-sup.dtd,v 1.2 2001/06/16 18:37:04 roger Exp $

PRML supplemental definitions and declarations, derived from
CPExchange.

To incorporate the elements in this file into a higher level document
structure, the root, (or some other) element of the new document should
include the elements as desired. This file imposes no structure on the
use of these elements.

=====
-->

<!ENTITY % Identifiable " oid">
  <!-- properties -->
  <!ELEMENT oid (%key;)>

<!ENTITY % Named " name?">
  <!-- properties -->
  <!ELEMENT name (%string;)>

<!ENTITY % Implementable " implementation?">
  <!-- properties -->
  <!ELEMENT implementation (%string;)>

<!ENTITY % Describable " description?">
  <!-- properties -->
  <!ELEMENT description (%string;)>

<!ENTITY % Hierarchical " extends?">
  <!-- properties -->
  <!ELEMENT extends (%key;)>

<!ENTITY % Versionable " version?" >

```

```

    <!-- properties -->
        <!ELEMENT version (%string;)>

<!ENTITY % ExternalReference-Attrs " external-ref-location CDATA #IMPLIED">

<!ENTITY % TypeAttribute " type CDATA #IMPLIED">

<!ENTITY % HasValue " value">
    <!ELEMENT value (%string;) >
    <!ATTLIST value %TypeAttribute; >

<!ENTITY % PropertyContainer " property-set?">
    <!ELEMENT property-set (property*)>
    <!ELEMENT property (%Named;, %HasValue;)>

<!ENTITY % PRMLObject " %Identifiable;,
                        %Named;,
                        %Describable;,
                        %Implementable;,
                        %PropertyContainer;" >

<!ENTITY % ActiveObject " %PRMLObject;" >

<!ENTITY % StaticObject " %PRMLObject;,%Hierarchical;" >

<!ENTITY % Specifier " %PropertyContainer;" >

```

8.4.4. prml-v1.0-types.dtd

```

<!--
=====

$Id: prml-v1.0-types.dtd,v 1.1 2001/06/14 18:43:16 corey Exp $

File: prml-v1.0-types.dtd
Privacy Rights Markup Language DTD Version 1 Datatypes include file.
To use declare this file as an ENTITY and include it in the dtd
that uses these definitions.

Borrowed from CPExchange

=====
-->

<!-- primitive XML Schema data types -->

<!ENTITY % real "#PCDATA">
<!ENTITY % boolean "#PCDATA">
<!ENTITY % language "#PCDATA">
<!-- RFC 1766 format -->
<!ENTITY % string "#PCDATA">
<!-- ISO 10646 unicode -->
<!ENTITY % uriReference "#PCDATA">
<!-- Uniform Resource Locator, IETF RFC 1738 -->

<!-- generated builtin XML Schema data types -->

<!ENTITY % decimal " %real;">
<!ENTITY % integer " %decimal;">
<!ENTITY % non-negative-integer " %integer;">
<!ENTITY % non-positive-integer " %integer;">

```

```

<!ENTITY % positive-integer " %non-negative-integer;">
<!ENTITY % negative-integer " %non-positive-integer;">
<!ENTITY % mime " %string;">

<!-- user defined types representing enumerated element values -->

<!ENTITY % Enum "#PCDATA">
<!ENTITY % OpenEnum " %Enum;">
<!ENTITY % ClosedEnum " %Enum;">
<!ENTITY % CountryCodeEnum " %ClosedEnum;">

<!-- user defined simple data types -->
<!ENTITY % key " %string;">
<!-- string value is any supplier-specific object identifier -->

<!-- property extensibility type -->
<!ENTITY % Property " %string;">
<!ENTITY % PropertyAttrs " name CDATA #REQUIRED enumType CDATA #IMPLIED">

```

8.4.5. prml-v1.0-data.dtd

```

<!--
=====
$Id: prml-v1.0-data.dtd,v 1.2 2001/06/16 18:37:04 roger Exp $

This file describes the high level constructs that make up a PRML data
definition. It depends on the DTD files which define the basic and
supplemental types. To use this file, declare it as an ENTITY and
include it.

To incorporate the elements in this file into a higher level document
structure, the root, (or some other) element of the new document should
include the elements as desired. This file imposes no structure on the
use of these elements.

