On Parametric Obligation Policies: Enabling Privacy-aware Information Lifecycle Management in Enterprises

Marco Casassa Mont, Filipe Beato
Trusted Systems Laboratory
HP Laboratories Bristol
HPL-2007-7
January 29, 2007*

identity management, privacy, information lifecycle management, obligation policies, obligation management system, scalability

Enterprises that collect and process personal data must deal with related privacy management issues. It is not just a matter of privacy-aware access control: privacy obligation policies, dictating duties and expectations on how personal data has to be handled, must be considered too. The management of obligation policies is a promising area (affecting the lifecycle management of personal data) but it is still underestimated. Enterprises require solutions that enable automation and leverage their current identity management solutions. HP Labs have been working on this topic in the last few years, also in the context of the EU PRIME project. In this paper we present our recent work on parametric obligation policies and a related obligation management framework to deal with a scalable management of these obligation policies on large amounts of data, stored in distributed data repositories.

* Internal Accession Date Only
Approved for External Publication

On Parametric Obligation Policies: Enabling Privacy-aware Information Lifecycle Management in Enterprises

Marco Casassa Mont, Filipe Beato
Hewlett-Packard Laboratories, Trusted Systems Lab, Bristol, UK
marco.casassa-mont@hp.com, filipe.beato@hp.com

Abstract

Enterprises that collect and process personal data must deal with related privacy management issues. It is not just a matter of privacy-aware access control: privacy obligation policies, dictating duties and expectations on how personal data has to be handled, must be considered too. The management of obligation policies is a promising area (affecting the lifecycle management of personal data) but it is still underestimated. Enterprises require solutions that enable automation and leverage their current identity management solutions. HP Labs have been working on this topic in the last few years, also in the context of the EU PRIME project. In this paper we present our recent work on parametric obligation policies and a related obligation management framework to deal with a scalable management of these obligation policies on large amounts of data, stored in distributed data repositories.

1. Introduction

Personal data, digital identities and users' profiles are collected by enterprises and other organizations to enable their business processes and provide required services. Privacy laws and legislation [1,2] dictate policies and constraints on how this personal data should be handled, stored, processed and disclosed by enterprises. Part of these policies have an impact on access control aspects i.e. how data should be accessed, based on data subjects' consent, stated purposes for collecting data, etc. [1,2]. Another part of these policies dictate obligations that enterprises need to fulfill on collected data, i.e. expectations and duties on how to handle this data in terms of data retention/deletion, notifications, data transformation, etc. [1,2].

This paper focuses on ***privacy obligation policies***. The management of obligations has an impact on how the lifecycle of personal data is handled in distributed data repositories and systems within enterprises [15]. This area is still underestimated and open to innovation. HP Labs have been working on this topic in the last few years, both in the context of the EU PRIME project [3] and internal R&D projects. Our aim is to provide a pragmatic approach to the representation, management and enforcement of obligation policies to be deployed within enterprise IT infrastructures, in particular state-of-the-art identity management solutions. This is a key requirement made by enterprises, as well as the need for automation and cost reduction.

This paper provides an overview of our vision and previous work in the area of obligation policies and their management: it describes lessons we learnt, including the need to address scalability issues. It then focuses on our recent R&D work on *parametric obligation policies* that aims at addressing these issues. This paper is a follow-up of [4]: it introduces our actual approach to *parametric obligation policies* and provides details on how these policies are represented. It also describes a related *scalable obligation management system* prototype that has been fully implemented and, as a proof-of-concept, integrated with an HP state-of-the-art identity management solution. Technical details are provided, as well as a comparison against related work.

2. Our Vision on Privacy Obligation Policies

Privacy obligations are policies that dictate constraints, expectations and duties on how personal data must be managed by enterprises [5]. They require dealing with data deletion, data retention, data transformation and minimisation, notifications, execution of (potentially complex) workflows on data by involving human and system interactions, etc. [5]. Privacy obligations could be short-termed, long-termed or have ongoing implications [5]. Their management and enforcement is at the very core to enable privacy-aware information lifecycle management in enterprises [5,6,15]. In previous papers on this topic [5,6] we

argued on the importance of explicitly dealing with privacy obligation policies as first-class entities i.e. by not subordinating their representation and enforcement to access control policies - as, instead, mandated by related work, for example EPAL [7], XACML [8], etc.

This is a key aspect of our vision. The main reason for this is that access control policies and their enforcement framework are not fully suitable to capture and manage a broad variety of obligation policy aspects. For example, deletion of personal data after a predefined period of time must happen at the right time, independently if that data has ever been accessed.

