Personal Data Protection in the Semantic Web

Masters of Engineering Thesis

Ryan Lee

August 22, 2002

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical Engineering and Computer Science
and Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

Author _____

Department of Eletrical Engineering and Computer Science
August 22, 2002

Certified by _____

Ralph Swick
Thesis Supervisor

Accepted by _____

Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Personal Data Protection in the Semantic Web

Submitted to the Department of Electrical Engineering and Computer Science

August 22, 2002

In Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical Engineering and Computer Science
and Master of Engineering in Electrical Engineering and Computer Science

---

Abstract

Growing concerns over the abuse of personal information via the World Wide Web can be addressed at political, social, and technical levels. As the Web evolves into the Semantic Web, where machines understand the information they process, technical solutions such as PEDAL become feasible. PEDAL, the Personal Data Access Language, defines a vocabulary for composing policies that describe characteristics of clients who are allowed or denied access to the personal information a policy governs. Policies can be merged together using PEDAL negotiation rules. Semantic Web logic processors reason through policies, arriving at a final determination on information distribution for each request. Software for implementing PEDAL and test cases for exercising its features demonstrate basic PEDAL functionality.

Thesis Supervisor: Ralph Swick, swick@w3.org

Title: Research Associate, World Wide Web Consortium/MIT Laboratory of Computer Science

Acknowledgements

Funding for this work was provided by US Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-00-2-0593, "Semantic Web Development".

Thanks to Ralph Swick for his supervision, suggestions, criticisms, insight, and time.

To Tim Berners-Lee for initiating the World Wide Web, for carrying on a vision for something even better, and for inspiring a younger version of myself to eventually find my way to the W3C.

To everybody who's joined in with Tim inside and outside the W3C. Particularly to Sean Palmer for elucidating the myriad uses of *cwm*.

To my parents for being parents and doing the things parents do for kids like me. To my family as a whole for getting me through.

To my friends.

To Cze, for everything while this thing was being written.

Philippians 4:13. Soli Deo Gloria.

Contents

> "The hope [is] that when we've got all of our organization communicating together through this medium which is accessible to machines, to computer programs, that there will be some cool computer programs which we could write to analyze that stuff. And to do that, of course, the information on the Web would have to be understandable to some extent by a machine and *at the moment it's not.*"
>
> Tim Berners-Lee [1]

1 Introduction

The driving concern behind this research is the online privacy of personal data. Contact information, health status, financial activity, and browsing habits can all constitute personal information, and the extent to which it can be shared online should be under the control of the individual to whom it relates, depending on how much privacy they desire. The debate over online privacy is varied and heated, with several competing interests and agendas involved: commercial interests want to track browsing habits to improve marketing and profits; the U.S. Federal Trade Commission (FTC) recognizes the need to give people the power to protect their privacy online while noting that law enforcement needs to have access to private information to track down criminals [2]. Without aligning with any particular interest, this research is an academic and technical examination of how to prevent the misuse of personal information by protecting its online distribution.

In seeking to restore control over private information to the individual, one must look to the field of online access control. Access to computers has always been a part of computer history since users were billed for the computing resources they used and had to be identified through a username and password for record keeping. The method of usernames and passwords was kept as computers connected to each other in networks for remotely accessing a machine and for broader Internet applications such as FTP. In its earliest days, the Internet dealt primarily with communication

and sharing data. But as computer networking evolved, the number of users exploded, particularly due to the popularity of the World Wide Web, people involved with Web development began to recognize the need to provide finer grained access control over personally sensitive data. Early implementations of the Common Gateway Interface (CGI) and HTTP Authentication provided methods for differentiating between users, again with usernames and passwords. While these allowed individual servers to implement their own data protection schemes by somehow restricting access to certain users or groups of users, there was no way to universally express their criteria such that different software implementations of servers could enforce data protection. Even as the Web has evolved, no standard has emerged in declaring who has access to what.

With the Semantic Web under development, online privacy through access control is one serious issue that should be addressed. The Semantic Web offers an opportunity for access control policies to fully describe their access policies instead of resorting to usernames and groups, which rely on some pre-agreed contractual understanding that one individual or organization corresponds to some local username.

PEDAL, the Personal Data Access Language, offers an access control vocabulary for protecting personal information using the expressive power and reasoning available in the Semantic Web framework. Software applications Peddler and Bouncer implement the PEDAL vocabulary, demonstrating and testing its strengths and weaknesses.

Section 2 discusses the problem at hand in more depth. Section 3 refers to existing literature about proposed solutions. Section 4 provides background about technologies involved in building PEDAL, which is fully described in 5. Section 6 lays out the test cases used to illustrate PEDAL's features and shows the results of testing. Section 7 examines PEDAL's shortcomings and suggests future research possibilities; section 8 concludes.

See the appendixes of acronyms in section 9 and vocabulary terms in section 10 for the definitions of abbreviations technical terms used throughout this paper. Appendixes of code can be found in

sections 11 and 12.

2 Problem Statement

The fundamental issue PEDAL proposes to solve is a need to protect information and to protect it in such a way as to eliminate the need for prearranged rendezvous between one who has information and one who wants information. In the context of this research, information refers to published metadata, data which exists online to describe another resource.

There are several component issues to address behind the overall design of an information protection system in the Semantic Web. These include: a standard vocabulary for describing data access policies, a method for describing those who have access without relying on usernames, negotiation between data publishers, negotiation between clients and a data publisher, concealment, and extensibility. The W3C can often serve as its own test ground when implementing and trying out new ideas. Relevant examples for some portions of the problem description are given in the context of the W3C's services and needs to better illustrate the concern.

Standards prevent the proliferation of incompatible methods, given that a sizeable majority adheres to them. As the Semantic Web architecture evolves, the need for standards for some of its design components grows in importance. Personal information protection is one issue with a great deal of interest surrounding it. Since personal information may legitimately flow around the Semantic Web, enforcing its protection requires a standard which all, regardless of platform, can adhere to. The methods people use now to implement privacy protection are either propietary with no thought given to widespread distribution, insufficient for the Semantic Web, or altogether nonexistent. Anyone with a concern for maintaining data protection across the Semantic Web, and not just the W3C, may find a standard vocabulary for describing information protection to be of great importance.

Along with being a standard, the vocabulary should fit well into the Semantic Web's existing standards. Database research repeatedly runs into the problem of vocabularies describing a certain realm of information being biased towards the needs of the organization which originally developed the

vocabulary. Data models represent the same concepts with different words. Mapping between different data model terms becomes tedious without a common ontology to compare to. The Semantic Web, which strongly resembles a large network of databases, uses namespaces and statements of equivalence between terms inside those namespaces to try to overcome the common ontology problem. The W3C designs ontologies with this consideration in mind. A standard vocabulary which makes use of other ontologies through equivalence statements is of interest to anyone who wants to see the Semantic Web overcome ontology mapping issues.

A vocabulary in the Semantic Web should take advantage of its expressive ability to describe any resource. Data protection should be able to describe who has access to information based on people's characteristics and attributes instead of something abstract, like their username or group membership. Some sites don't even carry usernames or groups; protecting data is currently difficult to implement without some database of users to refer to. Scale the problem to one of global proportions, and it is clearly impossible for an organization with some username scheme to maintain a database of every person for their own use, nor would it even be desirable in most people's eyes for any such database to exist. Constructing an access policy based on actual characteristics instead of abstract naming allows for clearer policies and more widespread use.

