Whitepaper published April 2, 2019

# Enforcable Privacy Policy Whitepaper

PrivacyPortfolio's "Enforcable Privacy Policy" supports a fully-automated, negotiable exchange of policies between two or more parties.

Although our primary focus is on privacy policies, these electronic document interchanges can also work for other types of policies. [Design Goal #1]

The protocol for exchanging policies is not specified, but any transport protocol such as REST, SOAP, etc. could be used. [Design Goal #2]

We use schemas as the mechanism for representing and interpreting policies. These schemas can be in a variety of formats such as xml, rfd, json-ld, etc. and support a variety of markup vocabularies. Our privacy policy schema is a composite of other schemas and vocabularies, which support existing standards for specific industries, frameworks, organizations, or mandates. [Design Goal #3]

A policy is considered to be "enforcable" when policy statements can be interpreted as claims, governed by a set of rules, and programmatically tested by another organization or person. [Design Goal #4]

Policies should be discoverable, and have standard interfaces for conducting queries and matching claims between two or more policies to promote the programmatic transformation into negotiated, mutual agreements. [Design Goal #5]

Conditional rulesets within policies must support role-based identity contexts of each party a rule applies to. [Design Goal #6]

Each policy should contain all the information needed to support implementation services of discovering, accessing, validating, negotiating, and enforcing policies. [Design Goal #7]

## Building on previous efforts

The Platform for Privacy Preferences (P3P) was developed by the World Wide Web Consortium (W3C) as a specification and vocabulary that enables Web sites to communicate their data management practices in a standard format that can be retrieved automatically and interpreted easily by user agents in a machine-readable format. The P3P standard failed to catch on, mostly due to political and technical issues with proprietary browser implementations, which acted as the "mediator" between a publishers policy and the user's preferences. The P3P explicitly states it is not an enforcement mechanism.

In our model, we remove the dependency on proprietary browsers, and assign the "enforcement" responsibility to the intended audience of a policy. The intended audience could be individual persons or organizations, so we use the term "Subscriber" to represent the intended audience of a policy "Publisher" , who could also be an individual person or organization.

Negotiating policies between two parties is not new. The “TLS handshake” is only one example of real-time automated policy negotiations: server and client arrives at a mutual agreement based on their existing capabilities and preferences.

Our goal is to compare two policies and find matching rules we can transform into a mutual agreement. If both parties have a lot of conditions that must be satisfied, it makes sense economically, to pre-determine the chances of arriving at a mutual agreement. This evaluation also extends to “due diligence” testing: trying to estimate whether the other party has a reasonable capability to honor its end of the agreement.

PrivacyPortfolio’s approach is to level the playing field between “publisher” and “subscriber”, opening up more choice on how trust is established prior to entering into agreements, like so:

1) I trust the claims of the other party.
2) I trust independent verification of the other party’s claims.
3) I trust the other party will be punished if they fail to honor their claims.
4) I trust my own testing and validation of the other party’s claims.

Most privacy policies today are privacy statements.

Consider the difference between “we comply with all applicable laws”, and “we will notify you by email within 30 days of a suspected data breach”. Our goal is to elevate privacy statements into actionable data governance policies that can be executed manually by humans or automatically through services.

These policies define the data, assigns ownership of the data, and governs access and usage through permissions provided within the rulesets.

## Technical Design

In this whitepaper, we present the technical design of our Enforcable Privacy Policy.

Deliverables produced using this design includes:

1. PrivacyPortfolio’s Enforcable Privacy Policy (published on our website)

2. Machine-Readable Format of our Enforcable Privacy Policy (download from our website)

3. Our Privacy Policy schema (released on Github)

4. Our Privacy Policy XSLT stylesheet (released on Github)

5. Machine-Readable Format of Craig Erickson’s Personal Privacy Policy (available to subscribers)

6. Policy REST API (released on Github, self-hosted)

7. Test Results (published on our website)

**[Design Goal #1] Policy schema supports different policy types, and includes Common Policy Markup Language rulesets.**

To satisfy this goal we needed a schema for a privacy policy. Our search for existing schemas yielded only two: the P3P 1.0 and 1.1 schemas, and the W3C's Common Policy Markup Language. The Common Policy schema was originally designed for managing authorizations for privacy preferences in software applications, and consists of conditional rulesets. These rules can be used to evaluate any claim within a policy, but the Common Policy schema does not provide everything we need to represent an entire privacy policy.

Clearly, we would need to build a composite schema from a variety of other schemas. One of our principles is to accelerate acceptance by using existing standards whenever possible. We chose schema.org as our default standard for schemas representing commonly-used entities such as Organization, Person, Claim, Location, etc. Schema.org has the widest acceptance in terms of usage, and it also supports different schema formats, such as JSON-LD, RDF, xml, etc. We intend to contribute our Privacy Policy schema after completing the initial test and comment period.

The P3P schemas came closest to having elements that are specifically related to privacy, and in particular, attempted to resolve issues related to an entity having a multitude of distinct datasets associated with multiple role-contexts. The P3P schemas were only used for historical reference in the design of our Privacy Policy schema.

To demonstrate how our Policy Schema can be extended to other types of policies, we created a Cookie Policy document using the same underlying schema.