=====
-->

<!-- Bring in Attribute definitions -->

<!ENTITY % attribute-defs SYSTEM " prml-v1.0-attribute.dtd">
%attribute-defs;

<!--Define the other stuff that goes in the data schema -->

<!ENTITY % DataElement " %PRMLObject;">

<!ENTITY % DataField " %DataElement;">

<!ENTITY % DataGroup " %DataElement;, elements">
<!ELEMENT elements ((data-field-idref|data-group-idref)*)>

<!ENTITY % Subject " %PRMLObject;,
data-group-idref,
attribute-specifier-set?">

<!ELEMENT data-field (%DataField;)>
<!ELEMENT data-group (%DataGroup;)>
<!ELEMENT subject (%Subject;)>

<!ELEMENT data-field-set (data-field*)>

```

```

<!ELEMENT data-group-set (data-group*)>
<!ELEMENT subject-set (subject*)>

<!ELEMENT data-schema (data-field-set,
                        data-group-set,
                        attribute-set,
                        subject-set) >

<!-- To refer data elements by oid use the following element types -->

<!ELEMENT data-group-idref (%key;) >
<!ELEMENT data-field-idref (%key;) >
<!ELEMENT subject-idref (%key;) >

<!-- Used by declarations -->

<!ELEMENT data-set (data*)>
<!ELEMENT data (data-group-idref, subject-idref?)>

```

8.4.6. prml-v1.0-decl.dtd

```

<!--
=====

$Id: prml-v1.0-decl.dtd,v 1.2 2001/06/16 18:37:04 roger Exp $

This file describes the high level constructs that make
up a PRML rule declaration. It depends on the DTD files
which define the basic and supplemental types.

To use this file, declare it as an ENTITY and include it.

To incorporate the elements in this file into a higher level
document structure, the root, (or some other) element of the
new document should include the elements as desired. This
file imposes no structure on the use of these elements.

=====
-->

<!ENTITY % DeclarationSet " declaration*">
<!ENTITY % Declaration " %PRMLObject;
                        %Hierarchical;
                        role-idref*,
                        operation-idref*,
                        purpose-idref*,
                        recipient-idref*,
                        data-set?,
                        constraint-specifier-set?,
                        transformation-specifier-set?,
                        action-specifier-set?">

<!ELEMENT declaration-set (%DeclarationSet;)>

<!ELEMENT declaration (%Declaration;)>

```

8.4.7. prml-v1.0-roles.dtd

```

<!--
=====

$Id: prml-v1.0-roles.dtd,v 1.2 2001/06/16 18:37:04 roger Exp $

```

This file describes the high level constructs that make up a PRML role definition. It depends on the DTD files which define the basic and supplemental types. To use this file, declare it as an ENTITY and include it.

To incorporate the elements in this file into a higher level document structure, the root, (or some other) element of the new document should include the elements as desired. This file imposes no structure on the use of these elements.

```
=====  
-->  
<!ENTITY % RoleSet " role*">  
<!ENTITY % Role " %StaticObject;">  
  
<!ELEMENT role-set (%RoleSet;)>  
  
<!ELEMENT role (%Role;)>  
  
<!-- To refer to roles by oid use the following element types -->  
  
<!ELEMENT role-idref (%key;)>  
<!ELEMENT recipient-idref (%key;)>
```

8.4.8. prml-v1.0-ops.dtd

```
<!--  
=====  
  
$Id: prml-v1.0-ops.dtd,v 1.2 2001/06/16 18:37:04 roger Exp $  
  
This file describes the high level constructs that make  
up a PRML operation definition. It depends on the DTD files  
which define the basic and supplemental types.  
  
To use this file, declare it as an ENTITY and include it.  
  
To incorporate the elements in this file into a higher level  
document structure, the root, (or some other) element of the  
new document should include the elements as desired. This  
file imposes no structure on the use of these elements.  
  
=====  
-->  
  
<!ENTITY % OperationSet " operation*">  
<!ENTITY % Operation " %StaticObject;">  
  
<!ELEMENT operation-set (%OperationSet; ) >  
  
<!ELEMENT operation (%Operation;)>  
  
<!-- To refer to an operation by its oid use the following element type -->  
<!ELEMENT operation-idref (%key;) >
```

8.4.9. prml-v1.0-purpose.dtd

```
<!--  
=====  
  
$Id: prml-v1.0-purpose.dtd,v 1.2 2001/06/16 18:37:04 roger Exp $  
  
This file describes the high level constructs that make
```

up a PRML purpose definition. It depends on the DTD files which define the basic and supplemental types.

To use this file, declare it as an ENTITY and include it.

To incorporate the elements in this file into a higher level document structure, the root, (or some other) element of the new document should include the elements as desired. This file imposes no structure on the use of these elements.