Certain information may have different parties interested in creating policies surrounding its protection. Organizations need a corporate policy for data access. As a global organization composed of member corporations, the W3C often publishes documents intended for viewing only by members, or only for W3C employees. This is accomplished with a membership database and an internal access control application using the HTTP-Auth protocol. The W3C database is in fact based on evolving standards and is explained in detail in section 4.6, and is already moving towards the Semantic Web; however, this is not generally true of organizations on the Web.

Internal organizational groups may need their own group-wide policy. The W3C oversees a number of working groups that deal with specific interests concerning the Web. These groups are composed

of W3C employees and member organizations' employees. Their work is available to all member organizations, some to the public as a whole. However, some instances may arise where only group members would be allowed to see certain information, and the W3C database currently only addresses groupings in terms of membership organization employees or W3C employees. It can but does not use a working group perspective of its entries due to scaling issues.

Authors may need a document-specific policy. Contributors to standards drafts for the W3C may want to provide contact information to their fellow employees and some subset of that contact information to W3C employees, but not to the public. At the moment, the best way to accomplish this is to not provide any contact information within the document and hope that people already know how to reach the contributor.

With these separate parties trying to describe their own policies on the same data, some grounds for negotiation must be provided. It must be possible to generate an acceptable document policy to all parties involved since it is not necessarily feasible for all of them to meet and negotiate in person.

There is also a need for negotiation between people requesting data, the clients, and publishers of data. Applying a PEDAL policy to data results in a set of information required from the data requester to check if they match the criteria set forth in the policy. This is a request for personal information in exchange for personal information. Giving clients the ability to express rules over who has access to what information of theirs is a usability concern. It was not implemented during this research, though its design is discussed.

In discussing client personal information, a key question of authenticity of information must be considered. Clients can lie. This issue is not addressed in depth by PEDAL, though a fully working implementation of PEDAL must include solutions to the problem of authenticity. Suggestions for meeting and solving the problem are given in discussion on further research (section 7).

Sometimes a corporate policy must be kept private, for whatever reasons. The concealment of

policies such that they cannot be reverse engineered easily is an important component of PEDAL. However, without authentication, malicious users could repeatedly lie about whatever data is being asked for in order to discover some best guess recreation of the policy's rules. Brute force reverse engineering is another reason authentication of information and identities is necessary.

Lastly, a schema needs to be extensible such that future changes in the Semantic Web and the way people express personal data and their privacy preferences can be quickly adapted to.

3 Related Work

Many different parties are interested in providing solutions for the problem of online privacy. Some go through legislative channels, while those included here are technical in nature. Most of these live within the framework of the current World Wide Web, and several are biased towards the somewhat related goal of maintaining complete anonymity instead of protecting the diffusion of personal information.

3.1 Platform for Internet Content Selection (PICS) and PICSRules

PICS [3] was recommended by the W3C in 1996 as a language specification focusing on giving parents control over what kinds of content their children could access on the Web. The PICS platform had ratings services formulate ratings systems, much like the MPAA and its movie ratings. The content labels, like "level of violence" or "inappropriate language," comprising a ratings system formalized measurements along different dimensions so users could decide which ratings they were comfortable with for each type of label.

To supplement PICS, PICSRules [4] was recommended in 1997 as a language for users to specify which PICS rating services to use, where to obtain content labels, and what kinds of criteria about the labels' values made different Internet sites acceptable or unacceptable. Part of the motivation behind PICSRules was to ease the burden of creating profiles of rules for end-users.

There is a small degree of service negotiation in PICSRules in that users will either see or not see a site depending on their preferences and the ratings service used. This is useful for shielding children, but not for more complex client and server interactions. The PICSRules the language lacks expressiveness and uses formatting rules not completely consistent with today's standards.

PICS is mentioned here as a historical first attempt. While some still use it, P3P is the next generation in privacy protection.

3.2 Platform for Privacy Preferences Project (P3P) and APPEL

The W3C recently declared P3P a Recommendation [5], a project status that designates the work is approved and ready for widespread use. P3P is a standard for providing Internet users with notification of the use of their private information at Web sites they visit. Web sites provide their corporate privacy policy in P3P format, and users indicate whether or not the policy proposal is acceptable. Accepting a policy might then initiate the appropriate data being transferred to and from the site.

To further automate the process of policy exchange and user decisions, both a user information transfer mechanism and a policy negotiation module were originally included in the specification but later dropped due to worries about complexity and lack of public support [6]. In place of both components, the W3C proposed a draft of A P3P Preferences Exchange Language (APPEL) [7], a language for expressing preferences as a set of preference rules. APPEL is based on concepts taken from PICSRules. The W3C hopes to eventually supersede both PICSRules and APPEL rule sets with a more general rule language.

P3P may seem to address the same types of things PEDAL is intended to, namely personal privacy online, but its chief mechanism differs slightly from PEDAL's. In P3P, a service explains to its clients through a P3P policy what it intends to do with information it is given, giving clients the ability to moderate the flow of their personal information. PEDAL is a two way street. Clients can provide credentials depending on a service's credentials - this element is very much like P3P - and their supplied information determines what information they can obtain from a service. The main focus of this research is on the second portion.

P3P and PICS also differ from PEDAL in their feature sets. Neither PICSRules nor APPEL tries to address the issue of concealment, a design choice that may mitigate how much architecture can be taken from either system. P3P has been translated into RDF (see RDF section 4.2) [9] to insure

its compatibility with the Semantic Web, but a simple translation from XMLSchema to RDF does not take the expressive power of RDF into account. Furthermore, P3P and PICS do not provide techniques for merging rulesets.

Depending on where one stands on the political spectrum, PICS and P3P are either successes or major threats to civil rights. While there are interesting sociological debates over privacy standards on the Internet, their continuation is left to others. Though it attempts to contribute to those discussions, the W3C does not take any official positions regarding social issues stemming from technical ones, nor does this paper.

Sociological questions aside, both systems and their associated rule-based preferences languages should provide valuable groundwork from their design of client and service negotiation and expressing privacy policies.

3.3 Anonymizer, et al.

The nature of the underlying communications protocol used in the Internet, TCP/IP, allows computers to go through other computers called proxies while trying to establish a connection with a certain computer. The Anonymizer, found at <http://www.anonymizer.com/>, and similar services are all based on providing users a proxy through which to browse the Web anonymously. Since all requests handed to the proxy will appear to their end destination to be originating from the Anonymizer proxy, users cannot be tracked through normal IP address logging methods; additional actions by the proxy can prevent most other Web tracking methods, such as cookies. This requires a good deal of trust in the proxy service keeping its own tracking information private.

While anonymizing services provide a great deal of freedom to Internet users, particularly in shielding their browsing habits, they do not specifically address the problem of dealing with personal data. An 'anonymous' user who makes a purchase from an online vendor provides them with a good deal

of information, likely including their mailing address and credit card number. Information released to an unscrupulous vendor may very well be equated to releasing information to the entire Web. As Keith Little, a security activist, observed, "The Web has a long memory. Once exposed, personal information may often be, for practical purposes, exposed forever." [10] With no policies and one sole defense against determining information derivable from an IP address, this does not do enough to protect users. Anonymization is useful, but does not completely solve the problem at hand.

