Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2351696 A1 2002/12/26

(21) 2 351 696

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(22) Date de dépôt/Filing Date: 2001/06/26

(41) Mise à la disp. pub./Open to Public Insp.: 2002/12/26

(51) Cl.Int.^{7}/Int.Cl.^{7} G06F 17/00

(71) Demandeur/Applicant:
ZERO-KNOWLEDGE SYSTEMS INC., CA

(72) Inventeurs/Inventors:
SMIRNOV, ALEXIS, CA;
BOUCHER, PHILIPPE, CA;
VELAN, COREY, CA;
MCFARLANE, ROGER, CA

(74) Agent: FASKEN MARTINEAU DUMOULIN LLP

(54) Titre : SYSTEME POUR LA MISE EN APPLICATION DE POLITIQUES EN MATIERE DE CONFIDENTIALITE ECRITES A L'AIDE D'UN LANGAGE DE BALISAGE

(54) Title: SYSTEM FOR IMPLEMENTING PRIVACY POLICIES WRITTEN USING A MARKUP LANGUAGE

System for Implementing Privacy Policies written using a markup language

BACKGROUND OF THE INVENTION

In today's corporate environment, it is often difficult ot coordinate a corporate privacy policies with access control policies. A set of costly processes is necessary to assure that the two policies are consistently coordinated.

SUMMARY OF THE INVENTION

Privacy Rights Markup Language (PRML) is the foundation for the solution to this problem.

PRML is an XML-based language that allows for the definition of objects- *roles*, *operations*, *data elements*, *purposes*, *constraints*, *actions* and *transformations*- and a mechanism for linking these objects together to form PRML privacy *declarations*. *Declarations* specify that a *role* can do an *operation* on a *data element* for a *purpose* if (optionally) certain *constraints* are satisfied. It can also optionally specify that an *action* should take place when 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*.

The PRML standard offers a comprehensive set of constructs in order to represent privacy policies in full compliance with the Fair Information Practices (http://www.epic.org/privacy/consumer/code_fair_info.html).

PRML enables an organization to formalize its privacy policies and corresponding operational procedures. The language also enables the specification of policies around altering privacy policies.

Example: A PRML document may specify the following declaration: When a PRML document is modified (and thus a privacy policy changed), all individuals that have consented to the current policy must be notified.

PRML is flexible to unambiguously define every possible privacy policy a corporation might wish to enforce.

PRML can be used in an auditing capacity. In this capacity, instead of interpreting a declaration as specifying that a role *can do* an operation on a data element for a purpose, it can be interpreted to specify that a role *did* an operation on a data element for a purpose.

A mechanism for document extension is specified. A single PRML file may not contain a full set of declarations or objects.

Example: One PRML file may contain the objects and declarations associated with the activities of the marketing department while another contains the objects and declarations associated with the activities of the customer care department.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Prior to an in depth specification of PRML objects and declarations, some examples may aid the reader .

- 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 element* is: a patient's medical record containing the patient's name, address and medical condition.

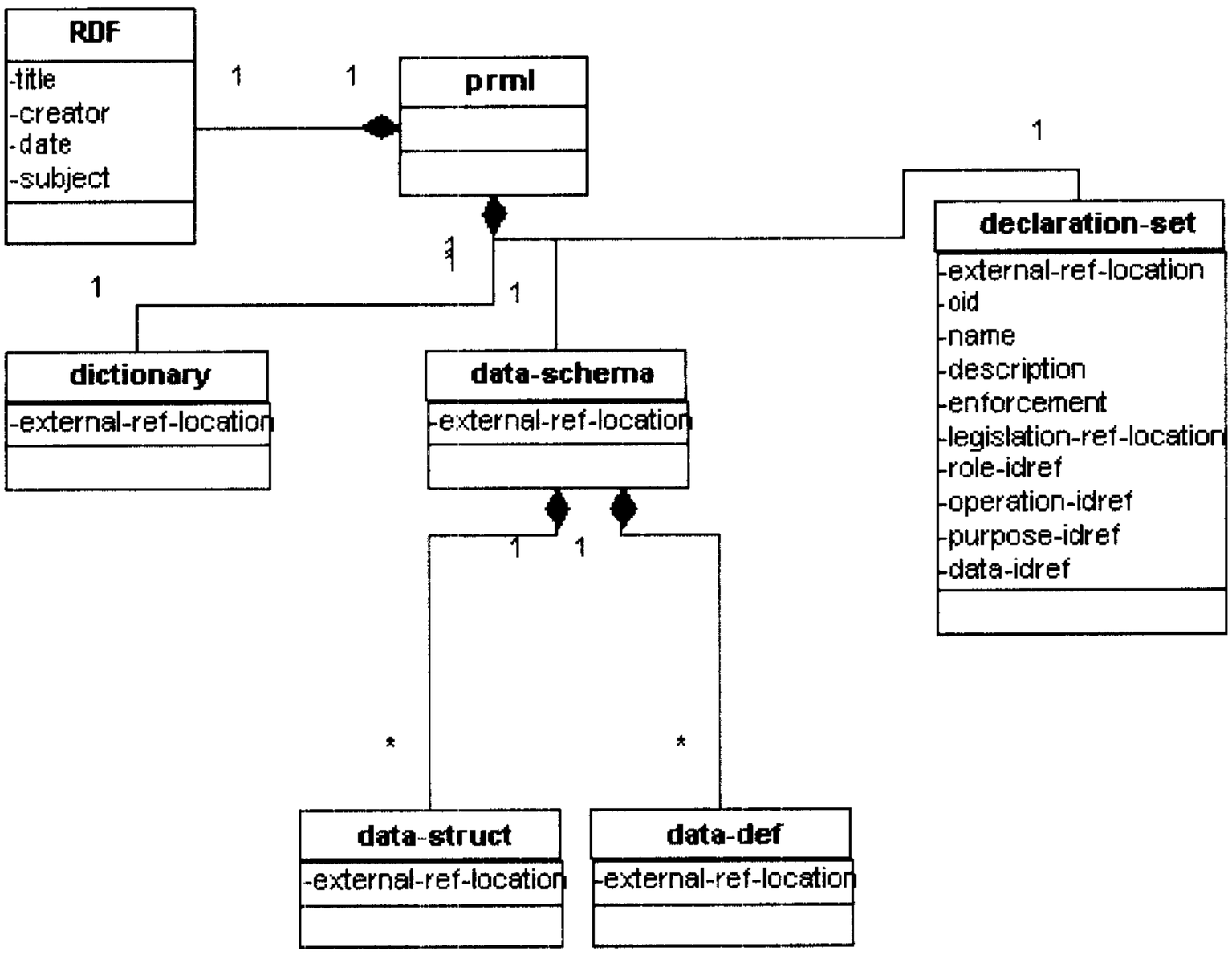
- An example of a *constraint* is: the physician is the patient's doctor.
- An example of an *action* is: the patient must be notified via email when her record is accessed.
- An example of a *transformation* is: the HIV test in the medical record should be replaced with "HIV test result not available".
- An example of a *declaration* is: A physician can view a patient's medical records for the purpose of providing medical care if the physician is the patient's doctor. The patient should be notified via email when this occurs. The HIV test result in the medical condition should be replaced with "HIV test result not available" in this circumstance.