**[Design Goal #2] The exchange of policies should not be dependent upon any specific transport protocol or application (REST, SOAP, EDI, etc.)**

The only thing we did to satisfy this goal is to not think about it, except when developing our schema, which can generate APIs, WSDLs, and SQL stored procedures, when they are appropriately designed.

**[Design Goal #3] Support a variety of markup vocabularies as standards for specific industries, frameworks, organizations, or mandates.**

As one example, there is a vocabulary for representing "provenance" in a DataCatalog. We intend to use this as a reference pointer to authoritative documents that prove one's identity within a specific role context.

Another example involves "location data", which supports transforming street addresses, IP addresses, GPS coordinates, etc. into logical "geo-regions" to protect privacy.

HL7 is an acronymn for Health Level 7, a US standard for exchanging healthcare information. HL7 is a structured vocabulary used in Electronic Healthcare Records, and we utilize HL7 vocabularies to represent roles such as patient and provider, as well as other generic structured documents such as consent directives.

These different vocabularies can be utilized in an XML Schema by declaring distinct namespaces.

**[Design Goal #4] A policy is considered to be "enforcable" when policy statements can be interpreted as claims, governed by a set of rules, and programmatically tested by another organization or person.**

Successful implementation of this design goal implies that any policy can be disassembled into these elements: descriptions, definitions, statements, claims, rules, roles, datasets. Every claim must be associated with a rule that tests it by evaluating conditions in the ruleset, within the appropriate context and in compliance with a particular standard.

**[Design Goal #5] Policies should be discoverable, and have standard interfaces for conducting queries and matching claims between two or more policies to promote the programmatic transformation into negotiated, mutual agreements.**

We ensured our policy schema has the attributes required to support metadata requirements for cataloging datasets. These datasets could be a repository of policies, or could represent the metadata for datasets requested or returned by an interface, relevant to a role context of a person or organization. The need we anticipate is providing the capability to discover and query thousands of policies to increase the number of matches, which increases the efficiency of identifying eligible parties to form agreements with.

Another function of the Data Catalog is to support exchanging information via API with a Subscriber, as an alternative to manual data entry in webforms, etc. As an example, a DataSet for AccountInformation could contain every data element required to register or modify login account information for a DataProfile of type: "Customer". A DataSet for subscribing to a newsletter, could have only one data element: "email", for a "Subscription" DataProfile, or could be used in combination with a CPML RuleSet, governing the validity of the subscription, in which an expiration date or consentDirective may be required as conditions of validity.

This goes beyond comparing two policies to find matches – it also finds out what personal information is required in order to become a customer.

**[Design Goal #6] Conditional rulesets within policies must support role-based identity contexts of each party a rule applies to.**

"Role-based identity contexts" is a topic we refer to frequently in our design goals, because it is such a huge challenge of policy enforcement. The personal dataset returned by my role as "patient" is not the same dataset returned by my role as a "taxpayer". Even the same dataset can have different values for identical fields: my legal name on my birth certificate is different from my legal name on my driver's license. This point demonstrates why support is needed for master data services: managing the quality and integrity of data depends on who we share information with and what it is used for.

**[Design Goal #7] Each policy should contain all the information needed to support implementation services of discovering, accessing, validating, negotiating, and enforcing policies.**

One of the benefits of using structured schemas to represent policies is being able to generate APIs, WSDLs, object-oriented classes, or database tables that support implementation services.

These services can and should be used to test the claims within a policy. Some tests may not be executible without access and authorization to resources. The rulesets using Common Policy Markup Language contain everything required to specify and check permissions for conducting these tests.

We used the following methods to test our implementation of Design Goal #7:

(after authoring our own “Enforcable Privacy Policy” for PrivacyPortfolio)

We created an XSLT stylesheet for transforming the xml format of our policy into the html markup on our website to prove that all the data is sourced from the xml version. (It is considered poor form to include HTML markup tags in the machine-readable versions.)

I used a privacy policy from one of our clients to build an “Enforcable Privacy Policy”. I followed the same methodology and was able to manually tag all the elements within an hour. The client is an organization, preparing for the GDPR, so I also tagged relevant elements from the GDPR. In effect, this process is somewhat like policy negotiation: I take their old privacy policy, p1, compare it with the GDPR articles, p2, and add what is missing, or merge what matches.

One of the proposed services is using text-mining and NLP techniques to automatically harvest policies in the public domain (those posted on websites) and use statistical or machine-learning techniques to transform them into machine-readable formats.

We also compared one privacy policy of an organization, with one privacy policy of an individual.

This is when we discovered an issue: An entity, whether its an organization or a person, must have some personal data contained within it, to identify the role contexts of the parties. Ideally, we would rather have a separation of metadata and data, in which the policy defines the rules or process for governing data, and the data itself is securely stored in an isolated repository. (Our model also proposes a separation of identity access credentials, but this will be addressed in a subsequent whitepaper.)

These tests are in-progress, while we build our API services and seek comment from other privacy and cybersecurity practitioners. It may take a few more weeks of working with these policies and APIs before we can enforce them, which is why we set the effectiveDate on our privacy policy to April 25, 2019.