3.4 Freenet, et al.

Some believe the Internet is flawed and the current architecture cannot successfully protect user privacy. The goal of projects such as Freenet [11], Publius [12], and Free Haven [13] is to spin off a separate network either through a new protocol or a new Internet architecture. Many disparate groups are doing research in designing protocols, though widespread use of any of them has not caught on.

As Ian Clarke, author of the paper inspiring Freenet, writes, "It is actually the case that the Internet could lead to our lives being monitored, and our thinking manipulated and corrupted to a degree that would go beyond the wildest imaginings of Orwell," and also, "There is an indelible electronic trail from the information to those that are responsible for it." [11] Clarke makes the point that the Internet as is provides too much personal information by design. An improvement would be to guarantee anonymity by incorporating it at the lowest level of a new design he calls Adaptive Network, preventing the potential exploration of as-yet uncharted grounds of information exploitation.

Freenet and its brethren make an interesting case for a new information retrieval system rivaling the current World Wide Web. Perhaps their most difficult hurdle will not be a technological one but a societal one, convincing the entrenched Web user base that something new is necessary (though Clarke argues in his paper that integration is possible between an Adaptive Network and the Web). The Semantic Web and PEDAL hold an advantage in that they do not rely on the underlying

communication protocols to function, so a change in distribution methods would not present any necessary reasons to modify them; indeed, Freenet's inherent use of caches allows more than one location to host a document, which would make it imperative for a document's privacy protection policy to mimic its movements.

4 Included Technologies

PEDAL integrates and builds upon a large body of work researched by the W3C. XML, RDF, N3, cwm and log, and ChACL all play some part in the formulation of PEDAL. Understanding the function of each on its own is integral to understanding how they come together in PEDAL. The following sections provide a brief summary, some providing more extended explanations if the subject is used broadly enough. Further reading references are given.

4.1 Extensible Markup Language (XML)

The W3C's Extensible Markup Language (XML) [14] is a convention for describing structured data documents. A Document Type Definition (DTD) or an XMLSchema file provides the actual definition for a given type of document. Documents refer to a DTD or Schema so readers of the document will know where each markup tag is defined. XML is intended to improve machine understanding of information; markup from a vocabulary carries meaning an XML-based application can "understand."

XML was chosen for RDF (see section below) because it is the most widely used and documented markup language for its given purpose. It is particularly well suited for exchanging messages across networks - if participating members of a network possess the same DTD or Schema for their messaging needs, or can at least access the document type vocabulary, then XML messages can be understood by all involved parties.

Due to the modular nature of XML, documents may also refer to multiple vocabularies, but the problem of separate vocabularies using the same name for non-equivalent concepts may arise. For example, one vocabulary about computers may contain the term 'keyboard' while another about musical instruments may also contain the term 'keyboard' - the same word, but a different concept. To overcome this problem, universally recognizable names are given by way of a mechanism called

namespaces. An XML namespace is "a collection of names, identified by a URI reference [15], which are used in XML documents." [16] The collection mentioned is often referred to in this paper as a vocabulary, grammar, schema, or ontology. The names can be used within a document without conflict if they are properly referred to by use of the namespace URI. A namespace can be assigned an abbreviation local to the document to make including markup from an external vocabulary easier to read and write; these are called qualified names. The namespace abbreviation is separated from the term's name by a colon. The qualified name syntax is used frequently throughout PEDAL; further reading from [16] may be beneficial if the concept is unclear.

Because XML is case sensitive, it is important to note that the occasional differences in capitalization in XML or N3 examples throughout the rest of this paper do actually indicate different concepts.

In the context of this research, XML is a stepping-stone; it is the underlying format of RDF, which is the main vocabulary from which PEDAL stems. Understanding XML may be helpful [14] in understanding PEDAL but is not essential.

4.2 Resource Description Framework (RDF)

If XML is the way to represent a document, then the Resource Description Framework (RDF) [17] is the way to describe the document's contents. RDF is billed as the foundation for the Semantic Web, a vocabulary in XML format designed for resource description and metadata applications. A resource is anything that can be described by a URI; a Uniform Resource Identifier, in turn, is "a compact string of characters for identifying an abstract or physical resource." [15] Resources on the Web usually refer to the location of a web page, or a Uniform Resource Locator (URL), a subset of URI. Phone numbers, images, services, and people could all be considered resources.

The root of RDF's development is the need to express metadata, or data about data. In the Web, this was first attempted by defining new tags for HTML. This was difficult to propagate and relied

on users to include metadata of their own initiative and accord. The next step was the Platform for Internet Content Selection (PICS), which was essentially a scheme for filtering out content based on ratings, an attempt to allow parents to protect their children from unwanted sites. While PICS allowed others to describe resources not under their own control, PICS never really took off.

With the advent of XML, the metadata initiative grew into RDF. "The broad goal of RDF is to define a mechanism for describing resources that makes no assumptions about a particular application domain, nor defines (a priori) the semantics of any application domain. The definition of the mechanism should be domain neutral, yet the mechanism should be suitable for describing information about any domain." [18]

RDF achieves the ability to "say anything about anything" through its structure. The way data is represented strongly resembles several different paradigms. In one sense, RDF provides values for attributes of resources, much like attribute-value pairs. In another sense, RDF represents relationships between resources, much like ER diagrams. RDF can be visualized in a node and arc diagram, and some examples in this paper are presented in text and visual format for greater comprehension.

The data model consists of resources, properties, and statements. Anything being described in RDF is a resource, as detailed above. Properties are some specific characteristic of a resource or a relation between resources. Statements bring resources and properties together; they are composed of a subject, which must be a resource, a predicate (also verb), which must be a property, and an object that can either be a resource or text.

The XML RDF syntax is normative but sometimes difficult to comprehend, as XML is more a machine format, not a human one. Instead, examples using RDF are given in N3, which provides a more compact syntax similar to the English language. To find out more about RDF (in its XML syntax), see [18] for the specification or [19] for RDF Model Theory. Otherwise, N3 (section 4.3) covers the nature of RDF more deeply.

4.3 Notation 3 (N3)

Since RDF statements describe resources in the form of *subject predicate object*, also known as a *triple*, an alternate syntax to RDF that more closely follows the triple form was created. This syntax, called Notation3, [20] or N3, looks more like English grammar. It is possible to losslessly convert between RDF and N3. N3 does not have an official status within the W3C, but it is used frequently by Semantic Web developers as a syntactically simpler and shorter representation of information in place of RDF/XML and is often used as an alternate input to RDF/XML processors.

Understanding the basics of N3 is fairly simple because of its similarity to the English language; for a deeper discussion, Tim Berners-Lee wrote an N3 Primer [21]. A statement in N3 is composed of a subject, predicate, and object, ending with a period. A subject may have multiple predicates, so the semicolon is used to separate each individual predicate-object clause. A predicate may have multiple objects; the comma is used to separate each object in a list.

An identifier represents each of the elements in a statement. The identifier can be some combination of alphanumeric characters, which should be a reminder that the identifier is merely some unique way of representing a concept within a namespace and not necessarily a descriptive name from which information about the element being identified could be inferred. Each identifier is prefixed with the namespace it comes from with a colon separating namespace from identifier (discussed further in 4.4). The prefixes are bound to a full URI using the `@prefix` syntax seen in the grammar appendices, section 11. The colon on its own is generally used to represent the most commonly used namespace in a document. The examples below use the lone colon because of its syntactic necessity; the reader can imagine a namespace to fit the picture.