Terminology and Conventions

- **PRML Declaration**: A statement that establishes a permissible relationship between PRML objects.
- **PRML Object**: A role, operation, data element, purpose, constraint, action, or transformation. PRML declarations refer to these objects and only these objects.
- **PRML File**: An XML file containing a full or partial 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 comprehensive set of declarations and objects definitions.
- **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:** `<Text in courier>` is used to represent portions of an XML file.

PRML Document Structure

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



RDF Header

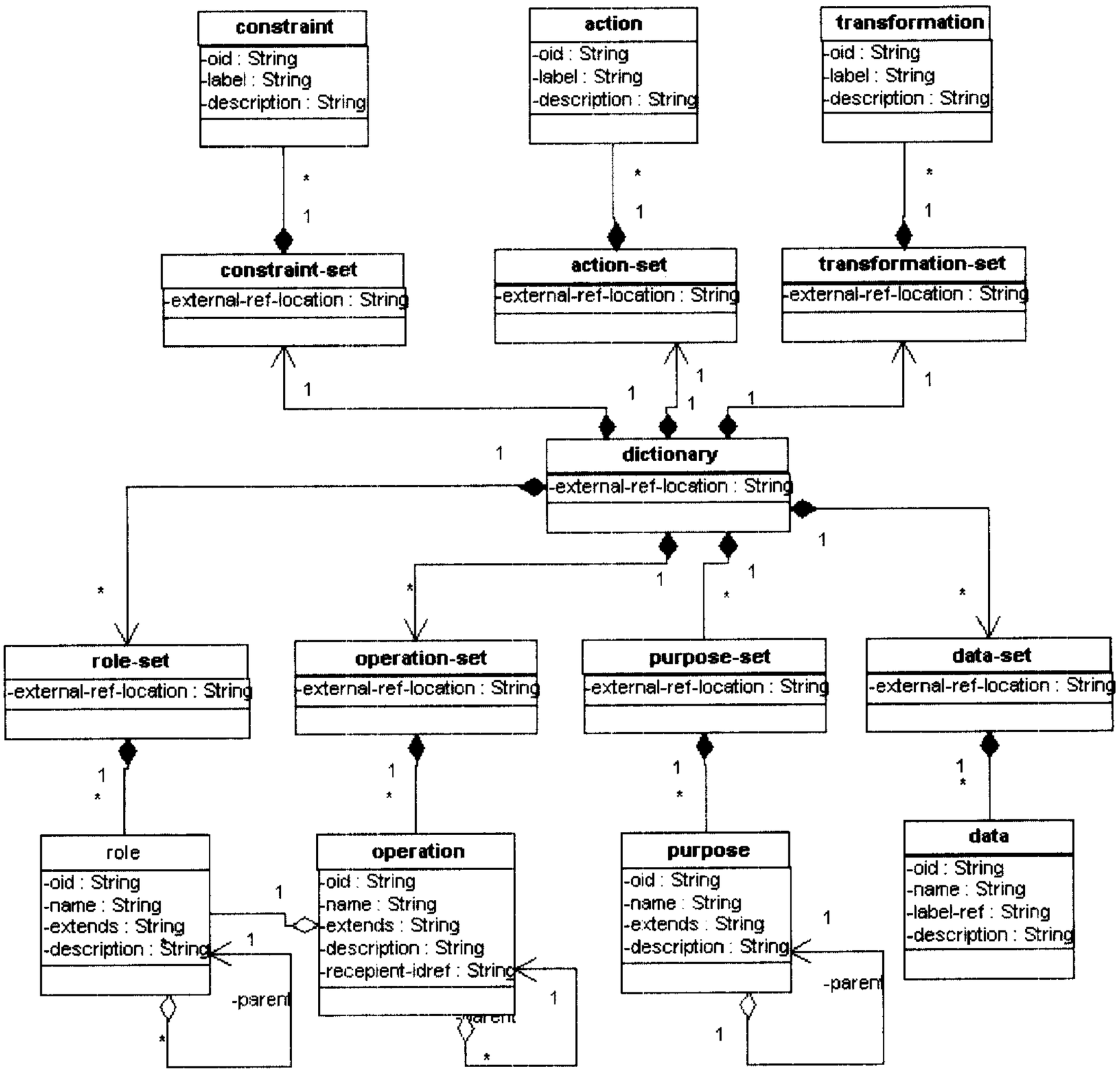
The Resource Definition Framework (RDF) specifies the file header. The header includes document meta-data information such as author, title, subject, and creation date.