Our approach has been refined and implemented both in PRIME and HP Labs projects. In our vision, a privacy obligation policy is a self contained entity having a *unique identifier* and consisting of: *Target*, *Events* and *Actions* sections. The *Target* section describes the storage location and properties of sensitive data (e.g. personal data, digital credentials, user profile, etc.) subject to the obligation policies. The *Events* section describes a logical combination of *events* (e.g. time-based event, context-based events), that - once happen - trigger the enforcement of the obligation. Specifically, an event happens if its (internal) conditions are satisfied. The *Actions* section describes the actions to be enforced (e.g. deletion of data, sending notifications, etc.) once events are satisfied. Simple examples of privacy obligations are: (1) "Delete credit card details of User X at time T and Notify this User"; (2) "Notify Administrator A if financial details of User X have been accessed more that Y times in T hours"; (3) "Execute Workflow W on Information X of User Y if Context C has property P". More details can be found in [6].

From an **operational perspective** (i.e. actual representation of privacy obligation policies in a format that can be programmatically interpreted, managed and enforced) we proposed an explicit representation of obligation policies in an XML format, as *reactive rul*es: **WHEN *Events* happens THEN trigger the execution of *Actions* on *Target***. Details and examples can be found in [6]. Based on our XML representation of obligation policies, we also proposed an *obligation management framework model* and a related *obligation management system* to interpret, schedule, enforce and monitor these policies [5,6]. Our obligation management technology and framework was designed to allow users (at the time of disclosing their personal data or afterwards) to express *privacy preferences* (e.g. on deletion time of some of their attributes or notification preference) on how their personal data should be handled by the enterprise. Our *obligation management system* was then able to automatically derive and instantiate related obligation policies based on these privacy preferences. We achieved this capability by introducing the concept of obligation policy *template*. In our approach, a template consisted basically of an obligation policy which contained simple "placeholders" in its *Events* and *Actions* sections [4]. Templates were defined upfront, by privacy administrators, to cover all the types of obligations supported by an enterprise. In this context, a template was instantiated just by replacing its placeholders with the actual privacy preference values (for example a deletion date or a notification preference, etc.).

In this context an "instantiated" obligation policy was (1) uniquely associated to a piece of data and (2) it embedded privacy preferences in its *Events* and *Actions* sections. The resulting "instantiated" obligation policies were then scheduled, enforced and monitored by our obligation management system [5,6]. A working prototype was fully implemented and integrated with HP Select Identity [9], a state-of-the-art identity management solution, to demonstrate the feasibility of our ideas and its deployment in enterprise contexts.

2.1. Lessons Learnt

In PRIME we initially explored how to represent privacy obligation policies in RDF [13] (in the context of semantic web [14]), coupled with ontologies [13] to reason on involved types of data. Our RDF-based policies (basically graphs of inter-related policy components) where directly associated to RDF "graphs" representing personal data and stored in protected data repositories. We soon discovered that this representation was too heavy, in terms of the redundancy introduced in the notation and explosion of stored data (as triples) [13]. This approach was also limiting the expressiveness of our policies, hence the decision to use XML instead. This is currently work in progress in PRIME: we are exploring alternative and more efficient RDF representations of policies.

In the meanwhile, the implementation of our prototype (and a related demonstrator), related tests and feedback received by HP customers/third parties helped us to identify another key problem: the scalability of our approach. On one hand our approach provided great flexibility in defining a broad range of privacy obligation policies, potentially customisable to users' needs and directly associated to personal data. On the other hand *for each piece of managed data* (and related privacy preferences), *one or more "instances" of our obligation policies had to be created and associated to this data*, as shown in Figure 1.

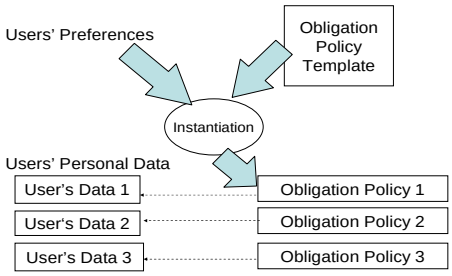

Figure 1. Association of Obligation Policies to Data

In real world scenarios, large amounts of user's data (greater than 100K records) are collected and managed by enterprises. In our approach, this meant having to deal with a similar (large) amount of associated obligation policies with negative implications and impacts in terms of required resources and processing power to run our obligation management system.

Additional feedback highlighted the need not only to passively monitor failures in enforcing privacy obligations (i.e. spotting cases where the enforcement of stated *Actions* fails or changes in the status of managed data invalidates previously enforced actions [5]) but also being able to proactively remediate to these failures (e.g. by notifying administrators or trying to reinforce failed actions).

Usability tests carried out on our obligation management system (by the Karlstad University, in the context of PRIME) highlighted that end-users are looking for simple ways to express their privacy preferences, via graphical GUI, on a well defined, small and clear set of stated obligation policies. This finding reinforced the validity of our approach based on using pre-defined *templates* for privacy obligation policies, as a way to reduce the "types" of obligation policies to be managed in our obligation management system [4]. This aspect was actually taken into account and implemented in PRIME.