The object of a statement may be a string literal instead of an identifier, represented by quotation marks. These are strict rules, though there are exceptions due to a layer of syntactic sugar for ease of writing N3. It is conventional to capitalize the first letter of identifiers that are subjects and

objects, and to write verbs with a lowercase first letter.

```
:Jane :daughterOf :John, :Jennifer ; a :Female .
```

The above example illustrates a number of basic concepts. The subject is `:Jane`, with two verbs `:daughterOf` and `a`. `:daughterOf` has two objects, `:John` and `:Jennifer` while `a` has `:Female`. Note the uses of semicolons and commas as well as the concluding period. A graphical representation (produced automaically by an RDF visualization tool) can be seen in 1. Nodes represent classes, and arcs represent properties.

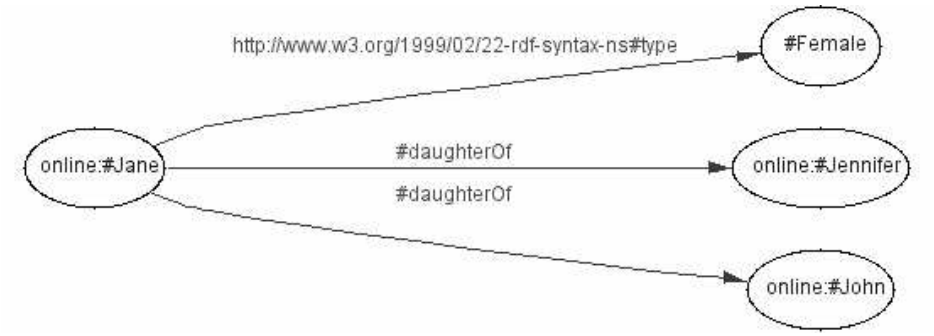

Figure 1: Node and arc representation of N3 example

N3 uses a few reserved words and symbols. One is the verb `a` as seen above, which indicates the type of object the subject of a clause is; this is shorthand for `rdf:type` (see RDFS, section 4.4). Thus in the example, the object `:Jane` is a type of `:Female`. Another is the verb `=`, which indicates equivalence between objects, shorthand for `log:equalTo` (see log, section 4.5).

Sometimes statements can be made about objects that do not need to be referred to, or are only referred to once, and go unnamed. These are called *anonymous nodes* and can be represented with square brackets. One way to describe a child with red hair:

```
[ a :Child ; :hairColor ‘‘Red’’ ] .
```

Statements can be made about other statements as well. Curly brackets are used to mark statements

to be used as subjects or objects, such as:

```
{ :Child a :Daughter } :statedBy :Father .
```

Here, the `:Father` object interacts with a triple. Using triples as subjects or objects is termed *reification*.

Examples and grammars are written in N3 throughout this paper.

4.4 RDF Schema

The above examples refer to `:Child`, `:Father`, and similar family-oriented terms. These RDF terms can only make sense in the context of some vocabulary. RDF Schema (RDFS) [22] is used to create these vocabularies, providing resource description for writing RDF files. Much like XML's DTD's provide a definition for a certain document type, RDFS provides an ontology from which RDF documents can be written. If someone chose to create a schema about families and relations between family members, they might use RDFS to define exactly what is meant by `:Child` and `:Father`. RDFS itself has an RDF Schema at <http://www.w3.org/2000/01/rdf-schema>. These schemas are usually the namespaces bound to prefixes in an N3 file. For exampe, a vocabulary using RDFS might use `@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema>`.

The terms in a schema are defined by the RDF classes and properties of RDFS. RDFS builds on the foundation of RDF, using the concepts of resources and properties to define a vocabulary that is in turn used to define other vocabularies. The important resources from RDFS as used in PEDAL are `rdfs:Resource`, `rdfs:Class`, and `rdf:Property`. The main properties are `rdf:type`, `rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:domain`, and `rdfs:range`.

Everything (except plain text) is a resource in RDF, so everything belongs to the `rdfs:Resource` class. A class can also be thought of as a type or a category. Other classes are identified as

`rdfs:Class`es. The `rdf:Property` represents the concept of an RDF property as discussed earlier in 4.2

When a resource belongs to a class, it has an `rdf:type` of that class. If a resource is instead a specialization of a class, like daughter is a specialization of child, then it is a `rdfs:subClassOf` the superclass. The same type of reasoning applies to `rdfs:subPropertyOf`. Properties can additionally define what makes sense as a subject and an object of the property. `rdfs:range` specifies which classes can be objects, `rdfs:domain` specifies which classes can be subjects.

Using these basic concepts, the following is a partial, basic schema for a nuclear family; a graphical version can be seen in figure 2:

```
:Parent a rdfs:Class .
:Father a rdfs:Class ;
    rdfs:subClassOf :Parent .
:Mother a rdfs:Class ;
    rdfs:subClassOf :Parent .
:Child a rdfs:Class .
:Daugther a rdfs:Class ;
    rdfs:subClassOf :Child .
:Son a rdfs:Class ;
    rdfs:subClassOf :Child .
:parentOf a rdf:Property ;
    rdfs:domain :Parent ;
    rdfs:range :Child .
:fatherOf a rdf:Property ;
    rdfs:subPropertyOf :parentOf ;
    rdfs:domain :Father ;
    rdfs:range :Child .
:motherOf a rdf:Property ;
    rdfs:subPropertyOf :parentOf ;
    rdfs:domain :Mother ;
    rdfs:range :Child .
:childOf a rdf:Property ;
    rdfs:domain :Child ;
    rdfs:range :Parent .
```

PEDAL and other vocabularies are written in RDFS.