```
=====
-->

<!ENTITY % PurposeSet " purpose*">
<!ENTITY % Purpose " %StaticObject;">

<!ELEMENT purpose-set (%PurposeSet;) >

<!ELEMENT purpose (%Purpose;)>

<!-- To refer to an operation by its oid use the following element type -->
<!ELEMENT purpose-idref (%key;) >
```

8.4.10. prml-v1.0-constraint.dtd

```
<!--
=====

$Id: prml-v1.0-constraint.dtd,v 1.2 2001/06/16 18:37:04 roger Exp $

This file describes the high level constructs that make up a PRML
constraint definition. It depends on the DTD files which define the
basic and supplemental types. To use this file, declare it as an
ENTITY
and include it.

To incorporate the elements in this file into a higher level document
structure, the root, (or some other) element of the new document should
include the elements as desired. This file imposes no structure on the
use of these elements.

=====
-->

<!ENTITY % ConstraintSet " constraint*">
<!ENTITY % Constraint " %ActiveObject;">

<!ELEMENT constraint-set (%ConstraintSet;) >

<!ELEMENT constraint (%Constraint;)>

<!-- Use HasConstraints when an element includes references to constraints
-->
<!ELEMENT constraint-specifier-set (constraint-specifier*)>

<!ELEMENT constraint-specifier (constraint-idref, %Specifier;)>

<!-- To refer to an operation by its oid use the following element type -->
<!ELEMENT constraint-idref (%key;)>
```

8.4.11. prml-v1.0-trans.dtd

```
<!--
=====

$Id: prml-v1.0-trans.dtd,v 1.2 2001/06/16 18:37:04 roger Exp $

This file describes the high level constructs that make up a PRML
transformation definition. It depends on the DTD files which define
the basic and supplemental types. To use this file, declare it as an
ENTITY and include it.

To incorporate the elements in this file into a higher level document
structure, the root, (or some other) element of the new document should
include the elements as desired. This file imposes no structure on the
use of these elements.

=====
-->

<!ELEMENT transformation-set (transformation*) >

<!ELEMENT transformation ( %Identifiable;,
                           %Named;,
                           %Implementable;,
                           %Describable;)>

<!ELEMENT transformation-specifier-set (transformation-specifier*)>
<!ELEMENT      transformation-specifier      (transformation-idref,
%PropertyContainer;)>
<!ELEMENT transformation-idref (%key;)>
```

8.4.12. prml-v1.0-action.dtd

```
<!--
=====

$Id: prml-v1.0-action.dtd,v 1.2 2001/06/16 18:37:04 roger Exp $

This file describes the high level constructs that make up a PRML
action
definition. It depends on the DTD files which define the basic and
supplemental types. To use this file, declare it as an ENTITY and
include it.

To incorporate the elements in this file into a higher level document
structure, the root, (or some other) element of the new document should
include the elements as desired. This file imposes no structure on the
use of these elements.

=====
-->

<!ENTITY % ActionSet " action*">
<!ENTITY % Action " %ActiveObject;">

<!ELEMENT action-set (%ActionSet;) >
```

```

<!ELEMENT action (%Action;)>

<!ELEMENT action-specifier-set (action-specifier*)>

<!ELEMENT action-specifier (action-idref, %Specifier;)>

<!-- To refer to an action by its oid use the following element type -->
<!ELEMENT action-idref (%key;)>

```

8.4.13. prml-v1.0-attribute.dtd

```

<!--
=====

$Id: prml-v1.0-attribute.dtd,v 1.1 2001/06/16 18:37:04 roger Exp $

This file describes the high level constructs that make up a PRML
attribute definition. It depends on the DTD files which define the
basic and supplemental types. To use this file, declare it as
an ENTITY and include it.

To incorporate the elements in this file into a higher level document
structure, the root, (or some other) element of the new document should
include the elements as desired. This file imposes no structure on the
use of these elements.

=====
-->

<!ENTITY % AttributeSet " attribute*">
<!ENTITY % Attribute " %ActiveObject;">

<!ELEMENT attribute-set (%AttributeSet;) >

<!ELEMENT attribute (%Attribute;)>

<!ELEMENT attribute-specifier-set (attribute-specifier*)>

<!ELEMENT attribute-specifier (attribute-idref, %Specifier;)>

<!-- To refer to an attribute by its oid use the following element type -->
<!ELEMENT attribute-idref (%key;)>

```