However, the usage of *templates*, on its own, does not solve the scalability problem: even if the enterprise could just define a reduced set of obligation templates, these templates have nevertheless to be instantiated for each piece of managed data - based on related privacy preferences. Hence the scalability problem was still there. A complete analysis of this issue and other related aspects can be found in [4].

3. Addressed Problem and Requirements

The key problem addressed in this paper is how to manage obligation policies in a scalable way, on a potentially large set of personal data stored in various enterprise data repositories. The following related requirements must be satisfied (based on customers' feedback, our analysis and lessons learnt):

- Limit the number of "instantiated" policies (and related management resources) independently on the amount of managed data and related privacy preferences;
- Preserve the key capability to "customize" the management of each individual piece of personal data, based on users' privacy preferences;
- Provide a more comprehensive automation of obligation policies, ensuring that obligations (once enforced) are not only passively monitored but also actions are taken to remediate/react to any violation. This to reduce the need for human intervention in case of large datasets.

Addressing this problem has implication on two key aspects: (1) how to represent obligation policies; (2) how to manage, enforce and monitor these policies.

Next section introduces a reference scenario (used in the remaining part of this paper). Section 5 describes our approach and solution to the problem, based on the concept of *parametric obligation policies*.

4. Scenario

We consider an enterprise scenario where a potential large number of users (customers, employees, etc.) have to disclose their personal data in order to access services. This personal data is provided by users at the registration time, potentially via a web-based self-registration service. In this context a user can check which obligation policies (e.g. in terms of deletion of data, data minimization, notifications, etc.) the enterprise can support (and on which data). The user can make decisions in opting-in/opting-out some of these obligation policies (others might be mandatory). For each selected obligation policy the user can instantiate specific privacy preferences and submit the overall information. The user could later on access this "registration" web service and make changes to their personal data, selected obligations and privacy preferences. A privacy administrator, in the enterprise, can set additional obligation policies (derived from laws and/or internal guidelines) on any subset of collected personal data.

The enterprise can enforce these obligation policies on managed data, by means of a privacy-aware information lifecycle management solution – that leverage our approach and technology. This automates the enforcement of these policies, their monitoring and remediation activities (in case of violation of policies).

5. Parametric Obligation Policies

To address the stated problem and keep into account related requirements, we introduce the concept of *parametric obligation policies*. A *parametric obligation policy* is a policy that leverages the concepts of our previous version of obligation policies [5,6]. The same categories of obligation policies [5,6] are managed. However, the key differences are:

- A **parametric obligation policy** can be associated to a **potentially large set of personal data** (i.e. no multiple instantiations) and, at the same time, it can dictate customized obligation constraints (based on users' privacy preferences) on each data item;
- A parametric obligation policy **does not embed** privacy preferences in its *Events* and *Actions* sections (as instead happens in our previous version of obligation policies). Instead, this policy contains explicit **references** to these preferences, that are stored elsewhere - in data repositories;
- **The Target section** of parametric obligation policies explicitly model and describe the data repositories that will contain preference values pointed by these references - in addition to repositories containing personal data;
- **A new "On Violation" section has been introduced** to explicitly automate the process of "remediation" of violated obligations – as described in the requirement section.

The key feature introduced by parametric obligations is that *privacy preferences* are stored separately from *parametric obligation policies*: references are used to retrieve these preferences. This ensures that a *parametric obligation policy* can apply to a potentially large set of personal data – as defined in its *Target* element – and, at the same time, allows the "customization" of its *Events* and *Actions* based on references to external privacy preferences.

From a **formal perspective** a *parametric obligation policy* is a **<i,t,L(e[r]),C(a[r]),C(va[r])>** tuple, where **<i,t,e,a,va>** $\in$ **<I, 2^T, 2^E, 2^A, 2^{VA}>** and **r** $\in$ 2^R :

- **I**: set of unique identifiers, associated to parametric obligation policies;
- **T**: set of possible obligation ***targets***, i.e. data entities subject to obligations;
- **E**: set of possible ***parametric events*** that can trigger an obligation i.e. events that might contain references (e.g. to privacy preferences);
- **A**: set of all possible ***parametric actions*** that can be executed as an effect of enforcing an obligation i.e. actions that might contain references (e.g. to privacy preferences);
- **VA**: set of all possible ***parametric "on violation" actions*** to be executed to remediate any violation of enforced (parametric) obligations. These actions might contain references to preferences as well;
- **R**: set of all possible references (e.g. to privacy preferences) that could be used in a policy.