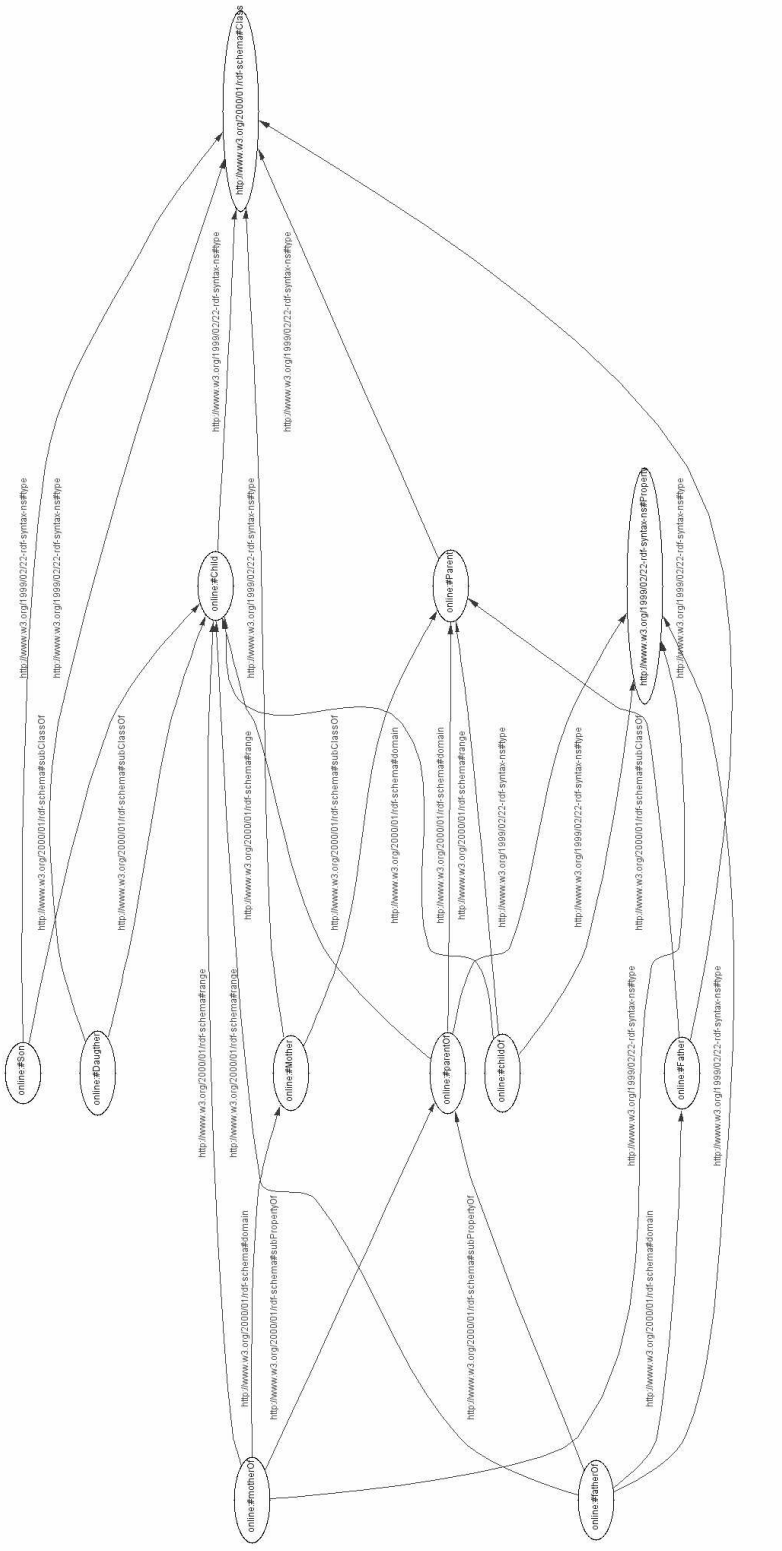

Figure 2: Node and arc representation of family schema example

4.5 cwm and log

N3 is tied closely to Tim Berners-Lee's logic namespace. In particular, it is humanly easier to write logical formulas in the English grammar-like N3 than in RDF/XML. Berners-Lee and Dan Connolly also wrote *cwm*, an RDF/XML, N3, and rule processor. *cwm* natively understands the logic namespace and is one of the main tools being used for reasoning through the Semantic Web.

The logic namespace (log) can be found at <http://www.w3.org/2000/10/swap/log.n3> in N3. The vocabulary defined in log is used to write rules that a rule processor can parse and draw conclusions from. For example, the English sentence "If John is Joe's father and Joe is Jane's father, then John is Jane's grandfather" could be written using log in N3 as such:

```
{{ :John :fatherOf :Joe .
   :Joe :fatherOf :Jane. } log:implies
 { :John :grandFatherOf :Jane . }} a log:Truth .
```

On a side note, `log:implies` and `log:Truth` are qualified names, as detailed in section 4.1. The key `rdf:Property` in the logic namespace is `log:implies`. A rule processor can conclude that the object of a statement is true if the predicate is `log:implies` and the subject is somehow known to be true. In this example, including the first couple statements about `:John` and `:Joe` somewhere in the same file as this rule would produce the third about `:John` and `:Jane` when *cwm*'s logic engine is run on it. The `log:Truth` object tells the rule processor the statement is true so it can try to make inferences based on the content of the statement.

Conversely, the `log:Chaff` object (not used in this example) tells the rule processor a statement is of no interest; purging a set of statements in *cwm* eliminates anything marked as being of type `log:Chaff` and anything referring to it.

The above example is only useful for specific, local identities identified as `:John`, `:Joe`, and `:Jane`.

To be more useful, the rules can be written more generally to match placeholder identities:

```
{{ :x :fatherOf :y .
   :y :fatherOf :z . } log:implies
 { :x :grandFatherOf :z . }} a log:Truth ;
   log:forAll :x, :y, :z .
```

Now, any statements that match the first two patterns will conclude the third, replacing `:x` and `:z` with whatever match was found. Multiple matches can be made. Depending on how the `cwm` command is called, rules can be run through once or until no more conclusions can be made.

This last, general form of rules is used in PEDAL. All such rules were passed through *cwm*, though other tools that understand log exist. The W3C publishes a suite of tools in Perl used in its own access control system at <http://www.w3.org/1999/02/26-modules/> (see also 4.6), and Jos De Roo of AGFA is developing a tool based on Euler Paths at <http://www.agfa.com/w3c/euler>.

4.6 ChACL

To continue experimenting with developing the Semantic Web, the W3C implemented ChACL, an RDF-based access control scheme for use on its own site. Designed and written by Eric Prud'hommeaux, ChACL uses an RDF Schema located at <http://www.w3.org/2001/02/acls/ns>. Prud'hommeaux also wrote [24] to document his work.

The ChACL RDF Schema provides identity management, group association, and privilege management. The `:Identity` superclass includes individuals and groups, denoting entities that may be granted access to some resource. Members of the `:Principle` class can be assigned some defining characteristics and also be collected together into members of the `:Group` class.

Members of the `:ResourceAccessRules` class are used to explain which `:Identity` can `:access` a resource. Using these key classes and properties, ChACL can reason through HTTP requests to

prove if a user agent can access the requested resource. ChACL uses the HTTP-Authentication protocol to determine the user's identity. Their relevant information is kept in the W3C member database. Another database contains the rules concerning a resource that form its basis for access reasoning.

PEDAL reuses ChACL's RDF Schema to handle users and groups.

4.7 Python

Python <http://www.python.org/> is used for writing all of PEDAL's associated software. This is fully because the choice of tool, *cwm*, and its associated libraries are written in Python. Documentation can be found at <http://www.python.org/doc/>. The basics of the language are very similar to most imperative languages, with some object-oriented principles mixed in. Perhaps its most unique characteristic is a reliance upon whitespace for demarcating blocks of code. Appendixes of PEDAL code are in section 12.

5 Personal Data Access Language (PEDAL)

5.1 Overview

PEDAL allows users of the Semantic Web to declare who has access to their personal information, whether their information is published online or stored on their own computer. There is no need for any prior arrangement to assign identifiers or usernames to people, bypassing the need for access control list methods in place today while still including the ability to make use of them. Instead, PEDAL-aware servers reason out who has access to what based on information they are willing to disclose. Since resources are not always under the control of one individual, the necessity for negotiating an acceptable policy for all parties involved is built into the vocabulary. Clients can also negotiate with servers over personal information being used in access control. The semantics of most of the PEDAL vocabulary features are expressed in logic rules. The rules are wrapped into the implementation of two pieces of PEDAL software, Peddler and Bouncer.

5.2 PEDAL Vocabulary

The full PEDAL vocabulary definition can be found in section 11.1. The namespace is <http://www.w3.org/2002/01/pedal/pedal#>. A graphical representation of PEDAL's schema can be found in figure 3. The text of the image is unreadable, but the overall layout is actually fairly simple; schemas tend to have "tall" graphs because they use the same RDFS properties repeatedly. To see the image in full size, enter PEDAL's namespace into the RDF Validator, found at <http://www.w3.org/RDF/Validator/>.