Object Dictionary

This section describes the contents of the object dictionary section of a PRML file.

The purpose of the object dictionary is to define all objects that make up the declarations. All objects referenced in the set of declarations must be declared in the object dictionary. The dictionary includes exactly seven sets- one for each type of PRML object: `<role-set>`, `<operation-set>`, `<purpose-set>`, `<data-set>`, `<constraint-set>`, `<action-set>`, and `<transformation-set>`. Each set contains zero or more PRML objects of the corresponding type. Each set may refer to an external PRML file through the `<external-ref-location>` element.

The following diagram shows the data model of the PRML object dictionary:



PRML objects defined in the dictionary have several attributes. Every object has an Object ID (OID). The OID must be unique within the class for each organization. This means that one corporation must not have two roles, for example, with the same OID. However, a role and a purpose may (but are never required to) share the same OID. PRML does not use the ID and IDREF keywords as specified in the XML specification because the PRML document may refer to objects residing in a different PRML file. All PRML objects also have human-readable names. All objects may optionally include a human-readable description.

Attributes of PRML Objects:

Element	Description	Datatype	Applicable Objects
`oid`	A unique object ID- must be unique among all objects within a class	Key	All
`name`	Human readable name	String	All
`Description`	Human readable description (optional)	String	All
`extends`	Reference of the oid(s) of the base object (optional)	Key	Roles, Purposes, and Operations
`label-ref`	Reference of the label of a <data-def> element in the <data-	Key	Data Elements

	schema> section. (See section 0)		
recipient-idref	Reference to the oid of the role on the receiving end of this operation (typically used in the case of data sharing) (optional)	Key	Operations
label	The label is assumed to refer to a function that implements the constraint, action, or tranformation (optional)	String	Constraints, Actions, and Transformations

A description of attributes specific to certain objects, and corresponding examples for each object type follow.

Roles

Examples of roles include 'customer care rep', '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 prohibits (which is implied by not granting) access to a certain role, regardless of which specific user happens to be playing that role.

One role can extend multiple other roles. The fact that an role X extends role Y means that if a declaration exists that contains a reference to X, an implied declaration exists with a reference to Y. For example, a child operation 'credit check' may extend a base operation 'read.' This means that if the operation 'credit check' is permissible, the base operation 'read' is also implied to be permissible. To extend an object, the <extends> element of the child object should reference the base object.

Example:

```
<role-set>
  <role>
    <oid>ROLE-001</oid>
    <name>CUSTOMER_CARE_REP</name>
    <description>Customer care reps are users in the CS
  department.</description>
  </role>
  <role>
    <oid>ROLE-002</oid>
    <name>CUSTOMER_CARE_SUPERVISOR</name>
    <description>Customer care supervisors are extensions of customer care
  reps.</description>
    <extends>
      <base-id>ROLE-001</base-id>
    </extends>
  </role>
</role-set>
```

Operations

Examples of operations include 'read', 'update', 'delete', 'send email,' and 'send surfing profile to ad network.' Operations can define complex functions that require the combination of several atomic operations. Examples of non-atomic operations include 'send email,' and 'check credit.'

The optional <recipient-idref> element is used to reference the recipient role for operations that share data with a particular role. A declaration may allow sharing of data only with a particular recipient.

Similar to roles, operations may form a hierarchy using the <extends> element.

Example:

```
<operation-set>
  <operation>
    <oid>OPERATION-001</oid>
    <name>Read</name>
  </operation>
  <operation>
    <oid>OPERATION-002</oid>
    <name>Send surfing profile to ad network</name>
    <extends>
      <base-id>OPERATION-001</base-id>
    </extends>
    <recipient-idref>ROLE_001</recipient-idref>
  </operation>
</operation-set>
```

Purposes

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

Simila to roles and operations, purposes may form a hierarchy using the `<extends>` element.

Example:

```
<purpose-set>
  <purpose>
    <oid>PURPOSE-001</oid>
    <name>providing customer care</name>
    <description />
  </purpose>
</purpose-set>
```

Data Elements

Data elements are objects that group various pieces of data as defined in the data schema. The decoupling of physical data location and declarations isolates the declarations from changes of the data location (e.g. changes to a database schema). When a database schema changes, declarations remain unchanged unless personal information is added or removed. For a description of how phyical data locations are represented, see section 0.

Example:

```
<data-set>
  <data>
    <oid>DATA-001</oid>
    <name>email</name>
    <label-ref>email</label-ref>
  </data>
</data-set>
```

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 which is implemented by some application.