Specifically, this **<i,t,e,a,va>** tuple is defined as:

- **i** $\in$ **I**: *i* is an element that belongs to I;
- **t** $\subseteq$ **T**: *t* is a set of targets included in T;
- **e[r]** $\subseteq$ **E**: *e[r]* is a set of parametric events included in E;
- **a[r]** $\subseteq$ ***A***: *a[r]* is a set of parametric actions included in A;
- **va[r]** $\subseteq$ ***VA***: va[r] is a set of parametric "on violation" actions included in VA;
- **r** $\subseteq$ **R**: *r* is a set of references (to values) included in R.

In this context the ***L*** operator and the ***C*** operator mean:

- **L(e[r])**: a logical combination of parametric events, for example AND, OR and NOT combination of events contained in *e*;
- **C(a[r])**: a combination of parametric actions, such as a sequence of actions;
- ***C(va[r])***: a combination of parametric actions to be executed in a sequence, when an enforced obligation is violated.

A set of *parametric obligation policies* can be created by a privacy administrator to dictate the "criteria" by which personal data should be handled: the referencing mechanism (coupled to appropriate data descriptions in the *Target* section) ensures that these policies are "instantiated" on-the-fly by our obligation management system - based on associated privacy preferences, enforced and monitored on a potentially large set of managed data (Sections 6,7).

From an **operational perspective** a parametric obligation policy in still represented as an XML format, as a *reactive rule*. XML has been used because of its versatility and suitability to extensions. The XML skeleton of a parametric obligation policy is:

```
<?xml version="1.0"?>
<obligation oid="">
    <target> …….. …</target>
    <metadata> … ….</metadata>
    <events> …….</events>
    <actions> ……</actions>
    <onViolation> …</onViolation>
</obligation>
```

The remaining part of this section provides more details about the actual content of parametric obligation policies i.e. their *Target*, *Metadata*, *Events*,

Actions, *OnViolation* sections. For illustration purposes we consider a simplified scenario based on Section 4. This scenario consists of an e-commerce site that collects personal data about users and their preferences and stores this information in database tables. In this context, we consider a very simple parametric obligation policy dictating that: for each piece of managed personal data (*Target*), credit card information must be deleted (*Parametric Action*) based on time-based deadlines specified by users via their privacy preference (*Parametric Event*). When this happens the correspondent user must be notified (*Parametric Action*). Should the enforcement of any of these actions fail, the obligation management system should try to reinforce them and notify an administrator (*"On Violation" Actions*).

5.1. Target

The *Target* section of a parametric obligation policy is used to provide the following information:

- **A description of data repositories containing (personal) data** that is subject to privacy obligations. In this context one or more data repositories can be described (e.g. RDBMS database or LDAP directory, etc.). A data repository description includes location and name of the data repository, data schema structures (e.g. database tables) and primary keys. It is important to notice, that by default, all data stored in these repositories will be affected by this obligation policy. A more selective choice of which data items must be managed can be made by instantiating a "Conditions" sub-section (e.g. by testing properties/values of the stored data).
  Each data repository is identified by a unique *alias* that is used as a shortcut in other parts of the parametric obligation. If multiple data repositories are described, it is possible to specify any relationship (i.e. links between primary keys) existing on data stored in these repositories;
- **A description of data repositories used to store *privacy preferences*.** The definition of this sub-section is identical to the previous one, with the exception that it refers to repositories storing preferences/parameters. These preferences are associated to the managed personal data and used to customize other sections of the privacy obligations;
- **A cross-links sub-section** defining how to link *preferences* to *personal data*, by using relevant keys defined in the other two sub-sections.

The XML skeleton of the *Target* section (low-level details have been omitted for space reasons) follows:

```
<target>
   <DataRepositories>
     <Repositories>
       <DataRepository alias= "…"
          <DRType> … ……..</DRType>
          <DBname> … …….</DBname>
          <TableName>……...</TableName>
          <Conditions>
              <Condition> …..</Condition>
          </Conditions>
          <UniqueIdentifier>
              <References> … </References>
          </UniqueIdentifier>
       </DataRepository>
       <InternalLinks>
           <Link> ……..</Link>
       </InternalLinks>
   </DataRepositories>
   <PreferenceRepositories>
       <Repositories> … </Repositories>
        <InternalLinks>
            <Link>…….. </Link>
       </InternalLinks>
   </PreferenceRepositories>
   <CrossLinks> ……. </CrossLinks>
 </target>
```

In case of our simple example of privacy obligation policy, the above skeleton could be instantiated with the following information: (1) a *data repository* entry, containing the database and table names where personal data is stored, the table's "primary key" (e.g. UserId) and an alias (e.g. DataRepAlias) for this repository; (2) a *preference repository* entry, containing the database and table names where preferences are stored, the table's "primary key" name (e.g. PrefId) and a alias for this repository (e.g. PrefRepAlias). A field in this table, for example called TimePreference, could be used to store users' preferences about *deletion time* of Credit Card details; (3) a description (in the "Cross link" sub-section) of how to link personal data to preferences (e.g. DataRepAlias.UserId = PrefRepAlias.PrefId)