While the N3 version is provided in the appendix, the RDF version, without rules, is presented online[1]. However, the rules should not really be considered a part of the PEDAL namespace. They

[1] *cwm* does not properly translate N3 rules to RDF rules at this time

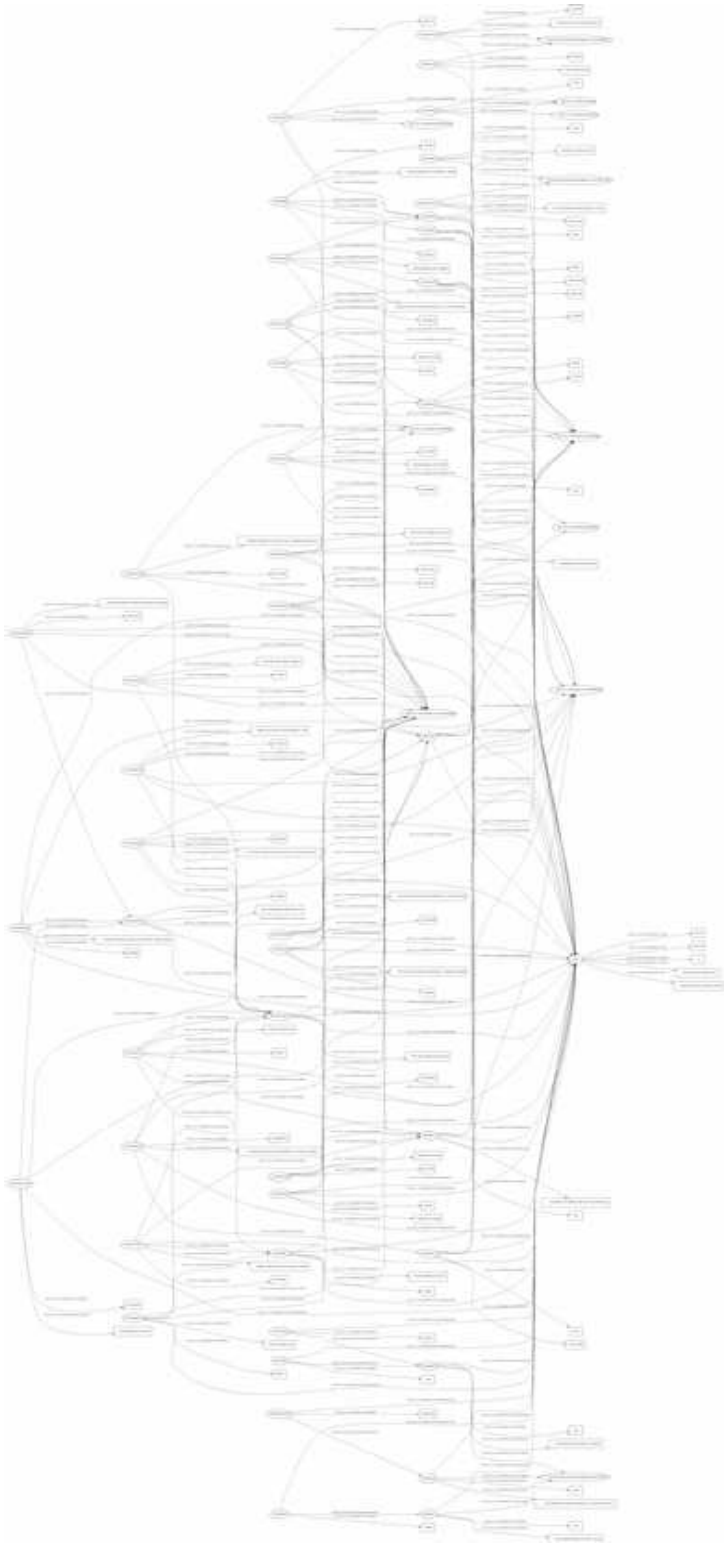

Figure 3: Node and arc representation of PEDAL schema

may need to change if others choose to extend their PEDAL usage, though the intended semantics of each portion of the vocabulary should be preserved.

The technical details of PEDAL will be thoroughly explored in the follow subsections, but a higher level snapshot in plain language may be useful in better understanding the minutiae. Some have said, in effect, the power of the Semantic Web lies in the ability to describe anything about anything. PEDAL documents utilize this power within a narrower paradigm of policies, metadata, and people. The policies describe the people and the metadata. What they describe for metadata is essentially information about that metadata; more specifically, access to metadata. Policies also describe people who can have access to the metadata; when someone sends a request for metadata, their credentials are first checked against the policy. They may get some metadata, all of the metadata, or none of it. Figure 4 graphically represents PEDAL's intended use

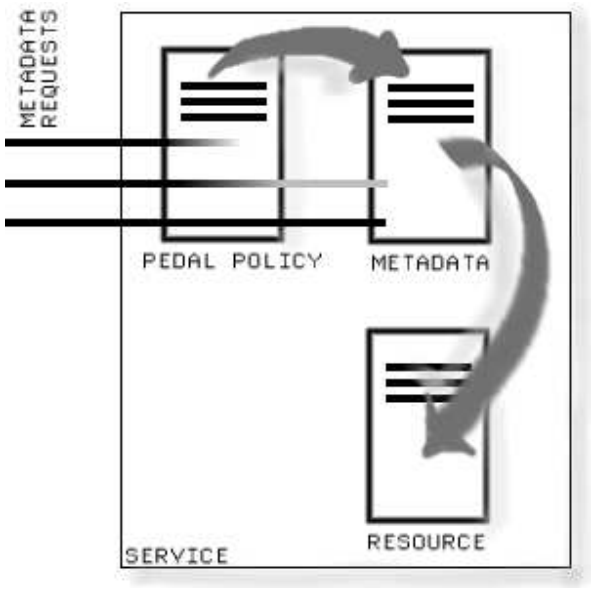

Figure 4: PEDAL flow

The PEDAL vocabulary is composed of terms that declare policies. They are used to describe metadata, to describe people, and, in private interactions, to describe negotiation values.

5.2.1 PEDAL Core

Part of the motivation of the Semantic Web is to provide common ontologies for disparate parties to agree upon and use in sharing information. PEDAL references ChACL (section 4.6), P3P (section 3.2), Contact (section 6.3.1), and FOAF (section 6.3.2) while defining its own core ontology.

Part of the PEDAL vocabulary provides for describing entities, meaning individuals or groups, for a Semantic Web user to use in writing a policy about who can see what information. This in particular is where PEDAL references other ontologies - others have already set about defining personal contact information (Contact) or general organizational information (P3P), it is to PEDAL's and the Semantic Web as a whole's advantage to reference their work instead of redefining the same concepts. Another portion of PEDAL provides the terms for writing basic access rules and for establishing the importance of those rules when negotiating with another party's policy. These portions are generally the whole of the PEDAL schema.

Since they are considered part of the core ontology, developers must fully include core portions in implementations if PEDAL is to be useful. They may be implemented differently internally, but interactions with each implementation should achieve the same desired effect.

Anything below prefixed with a colon is assumed to be in the PEDAL namespace unless otherwise specified by a prefix binding. Identifiers were given that do translate directly to the actual name of the concept (i.e., `:Policy` is really named "Policy" in the schema) to ease readability, but it should be noted that the identity to name mapping is not a generally safe one to assume.