Example:

```
<constraint-set>
  <constraint>
    <oid>CONSTRAINT-001</oid>
```

```
    <name>Consent has been granted by the user referred to by the data
  element</name>
    <label>OracleDB.StoredProcs.IsConsent</label>
   </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 label 'OracleDB.StoredProcs.IsConsent' refers to a function that returns true if the declaration should be permissible and false otherwise.

Transformations

When a declaration refers to a data-access operation, an optional transformation object can be specified. Transformations control how the data should be changed prior to delivering it to the requesting user when the data access operation is granted. For example, a declaration can request the data be generalized before a particular role can read the data. As for constraints, the label is assumed to refer to a function that implements the transformation.

Example:

```
<transformation-set>
   <transformation>
    <oid>TRANFORMATION-001</oid>
    <name>Remove HIV test results</name>
    <label>OracleDB.StoredProcs.RemoveHIVResult</label>
  </action>
</action-set>
```

Actions

Actions are procedures that should be executed whenever a declaration takes effect. As for constraints and transformations, the label is assumed to refer to a function that implements the action.

Example:

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

Declaration Set

A privacy policy consists of multiple PRML declarations. Each declaration creates a permissible relationship between PRML objects as defined in the Object Dictionary. 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 elements for any purpose.
- If a declaration exists for a relationship, then that relationship is permissible.

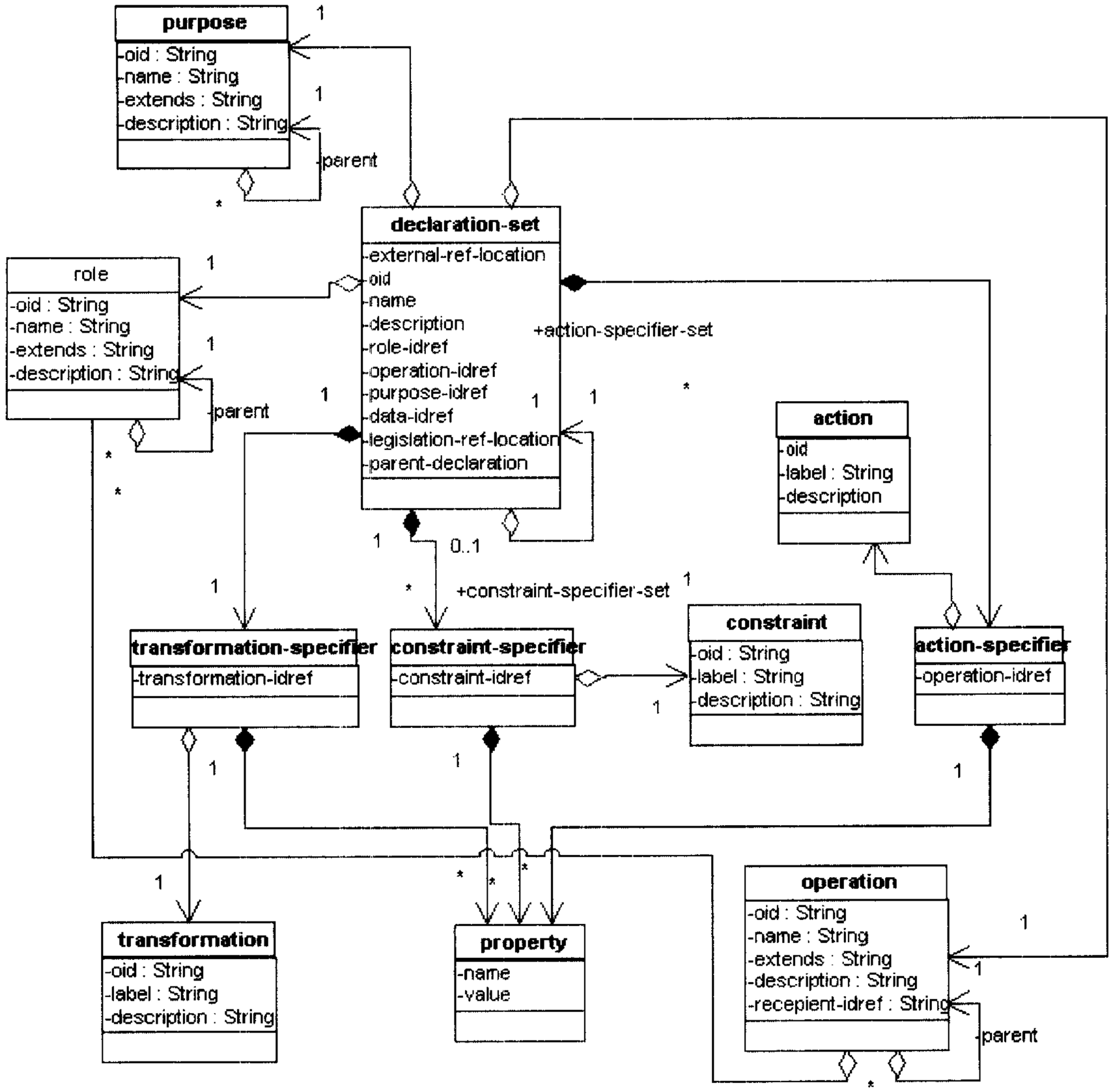
A declaration object consists of the following elements:

Element	Description	Datatype
oid	A unique object ID- must be unique among all declarations	Key

name	Human readable name	String
description	Human readable description (optional)	String
role-idref	Reference to an oid of a role object	Key
operation-idref	Reference to an oid of an operation object	Key
purpose-idref	Reference to an oid of a purpose object	Key
data-idref	Reference to an oid of a data object	Key
transformation-specifier	A transformation and (optionally) corresponding parameters. The paramters are passed to the function that implements the transformation.	See diagram below
constraint-specifier-set	A set of constraints and (optionally) corresponding parameters. The paramters are passed to the function that implements the constraint.	See diagram below
action-specifier-set	A set of actions and (optionally) corresponding parameters. The paramters are passed to the function that implements the action.	See diagram below
Parent-declaration	Reference of the oid of the parent declaration (used if this is an implicit declaration) (optional)	Key
legislation-ref-location	The URI is the legislation that mandates the existence if this declaration (optional)	UriReference

The three specified sets refer to an constraint. Action, and transformation objects within the dictionary and provide parameters. The parameters are not defined by this specification; they are used by implementers of transformations, constraints, and actions to allow implementation re-use by multiple declarations.

The static diagram is as follows:



1

Example Declaration:

```
<declaration-set>
  <declaration>
    <oid>DECLARATION-001</oid>
    <name>purge-email</name>
    <description>This declaration allows the purging of an email address.
    <role-idref>ROLE-002</role-idref>
    <operation-idref>OPERATION-001</operation-idref>
    <purpose-idref>PURPOSE-002</purpose-idref>
    <data-idref>DATA-001</data-idref>
    <action-specifier-set>
      <action-specifier>
        <operation-idref>ACTION-004</operation-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-set>
```

Explicit and Implicit Declarations

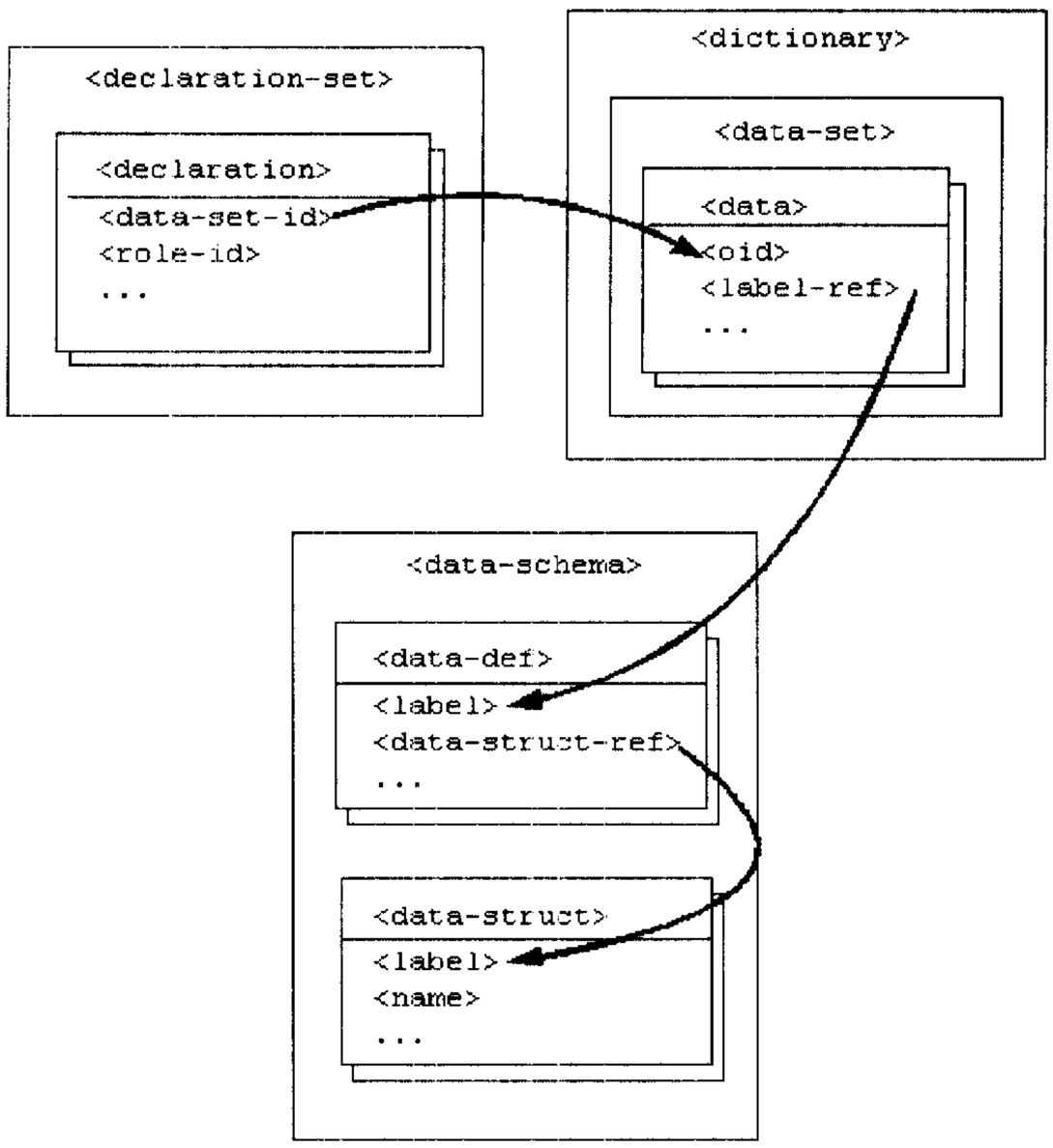
In the Object Dictionary, roles, operations, and purposes can be structured in a hierarchy using the <extents> relationship (see section 0). When a declaration contains a reference to an object X that has an <extents> relationship (the object X extends an object Y, for example), implicit declarations exist. The implicit declaration is a declaration that is identical to the original declaration with the exception that object Y replaces object X in the implied declaration. 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.

Data Schema

The goal of data representation in PRML is to provide sufficient abstraction to eliminate the need to re-write declarations when the storage of data changes (e.g. a database data structure is modified), while providing the ability to reference a data element's exact location (e.g. a specific column in a database).

The following diagram demonstrates how data is represented and referenced:



Declarations must contain a <data-set-id> element. The <data-set-id> element refers (via an <oid>) to a <data> element in the <data-set> of the <dictionary>. A <data> element contains a <label-ref> element which references (via a <label>) a <data-def> element in the <data-schema>. An optional (or several optional) <data-struct-ref> element(s) contained in a <data-def> element reference (via a <label>) a <data-struct> element which is contained in the <data-schema> element. The <data-struct> element contains a <name> element which identifies where the data is. The format of this description is not specified, but could be, for example, the "database.schema.table.column" tuple where the data is physically located.

Example:

```
<declaration-set>
  <declaration>
    ...
    <data-set-id>DATA-001</data-set-id>
  </declaration>
</declaration-set>

<dictionary>
  <data-set import="data-def.xml">>
    <data>
      <oid>DATA-001</oid>
      <name>Customer Contact Info
      <label-ref>#customer-contact-info</label-ref>
    </data>
   </data-set>
</dictionary>

The contents of the data-def.xml file:

<data-schema>
  <data-def>
    <label>customer-contact-info</label>
    <data-struct-ref>#custEmailAddr</data-struct-ref>
    <data-struct-ref>#custName</data-struct-ref>
  </data-def>
  <data-struct>
    <label>custName.firstName</label>
    <name>OracleDB.CUST_INFO.CUST_TABLE.NAME</name>
  </data-struct>
  <data-struct>
    <label>custName.lastName</label>
    <name>OracleDB.CUST_INFO.CUST_TABLE.NAME</name>
  </data-struct>
  <data-struct>
    <label>custEmailAddr</label>
    <name>OracleDB.CUST_INFO.CUST_TABLE.EMAIL</name>
  </data-struct>
  <data-def>
    <label>customer-surfing-profile</label>
    <data-struct-ref>http://someplace.com/schema#profile</data-struct-ref>
  </data-def>
</data-schema>
```

- A <data-def> element is not required to contain a <data-struct-ref> element . If no <data-struct-ref> element exists, the data exists, but its description is unavailable or not required.

- <data-set> elements can include the import=URI attribute which will indicate that the specified data is described in a <data-schema> element of the referenced document. It is recommended that Data Schemas be defined in a separate file, so this attribute should typically be present, but it is not required. If this attribute is not present, the PRML file must contain a <data-schema> that describes the <data> items.
- There must be exactly one <data-set-id> per declaration.
- When referencing a <data-def> or <data-struct> which is contained in the document, you must use the URI convention of placing a hash ('#') character in front of the name. This character does not appear in the <name> element.
- The <data-struct> elements describe the structure of various types of data elements. Note that different <data-def> elements can actually have the same structure simply by pointing to the same <data-struct> element(s). Each <data-struct> can optionally point to a local or remote <data-struct> that further defines the description of the data.
- A <data-struct-ref> element can reference multiple <data-struct> elements. The <data-struct-ref> string references all <data-struct> elements for which the <data-struct-ref> string is a prefix of the <data-struct-ref> <label> element. The same logic applies to a <data> <label-ref> element and a <data-def> <label> element.

Conversion between PRML Data Schema and P3P

The following guidelines indicate the relationship and can be used to assist in the conversion from one format to the other. Familiarity with the P3P specification is assumed.

- PRML data definitions provide a name and an optional description. There is no "short-description" attribute, which can be specified so these are never generated when converting to a P3P data schema.
- P3P data elements have an attribute "optional" while PRML does not. This attribute (defaulted to "no") indicates whether or not a visitor to a site can withhold the specified piece of data. When converting from PRML to P3P, this value should be explicitly set to "no." Since PRML deals with releasing data rather than collecting it, a visitor to the site should be obliged to provide it.
- PRML does not define data categories. P3P can attach categories to DATA, DATA-DEF or DATA-STRUCT elements in order to provide a hint regarding the intended use of the data. This must be specified somewhere inside a P3P data schema. One means is to use P3P's extension mechanism and assign the following for each DATA-DEF:
<CATEGORY><other-category>PRML Data Schema</other-category></CATEGORIES>
- The <data-set> element maps directly to DATA-GROUP. <data-set> can specify an "import" attribute. This also maps directly to "base". It is assumed that the PRML data-schema will always be in a separate file. In this case, the link to that file will be identified through a "base" attribute specified for the <DATA-GROUP> element. If the PRML data-schema is exported to the P3P file itself, the "base" attribute value must be set to the empty string ("").
- When converting PRML <data> elements to P3P <DATA> elements, the <name> element must be converted to the attribute "ref".
- The <data-def> element maps to P3P's <DATA-DEF>. The <name> element becomes the "name" attribute. Similarly, the <name> element of a <struct-ref> element becomes a "structref" parameter. There is no equivalent to the "short-description" attribute. Since this is optional in P3P, the conversion process does not specify it.
- The PRML <data-struct> elements map to P3P's <DATA-STRUCT> elements and are treated the same way as <data-def> elements.
- Within PRML data definitions, instances of <description> elements become <LONG-DESCRIPTION> when transferred to P3P data schemas.

Examples- Privacy Policies Expressed in PRML

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

Sample PRML Document A

The following policy is encoded in the PRML document below:

- A customer care rep (role) may create (operation) an e-mail address (data element) in order to create an account (purpose). This email address must be deleted from the system no later than thirty (30) days after its creation (action). A customer care rep (role) may read (operation) a customer's email address (data element) 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) the email address (data element) in order to satisfy the company's minimal retention policy (purpose).