5.2. Metadata

The *Metadata* section *of a parametric obligation policy* describes: (1) Type of obligation policy (e.g. "Parametric"); (2) Natural language description of the obligation, presented to users and/or administrators. The XML skeleton of the metadata section follows:

```
 <metadata>
   <type>Parametric</type>
   <description>  … </description>
 </metadata>
```

5.3. Events

The *Events* section of a parametric obligation policy describes "parametric" events that must occur to trigger the obligation. These events can contain references to personal data and preferences described in the *Target* section. The high level XML skeleton of the *Events* section follows:

```
<events operator="AND/OR/NOT ">
   <event id="e1">
     <type> …</type>
    </event>
 </events>
```

One or more *event* or *events* sub-sections can be described in this section, in a recursive way, combined via logical AND/OR/NOT operators. Each "event" subsection has a unique, local identifier. The actual definition of these events depends on their types. Currently managed types of events are:

- **Time based event:** it describes a condition that checks the current time (NOW) against a stated time. The "stated time" can be retrieved via a reference (e.g. to a field in a Privacy Preferences data repository) ;
- **Data Access event:** it describes a condition on how many times a specified user's data item(s) can be accessed in a predefined period of time. The actual information (user's data item, number of accesses and period of time) can be retrieved via references to values stored somewhere;
- **Data Deletion event**: it describes a condition that is true when a specified piece of data has been deleted (by an external system). The location of this data can be specified via a reference.
- **Context-based event**: it describes conditions on contextual information (e.g. system attributes, OS or application-based information). References to this information can be used.

In our example of privacy obligation policy, a simple time-based event is described as follows:

```
<events operator=" ">
  <event id="e1">
   <type>TIMEOUT</type>
    <date">
        NOW > [#ref] PrefRepAlias.TimePreference
    </date>
  </event>
</events>
```

In our example, the "*NOW > [#ref] PrefRepAlias. TimePreference*" condition is verified if the current time (NOW) is greater then a time accessible via the "[#ref] PrefRepAlias.TimePreference" reference. This reference points to information stored in the Privacy Preferences repository (having the PrefRepAlias alias) in the "TimePreference" field, as declared in the Target (see the Target example in section 5.1). It is important to notice that, in our example, each piece of data has an associated preference value - specified by the user and stored in the Preference Repository ("TimePreference" field).

At this "declarative" stage, this *reference* is a "generic" *reference* to potentially many values stored in the Preference Repository. It must be *contextualized* to each specific "piece of data" the policy applies to. This happens at "runtime", during the interpretation of *events*. Our *scalable obligation management system* will achieve this by using the Target section of this policy: for each targeted piece of data it will retrieve the associated preferences based on the specified reference (e.g. "TimePreference" value in the Preference Repository) and check any related condition in the events section (in our simple example it is a simple time-based condition). This is done in an efficient way, via a few SQL queries to databases. In our example, when the time-based condition is satisfied for a given piece of data and an associated preference, the system triggers the enforcement of related actions (on that piece of data).

5.4. Actions

The *Actions* section of a parametric obligation policy describes "parametric" actions to be enforced when an obligation is triggered by its events. These actions can contain references to data and preferences consistently with the definitions in the Target section. A high level XML skeleton of the *Actions* section follows:

```
<actions>
   <action id="a1">
     <type> ……..</type>
     <onCondition>  … </onCondition>
       …
   </action>
 </actions>
```

One or more *action* sub-sections might be defined in this section. Each "action" sub-section has a unique, local identifier. Actions are executes in a sequence, potentially subject to the satisfaction of (optional) conditions (e.g. constraints on Privacy Preferences. By default these conditions are TRUE i.e. actions are just executed). The actual definition of these actions depends on their types. Currently managed types of actions are:

- **Notification Action:** this action sends a notification to an entity. The e-mail address of

this entity can actually be a reference to a value in the Data Repository;

- **Deletion Action:** this action deletes a piece of personal data or some of its attributes. A reference can be used to identify this piece of data;
- **Command Execution Action**: this action executes an external application or service (e.g. a workflow application to process a piece of data or transform it). References to personal data or privacy preferences can be passed as parameters;
- **Logging Action**: this actions logs information (including referenced information) for auditing purposes.

In our example of privacy obligation policy, two actions are defined, to delete user's credit card details and notify users:

```
<actions>
  <action id="a1">
    <type>DELETE</type>
     <data attr="part">
       <item>
          [#ref] DataRepAlias.CreditCardRef
       </item>
       <item>
          [#ref]DataRepAlias.CreditCardNumber
       </item>
     </data>
  </action>
      <action id="a2">
        <type>NOTIFY</type>
        <method>EMAIL</method>
         <to>  [#ref] DataRepAlias.Email </to>
         <text> some e-mail text here </text>
       </action>
</actions>
```

These actions contain references to personal data (credit card details and e-mail address). The same observations made in the "Events" section apply here. These references are "solved" at runtime, based on contextual information i.e. specific pieces of personal data for which obligations have been triggered.