5.2.2 PEDAL Policies

The overall unit of PEDAL is the policy. A PEDAL `:Policy` is composed of `:PolicyStatement`s. Each statement describes access rules for resources. Policies are not normally meant to be published. Instead, they guard published data. PEDAL policies are somewhat unique in that they interact

directly with published information on the Semantic Web but are never really seen with full disclosure by many more people than their authors.

Policy statements are concerned with one rule about one or more resources. A rule can have multiple components with an implied logical `and` connecting them. The components of a rule describe who is allowed to view that resource.

5.2.3 PEDAL Service Configuration

While providing a vocabulary for access rules is the main body of the PEDAL grammar, some service policy configurations are also included. PEDAL policies allow a policy writer to describe the type of user who can access information, though in a limited fashion. To widen the vocabulary of description, a service policy can declare that it supports a certain schema for both the author's and service's use in policies.

The `:supports` predicate indicates which namespaces a service can recognize in describing an entity. These namespaces are given the type `:Grammar`, partly so a more user-friendly abbreviation can be given using `:abbreviatedAs` to describe a `:Grammar`. These abbreviations and namespaces should also be used as namespace bindings for RDF/N3 headers in policy documents.

Either the author of a document (`:Author`) or the service (`:PolicyAuthority`) composes a policy. To note which, the `:authoredBy` predicate refers to the appropriate policy composer. The danger of allowing document authors to declare themselves as service policy composers is well-recognized and discussed in section 5.2.9. A policy can also list contributors to policy creation using `:contributedBy`, though this is normally a product of negotiation merged policies.

The policy states which resource's metadata it protects with `:forMetadataOf`. Policies can be given more meaningful names as well by supplying a string to the `:policyName` verb. Using the terms described so far, a policy might start out with:

```
@prefix :     <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix dc:   <http://www.purl.org/dc/elements/1.1/> .
@prefix here: <#> .

here:ExamplePolicy a :Policy ;
    :policyName ‘‘Example Policy’’ ;
    :authoredBy :PolicyAuthority ;
    :forMetadataOf <http://www.example.org/proposal> ;
    :supports dc: .

dc: a :Grammar ;
    :abbreviatedAs ‘‘dc’’ .
```

5.2.4 PEDAL Service Policy

PEDAL provides ways to write access rules for resources residing on a server. The following predicates and classes are a core of terms used to compose rules. A few terms are related to describing clients from a scheme of existing usernames and groups. Rules which go outside of an existing user scheme to describe clients' characteristics make use of ontologies recognized under `:supports` clauses.

Policy statements make rules. A logic processor using PEDAL's rules, described in section 5.2.8, transforms key parts of policy statements into parts of rules. A policy statement is made up of at least one instance of each of six properties. The objects of these six properties are used in rule generation.

This particular format of policy statements works better than having users directly writing access rules themselves. *cwm* makes logical reasoning relatively simple to understand and construct, but when it comes to comparing rules, *cwm* directives would become very opaque. The need for comparison, detailed in section 5.2.7, and the ease with which rules are generated from policy statements makes this the method of choice for expressing PEDAL policies.

One of the six properties a `:PolicyStatement` should include is `:forPolicy`, which refers to the policy the statement relates to. This becomes useful later in rules. `:forResource` indicates one or

more resources a statement is concerned with protecting.

Access is governed by a binary set, the `:visibleTo` and `:hiddenTo` `:visibility` properties. The rule being formed from a policy statement would use these in its conclusion to declare whether a resource is intended to be visible or hidden to the requester. In the end, the conclusion about a resource's visibility actually takes place, and a client is allowed to view whatever resources were declared visible to them. The `:visibility` superproperty simply makes it easier to refer to both at once. A policy statement will declare its sole `:withVisibility` to be either `:visibleTo` or `:hiddenTo`.

A policy statement may have one or more `:Component`s, referred to by the `:hasComponent` predicate. Where `:forResource` and `:withVisibility` specify the conclusion a rule could make, the elements of a component specify the requirements a client must fulfill to trigger its conclusion. In English, one might say that "a client can (or cannot) see a resource because their characteristics match up with a certain range of values."

A `:Component` mirrors the rules in a `:ruleSubject` using two properties, `:withPredicate` and `:withRange`. A client whose characteristic is the object of `:withPredicate`, and whose characteristic has the value of the object of `:withRange` would satisfy one component rule. Multiple components must all be satisfied (a logical *and*) for the rule to apply.

These rules are summarized in the `:ruleSubject`, which is an unfortunate product of *cwm*'s limitations. The information in rule subjects is essentially redundant with a statement's components; they contain the same information, just expressed differently. It should be possible to use the information in `:Component` classes to generate a rule's subject, but *cwm* is not quite finished when it comes to reasoning about lists. The utility of the `:ruleSubject` is described later in section 5.2.8.

Lastly, a policy statement's weight in negotiations is stated using the `:hasPriority` predicate, described in the section on negotiation, 5.2.7.

Adding on to the example policy:

```
[ a :PolicyStatement ;
    :forPolicy here:ExamplePolicy ;
    :forResource dc:title ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone :hasRole :Anonymous  } ;
    :hasComponent [ a :Component ;
                    :withPredicate :hasRole ;
                    :withRange :Anonymous ; ] ;
    :hasPriority :Must . ] .
```

The `:someone` resource may look out of place in this statement, but its use is described in 5.2.8; it is essentially a placeholding variable.

This statement could stand as a complete policy in combination with the initial policy presented in 5.2.3. More statements are usually provided. Since policy statements aren't referred to by anything else, they can be established as anonymous nodes. The same goes for components.

An empty policy automatically hides everything. This is consistent with lessons from the field: be explicit, and be paranoid. By assuming anything unmentioned should be protected, PEDAL guards against the chance of letting important information slip when it isn't intended to. Only when a resource is explicitly mentioned in a policy statement as being visible should it be revealed.

5.2.5 PEDAL Characteristics

PEDAL provides a small set of characteristics to use in describing clients. It takes its cues on `:Identity`, `:hasUsername`, `:Group`, and `:memberOf` directly from ChACL (section 4.6). These are used to reference an existing set of usernames and groups local to the service. An author who uses these classes and predicates should be well aware of what a service's username and group set looks like, generally not an unreasonable assumption. Using these terms, one possible component within a statement might look like:

```
:hasComponent [ a :Component ;
                :withPredicate :memberOf ;
                :withRange ‘‘Team’’ ; ] ;
```

Users could also have `:attended` a `:Meeting` - again, a class local to meetings the service is aware of. Likewise, users can be described as `:employedAs` an `:Occupation`. Lastly, users can hold `:Role`s. Roles are described later in the negotiation section 5.2.7, but one important case is the role `:Anonymous`. Everybody in the world is considered to hold this role, so a rule that specifies clients with `:hasRole` Anonymous is really saying something about everybody.

The example policy essentially states that everybody can see the document's value for its title.

5.2.6 PEDAL Client Policy

Client policies were not implemented in this research. However, a client policy should look substantially similar to a server policy. Clients write policies to protect information stored locally about themselves. This is the information that can be used to satisfy PEDAL server policies. By introducing a round of negotiation between client and service, one could automatically determine which information is acceptable for handing over to a server without the need for users to repeatedly enter data into HTML forms. The implementation would be some sort of user-agent plug-in.

The vocabulary used by a client policy to describe services would differ from those most often used by PEDAL services, such as Contact. Most likely the client vocabulary could take most of its cues from P3P and APPEL 3.2.