```
<?xml version="1.0" ?>
<!DOCTYPE prml SYSTEM "prml-v1.0.dtd">
<prml>
<RDF xmlns:dc="http://purl.org/metadata/dublin_core#">
<Description />
</RDF>
<dictionary>
  <role-set>
    <role>
      <oid>ROLE-CUSTOMER-CARE-REP</oid>
      <name>Customer care rep</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
                customer database respects minimal retention
                guidelines.</description>
    </role>
  </role-set>
  <operation-set>
    <operation>
      <oid>OPERATION-DELETE</oid>
      <name>delete</name>
      <description />
    </operation>
    <operation>
      <oid>OPERATION-CREATE</oid>
      <name>create</name>
      <description />
    </operation>
    <operation>
      <oid>OPERATION-SEND-EMAIL</oid>
      <name>Send Email</name>
      <description>email correspondence</description>
    </operation>
  </operation-set>
  <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>
  <data-set>
    <data>
      <oid>DATA-EMAIL</oid>
      <name>email</name>
      <label-ref>#email</label-ref>
    </data>
  </data-set>
  <constraint-set>
    <constraint>
      <oid>CONSTRAINT-PENDING-TRANSACTION</oid>
      <label>pending-transaction-exists</label>
      <name>There is an pending transaction.</name>
    </constraint>
  </constraint-set>
  <transformation-set />
  <action-set>
    <action>
      <oid>ACTION-PURGE-EMAIL-ADDRESS-TIMER</oid>
      <label>purge-email-address-timer</label>
      <name>purge-email-address-timer</name>
    </action>
  </action-set>
</dictionary>

<data-schema>
  <data-def>
    <label>email</label>
    <data-struct-ref>userdata.email</data-struct-ref>
  </data-def>
  <data-struct>
    <label>userdata.email</label>
    <name>OracleDB.schema.table.email</name>
  </data-struct>
</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-idref>DATA-EMAIL</data-idref>
    <action-specifier-set>
      <action-specifier>
        <operation-idref>ACTION-PURGE-EMAIL-ADDRESS-TIMER</operation-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-idref>DATA-EMAIL</data-idref>
  <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-idref>DATA-EMAIL</data-idref>
  </declaration>
</declaration-set>
</prml>
```

Sample PRML Document B

The following policy is encoded in the PRML document below:

- Marketing representatives (role) may send email (operation) in order to do targeted advertising (purpose).

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

- Marketing representatives (role) may read (operation) a customer's email addres (data) in order to do targeted advertising (purpose).

```
<?xml version="1.0" ?>
<!DOCTYPE prml SYSTEM "prml-v1.0.dtd">
<prml>
<RDF xmlns:dc="http://purl.org/metadata/dublin_core#">
<Description />
</RDF>

<dictionary>
  <role-set>
    <role>
      <oid>Role-1</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-1</oid>
    <name>Read</name>
  </operation>
  <operation>
    <oid>Operation-2</oid>
    <name>Send Email</name>
```

```
    <extends>
      <base-id>Operation-1</base-id>
    </extends>
  </operation>
</operation-set>
<purpose-set>
  <purpose>
    <oid>Purpose-1</oid>
    <name>Targetted advertising</name>
  </purpose>
</purpose-set>
<data-set>
  <data>
    <oid>Data-1</oid>
    <name>Customer's email address</name>
      <label-ref>email-address</label-ref>
    </data>
  </data-set>
  <constraint-set/>
  <transformation-set />
  <action-set />
</dictionary>

<data-schema>
  <data-def>
    <label>email-address</label>
    <data-struct-ref>#userdata.email_address</data-struct-ref>
  </data-def>
  <data-struct>
    <label>userdata.email_address</label>
    <name>OracleDB.userschema.userdata.email_address</name>
  </data-struct>
</data-schema>

<declaration-set>
  <declaration>
    <oid>Declaration-1</oid>
    <name>Reading email address to provide targeted advertising</name>
    <role-idref>Role-1</role-idref>
    <operation-idref>Operation-2</operation-idref>
    <purpose-idref>Purpose-1</purpose-idref>
    <data-idref>Data-1</data-idref>
  </declaration>
</declaration-set>

</prml>
```

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 Element: Customer email address

- Data Element: Customer contact info
- Purpose: Targeted marketing

An example of base declaration is:

- When a PRML document (data element) is modified (operation) (and thus a privacy policy changed) for any reason (purpose), all individuals that have consented to the current policy must be notified (action).

Assigning Declarations on Query Results

In order to formalize some policy declarations one needs to refer to the aggregated data structures. It is conservable 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: Air-miles points are collected for every transaction completed by the customer. The total number of points is not stored in the DB, but retrieved from the database as a sum of points in all transactions. The role 'customer representative' is allowed to read total number points. The same role isn't allowed to read details of individual transactions.

PRML currently does not support declarations on aggregated data structures.

Bin-size Declarations

Some policy declarations allow a role full access to PII for the purpose of data mining with the condition that people do not run queries that identify an individual. Such declarations must constrain the 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.

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 extended to provide meta-data on the more granular level of PRML objects and declarations.

Extending the Roles Model

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.

UML to XML Mapping

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

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

Scheme Datatypes standard. Others are PRML definitions of new XML Scheme 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;)>
```