5.5. On Violation Actions

The *"On Violation"* section of a parametric obligation policy describes "parametric" actions to be executed in case an enforced policy is violated i.e. if any of its enforced actions fail. The XML skeleton follows:

```
<onViolation>
   <ovAction id="ova1">
     <type> ……..</type>
     <onCondition>  … </onCondition>
      …
   </ovAction>
</onViolation>
```

An action can fail either at the enforcement time or afterwards (e.g. deleted data could reappear because of wrong database synchronisation): this latter case is detected by the monitoring component of our obligation management system. All actions described in the "Actions" section can be used in the "OnViolation" section. A specific "RE-ENFORCE" action has been introduced just for the "OnViolation" section: when used, it requires the system to re-enforce just the actions that have failed (in the Actions section).

6. Deployment of Parametric Obligation Policies

Parametric obligation policies must be deployed in an obligation management framework for their interpretation, enforcement and monitoring. Figure 2 provides a high level view of the key aspects involved in this process:

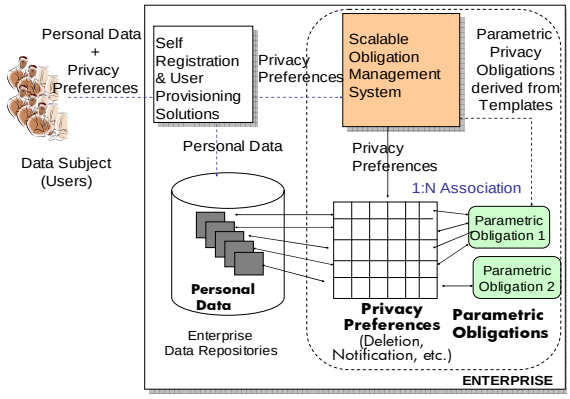

Figure 2. Deployment of Parametric Obligation Policies

Privacy administrators still need to interpret and refine privacy laws and guidelines and express them into obligation policies that can be managed within their enterprise realities. Administrators must also understand how personal data is collected and where it is stored within an enterprise IT infrastructure. They need to make decisions on which types of obligations an enterprise wants to enforce and which degree of customization (privacy preferences) provide to their users (customers, employees, etc.). Once this information is known, obligation policies can be expressed in an explicit format, programmatically interpreted and enforced.

An administrator can leverage our obligation management framework and related *scalable obligation management system* – also referred in this paper as SOMS system (see next section for more details) - to achieve this. In this context an administrator can use SOMS GUI capabilities to author

parametric obligation policies by describing all their sections (*Target*, *Events*, *Actions*, *On Violation actions*, etc.). Via these GUI tools, for each parametric obligation, information is collected about which privacy preferences are required, how they relate to the policy and how to present this policy to the end-user via a meaningful description.

A set of parametric obligation policies are then deployed in the SOMS system, to drive its privacy-aware information lifecycle management capabilities.

The SOMS system can be integrated with back-end Identity Management enterprise solutions, such as Self-Registration and Provisioning solutions, e.g. [9]. In this context when users self-register (i.e. provides their personal information) via an enterprise portal, they are also presented with a list of supported (parametric) obligation policies along with the required parameters. Users can make their choices, select (a sub-set of) obligations and provide their preferences. These solutions provision users' information (personal data) in enterprise data repositories. Thanks to adaptors, they will also provision the SOMS system with the list of selected parametric obligation policies and preferences. The SOMS system, based on the *Target* definition of selected parametric obligation polices, knows where to store related preferences and ensure that links to personal data are maintained.

A potentially large set of users and their personal data (>100K) can be provisioned to the enterprise. In this context, just a potentially small set of predefined parametric obligation policies is required to dictate all the criteria enabling privacy-aware information lifecycle management tasks. The SOMS system will manage them. There is no anymore need to instantiate an obligation policy for each provisioned data item: each predefined parametric obligation policy is dynamically associated to a set of managed data (that can change over time). This addresses the scalability requirement.

Next section provides more insight on the Scalable Obligation Management System (SOMS).

7. Scalable Obligation Management System

The SOMS system is an evolution of our previous version of the Obligation Management System (OMS) [5,6] to interpret, enforce and monitor *parametric obligation policies*. Figure 3 shows a high level architecture of the SOMS system. Its main components are:

- **Obligation GUI**: this is the graphical GUI used to: (a) author obligation policies; (b) check for their run-time status; (c) check the status of SOMS system components;
- **Obligation Server**: it is the core engine orchestrating interactions with other SOMS components. It interprets calls to SOMS APIs (1), stores privacy preferences in stated repositories (2), update associations between preferences and parametric obligation policies. Provides information to the Obligation Scheduler (3) to ensure that the SOMS system is aware of the need to manage obligations on new personal data, based on specified preferences;
- **Obligation Scheduler**: it is the component that checks if (parametric) events trigger any parametric obligation (5). It solves, at runtime, any reference contained in the *Events* section of obligations (4), based on the contextual personal data;
- **Event Manager**: it is the component that checks for incoming external events (time, access, context events, etc.) of relevance of SOMS, translate them in a meaningful internal format and transmit them to the Obligation Scheduler for further processing and correlation (4);
- **Obligation Enforcer**: it is the component that enforces the *Actions* part of triggered parametric obligations (6), by resolving, on-the-fly, related references, in the context of specific personal data and informs the Obligation Monitor (7);
- **Obligation Monitor**: it is the component that periodically checks the status of enforced obligation policies against the current status of data. Violations are reported and graphically visualized in the SOMS GUI. When specified in parametric obligations, this component will automatically try to remediate violations by executing the "On Violation" section of these policies (8).

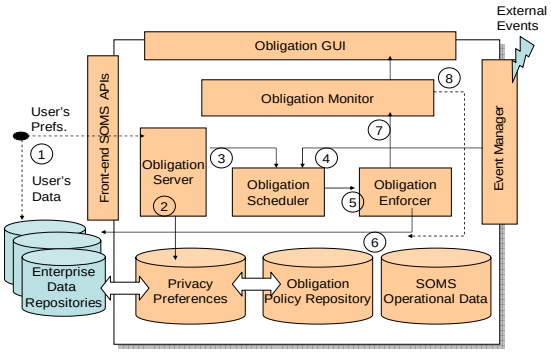

Figure 3. High-level SOMS System Architecture

The key innovation introduced in the SOMS system is its capability to dynamically interpret parametric obligation policies (i.e. their *Target*, *Events*, *Actions* and *OnViolation Actions* sections) and map their references on actual "targeted" data and preferences. This is done in an efficient way, via SQL queries that are instantiated on-the-fly – based on targeted data and related preferences. Figure 4 provides and high-level view of the related process implemented in the SOMS system, triggered by the occurrence of external events of relevance for a given parametric obligation policy.

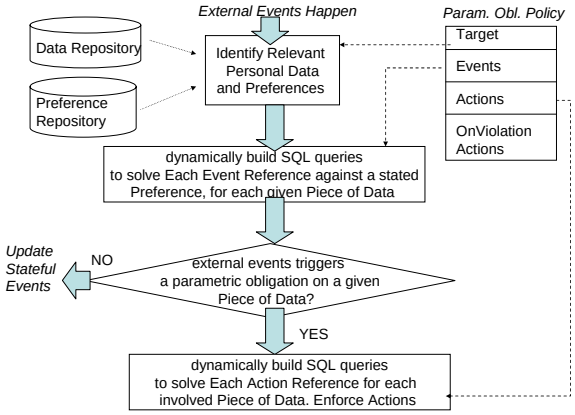

Figure 4. Reference Resolution Process

When external events happen for a given parametric obligation, the SOMS system identifies the targeted personal data and related preferences. Based on this context, a few SQL queries are dynamically built to solve any reference in the *Events* section and, at the same time, check their values against stated *Events* conditions. For each piece of data (targeted by this parametric obligation) where the "customized" *Events* section triggers the enforcement of *Actions*, the system will dynamically build SQL queries to solve references in the Actions section and enforces them.

It is important to notice that some of the *Events* and *Actions* defined in parametric obligation policies might be *stateful*. For example, a parametric "Access Event", that is triggered once targeted pieces of data are accessed more than X times, has to keep access counters specific for each piece of targeted data. The status of enforced *Actions* has to be stored for monitoring purposes, etc.

The SOMS system stores all this *metadata* associated to parametric obligation policies in its internal "*SOMS Operational Data*" repository.

Of course, this information can grow with the amount of managed personal data. However, it is just a matter of storage of simple data and efficient retrieval: this is done by using RDBMS databases and properly crafted queries (similar to the ones used to solve references).

The SOMS system extends our previous version of obligation management system [5,6]: it provides this new features in addition to the existing ones. As such the SOMS system can manage both parametric obligation policies and "traditional (non-parametric)" obligations. This allows an administrator to tune the system and take advantage of a hybrid obligation management approach [4] depending on: (1) the need for efficiency and scalability (hence using parametric obligations); (2) the need for flexible and ad-hoc definition of obligations on specific instances of data (hence the usage of "non-parametric" obligations).

A full working prototype of our SOMS system has been implemented and re-integrated with HP OpenView Select Identity solution [9], a state-of-the-art User Account and Provisioning solution for enterprises. This shows the feasibility of our approach in a real-world environment.