PRML DTDs

prml-v1.0.dtd

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

    $Id: prml-v1.0.dtd,v 1.9 2001/04/10 18:33:25 alexis Exp $

    File: prml-v1.0.dtd

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

    PRML v1.0 DTD

    ============================================================
-->

<!-- 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, data-set,
  constraint-set?, transformation-set?, action-set?)>

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

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

prml-v1.0-rdf.dtd

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

    $Id: prml-v1.0-rdf.dtd,v 1.1 2001/03/09 14:24:11 alexis 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?, subject?)>
<!ELEMENT title (%string;)>
<!ELEMENT creator (%string;)>
<!ELEMENT date (%string;)>
<!ELEMENT subject (%string;)>
```

prml-v1.0-sup.dtd

```
<!--
    ================================================================
    $Id: prml-v1.0-sup.dtd,v 1.6 2001/04/16 19:36:51 philippem Exp $

    PRML supplemental definitions and declarations.  Taken shamelessly
    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 % Labeled " label">
    <!-- properties -->
        <!ELEMENT label (%string;)>

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

<!ENTITY % ExtendsMultiple " extends?">
    <!-- properties -->
        <!ELEMENT extends (base-id+)>
        <!ELEMENT base-id (%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;)>
```

prml-v1.0-types.dtd

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

    $Id: prml-v1.0-types.dtd,v 1.2 2001/04/10 18:33:25 alexis 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 -->

<!ENTITY % currencyValue " %string;">
<!-- format is a [ISO 3-character country code]:[decimal number] -->

<!ENTITY % ipAddress " %string;">
<!-- format is a either the dotted-decimal string or the complete tcp
  host.domain name -->

<!ENTITY % numericPriority " %integer;">
<!-- format is an integer between 1 and 10, with 1 being the highest
  priority -->

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

prml-v1.0-data.dtd

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

    $Id: prml-v1.0-data.dtd,v 1.5 2001/03/21 16:49:33 philippem 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.

    ============================================================================
-->

<!ENTITY % DataSet " data*">
<!ENTITY % Data " %Identifiable;,
                  %Named;,
                  %Describable;,
                  label-ref+">


<!ELEMENT data-set (%DataSet;) >

<!ELEMENT data (%Data;)>

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

<!ELEMENT label-ref (%string;) >

<!ELEMENT data-struct-ref (%string;) >

<!ELEMENT data-def (%Labeled;,
                    %Named;,
                    data-struct-ref+) >
        <!ATTLIST data-def %TypeAttribute; >

        <!ELEMENT data-struct (%Labeled;,
                               %Named;,
                               %Describable;) >

        <!ELEMENT data-schema (data-def | data-struct)* >
```

prml-v1.0-decl.dtd

```
        <!--

          ============================================================================

            $Id: prml-v1.0-decl.dtd,v 1.12 2001/04/11 20:50:22 philippem 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 " %Identifiable;,
                         name?,
                         %Describable;,
                         parent-declaration?,
                         legislation-ref-location?,
                         role-idref,
                         operation-idref,
                         purpose-idref,
                         data-idref,
                         transformation-specifier?,
                         constraint-specifier-set?,
                         action-specifier-set?">

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

<!ELEMENT declaration (%Declaration;)>
<!ELEMENT legislation-ref-location (%uriReference;)>

<!ELEMENT parent-declaration (%key;)>
```

prml-v1.0-roles.dtd

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

    $Id: prml-v1.0-roles.dtd,v 1.3 2001/03/21 16:49:41 philippem 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 " %Identifiable;,
                  %Named;,
                  %Describable;,
                  %ExtendsMultiple;">

<!ELEMENT role-set (%RoleSet;) >

<!ELEMENT role (%Role;)>

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

prml-v1.0-ops.dtd

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

    $Id: prml-v1.0-ops.dtd,v 1.3 2001/03/21 16:49:37 philippem 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 " %Identifiable;,
                       %Named;,
                       %Describable;,
                       %ExtendsMultiple;,
                       recipient-idref?">

<!ELEMENT recipient-idref (%key;)>

<!ELEMENT operation-set (%OperationSet;) >

<!ELEMENT operation (%Operation;)>

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

prml-v1.0-purpose.dtd

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

    $Id: prml-v1.0-purpose.dtd,v 1.3 2001/03/21 16:49:39 philippem 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 " %Identifiable;,
                     %Named;,
                     %Describable;,
                     %ExtendsMultiple;">

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

<!ELEMENT purpose (%Purpose;)>

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

prml-v1.0-constraint.dtd

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

    $Id: prml-v1.0-constraint.dtd,v 1.6 2001/04/10 18:33:26 alexis 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 " %Identifiable;,
                        %Named;,
                        %Labeled;,
                        %Describable;">

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

<!ELEMENT constraint (%Constraint;)>

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

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

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

prml-v1.0-action.dtd

```
<!--

  ============================================================================

    $Id: prml-v1.0-action.dtd,v 1.3 2001/04/11 20:50:20 philippem 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 " %Identifiable;,
                    %Named;,
                    %Labeled;,
                    %Describable;">

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

<!ELEMENT action (%Action;)>

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

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

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