Initial results are very encouraging. Despite the fact that at this stage we cannot yet provide a quantitative analysis of SOMS performance, our prototype has been already tested with about 100K items of personal data – in a context where about 10 parametric obligation policies have been deployed (covering most common combination of event and action types). Each item of personal data was associated to specific privacy preferences.

The SOMS system (installed in a "standard" PC using MS Windows XP Professional, with data stored in MySQL databases) has gone through all the required steps in terms of event processing, action enforcement and monitoring - without noticeable problems.

We are currently performing additional tests on larger datasets and different types of parametric obligations and collecting information on the behavior of the system (future papers will provide this information). Future work includes further extensions of managed policies, performance tests and R&D in PRIME.

8. Related Work

As anticipated, this paper is a follow-up of [4] that described in details the scalability issues we encountered in our previous work and lessons learnt. Our previous papers [5,6] in the area of privacy obligation policies provide a detailed comparison of our vision and approach against related work. In a nutshell, existing approaches to privacy obligation policies (targeting personal data) such as EPAL [7] and XACML [8] subordinate obligation policies to access

control policies whilst our approach considers them as first-class entities, independent from access control aspects. In this paper we have already argued that this is required, if we want to enable a proper management and enforcement of obligation policies.

A relevant, formal definition of obligation policies and a related framework is described in [10]. However, also this work positions the concept of obligations in an authorization context.

We have not found relevant papers providing scalability analysis of obligation management systems, in the specific context of privacy management. Paper [11] describes an approach to obligation policies and a related solution that is very similar to our previous work on obligation management (affected by scalability issues). It looks like that their approach requires multiple instantiation of obligation policies on managed entities.

Relevant work on obligation policies (beyond the privacy realm) has been done in the Ponder Language and its management framework [12]. Obligation policies are expressed in terms of event conditions and actions, executed on targeted objects once events are triggered. These policies are deployed and used in a security and network management context, coupled with access control/authorization policies. The actual representation of policies and their enforcement framework is different from ours, as dependent of the deployment context: in our case the focus is on data and information – in theirs on network component and security properties. Further R&D work could be done to compare approaches and scalability properties of the two systems and explore opportunities for collaboration.

9. Conclusions

The management of privacy is very important for enterprises in order to deal with regulatory compliance and customer satisfactions aspects. In particular privacy obligations need to be managed. In this context solutions are required to automated privacy-aware information lifecycle management and reduce costs. Their scalability to large set of data is a key requirement.

This paper describes our recent work in this space (that keeps into account learnt lessons) based on the concept of parametric obligation policies and a scalable obligation management system and framework. A working prototype has been fully implemented and integrated with HP identity management solutions to show the feasibility of our approach in a real world domain. Initial tests demonstrate the scalability of our approach to handle obligation policies on large sets of data (about 100K). This is work in progress. Further research will be done in the context of PRIME and HP Labs.

10. References

[1] Rotemberg, M., Laurant, C.: Privacy International: Privacy and Human Rights 2004. Electronic Privacy Information Center (EPIC), Privacy International. http://www.privacyinternational.org/survey/phr2004/, 2004

[2] OECD: OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data. http://www1.oecd.org/publications/e-book/9302011E.PDF, 1980

[3] PRIME Project: Privacy and Identity Management for Europe, European RTD Integrated Project under the FP6/IST Programme, http://www.prime-project.eu/, 2006

[4] Casassa Mont, M: Towards Scalable Management of Privacy Obligations in Enterprises, TrustBus 2006, 2006

[5] Casassa Mont, M.: Dealing with Privacy Obligations: Important Aspects and Technical Approaches, TrustBus 2004, 2004

[6] Casassa Mont, M.: A System to Handle Privacy Obligations in Enterprises, HP Labs Technical Report, HPL-2005-180, 2005

[7] IBM: The Enterprise Privacy Authorization Language (EPAL), EPAL 1.2 specification. IBM, 2004

[8] OASIS: Extensible Access Control Markup Language (XACML) 2.0, 2005

[9] Hewlett-Packard (HP): HP OpenView Select Identity, http://www.openview.hp.com/products/slctid/index.htm, 2005

[10] Hilty, M., Basin. D., Pretschner, A.: On Obligations, ETH Zurich Switzerland, ESORICS 2005, 2005

[11] Gama, P., Ferreira, P.: Obligation Policies: An Enforcement Platform, POLICY 2005, 2005

[12] Damianou, N., Dulay, N. , Lupu, E. , Sloman, M.: The Ponder Policy Specification Language, 2001

[13] W3C: Resource Description Framework (RDF), http://www.w3.org/RDF/, 2004

[14] W3C: Semantic Web, http://www.w3.org/2001/sw/, 2006

[15] Casassa Mont, M: On Privacy-aware Information Lifecycle Management in Enterprises: Setting the Context, ISSE 2006, 2006