5.2.7 PEDAL Policy Negotiation

Parties may not agree on a policy's contents. Negotiation attempts to reconcile differing policy statements by choosing the most reasonable winner in conflicts. However, the ultimate choice goes to the Policy Authority role's policy when all other rules for choosing a winner fail.

To reach an agreement between parties on an acceptable policy, PEDAL uses the concept of roles and requirements. Policies are authored by an entity who is assigned a certain role. Depending on their role, their policy will carry more or less weight when it comes to negotiation. This is one particular area where authentication is absolutely necessary since policy writers may choose to maliciously claim to hold a role that isn't theirs. The end product is given the type `:NegotiatedPolicy`. Normally this is the Author role's policy reborn, in order to give some sense to the namespaces being used.

In addition, policy statements include a mechanism for describing the requirement associated with a rule much like the one described in *Key words for use in RFCs to Indicate Requirement Levels* [25], a scheme to describe implementation requirements described in design documents called Requests for Comments (RFCs).

"Weighting" statements in this sense is accomplished by rules. The roles and priorities have no inherent ordering, rather, their weight is established within negotiation rules. A true weighting scheme which assigns some value to roles and priorities could be used for greater extensibility and more general rules, though the complexity would increase.

The `:Role` class is as described above. There are two roles to consider, those on the server side and those on the client side. Server side roles are, at this time, all considered to be different classes of policy authors and are used in weighing out policy negotiations. Client side roles are used in access rules.

The `:Author` role is a server role. A document's author can provide metadata for the document and in addition provide a PEDAL policy governing the distribution of the metadata. That person or set of persons is considered to be the `:Author`.

The `:PolicyAuthority` role is a server role which is allowed to specify what happens if no `:Author` supplied rules are given. The Policy Authority role also has more weight than the Author role in

conflict resolution.

More roles can be created as may be useful, though the rules about negotiation must also expand to include the additional roles and their weights.

The `:hasPriority` predicate describes the requirement level which a policy author thinks their rule should carry.

Members of `:Priority` class are based off RFC 2119 [25], though only in positive form. As the next section shows, the combination of positive and negative priorities is unnecessarily complex. The three predefined priorities are `:Must`, `:Should`, and `:May`, each carrying the normal English denotation of each word.

As can be expected, rules with `:Must` carry more weight than those with `:Should`, which in turn carries more weight than `:May`. Synonyms are provided for these terms. Again, a finer granularity can be introduced, but the negotiation rules must be adjusted to fit the ranking of any new imperatives.

When enough information is included in a policy, negotiation rules are basically simple to formulate, though large in size. To negotiate, information about the policy author, the intended visibility, resource, priority, and predicate clause and range must be taken into account. Not all of the negotiation rules are listed here because they require so much information to be complete.

For example, the rule where the author and policy authority conflict over a resource's visibility using the same predicate criteria and the same `:Must` priority is "resolved" by eliminating the author's statement and providing a warning about that statement being superseded.

```
{{ :a pedal:authoredBy pedal:PolicyAuthority .
   :z pedal:authoredBy pedal:Author .
   :b pedal:forPolicy :a .
   :y pedal:forPolicy :z .
   :b pedal:forResource :m .
   :y pedal:forResource :m .
   :b pedal:withVisibility :l .
```

```
    :y pedal:withVisibility :n .
    :l log:notEqualTo :n .
    :b pedal:hasPriority pedal:Must .
    :y pedal:hasPriority pedal:Must .
    :b pedal:hasComponent :c .
    :c pedal:withPredicate :d .
    :y pedal:hasComponent :x .
    :x pedal:withPredicate :w .
    :w log:notEqualTo :d . } log:implies
  { :b pedal:forPolicy :z .
    :x a log:Chaff .
    :y a log:Chaff .
    :y pedal:hasWarning "Statement was superseded" .
    :y pedal:warnAuthor pedal:Author . }} a log:Truth ;
log:forAll :a, :b, :c, :d, :l, :m, :n, :w, :x, :y, :z .
```

This rule also moves policy statements which pass negotiation under the `:NegotiatedPolicy` while generating a warning with `:hasWarning` for whichever policy composer is losing its statement, specified by `:warnAuthor`. An application should extract any warnings for the author before purging the losing statements.

The author's policy is given the additional classification as a `:NegotiatedPolicy`. This allows its policy statements to remain where they are while merging the policy authority's statements in with it. This seems to be the best method for merging since the policy authority's document may govern more than one resource as a central policy; the author's policy, however, is only for one resource and may be safely modified with the author's consent.

Another set of rules cleans up the negotiation process by eliminating extra policy declarations and noting the policy authority as a contributor to the negotiated policy.

One of the issues encountered in designing the negotiation component is correctly determining that conflict exists. There are simple cases like the above where policy statements want to arrive at different conclusions using the same criteria. Obviously one or the other must win out in the end. But a more complex case exists where statements want different conclusions using different criteria. Unless some deep knowledge of each set of criteria in use can be guaranteed, the best fallback

method is to assume that the criteria will conflict for some set of users and to resolve the conflict at the negotiation stage. More definitive negotiation rules could probably be written, but the system may get arbitrarily complex as more variables enter the equation. The design tradeoff here is more simplicity in exchange for occasionally being wrong about the existence of conflict.

5.2.8 Miscellaneous PEDAL Rules

A number of rules concerning rule generation and rule application are used to apply PEDAL policies to metadata request situations. Each is a very small set of rules, demonstrating the power of the logic processor, even as developers continue to improve it.

PEDAL policy statements frame rules. That is, information from a statement can be logically manipulated by a rule to form a rule. Objects of the properties `:forResource`, `:withVisibility`, and `:ruleSubject` can be extracted and then combined to form a rule where any requester matching the predicates and ranges of the `:ruleSubject` will have the given visibility for the given resource. To write a rule using a rule requires an awkward syntax (see directly below). There is an inner `log:implies` and an outer one. When run with a policy, the generating rule, the outer `log:implies`, should conclude the formula with the inner one. These generated rules are then used later. The *generating rule*, in N3:

```
{{ :x a pedal:PolicyStatement .
   :x pedal:forResource :y .
   :x pedal:withVisibility :z .
   :x pedal:ruleSubject :p .
   :p log:includes { :a :b :c . } . } log:implies
 { {  :p log:implies
    { :y :z :a . }} a pedal:AccessRule ;
                    a log:Truth ;
                    log:forAll :a . }} a log:Truth ;
        log:forAll :x, :y, :z, :p, :a, :b, :c .
```

The `:ruleSubject` property, as stated previously, is an unfortunate product of *cwm*'s current in-

ability to process lists correctly. Using the `:withPredicate` and `:withRange` properties to generate rules was the intent, but it is impossible to generate rules with more than one statement in the subject without resorting to lists. When *cwm*'s list handling capabilities are improved, then perhaps the generating rule can be modified to reflect the improvement.

The current version of the rule succeeds due to the statements using the variable `:p`. Because the resource bound to `:p` is the object of a `:ruleSubject` predicate, `:p` must refer to a simple statement or set of statements. This means `:p` must include something that follows the normal triple form of `:a :b :c`. With the rule requirements satisfied, the rule in `:p` and the conclusion formed from `:y`, the resource under consideration, and `:z`, the visibility, can be included in the generated rule. `:a` is also of interest for binding. Whatever matches `:a` can really be anything since it actually turns into a variable, as seen in the inner `log:forAll` clause. The resource bound to `:a` is already in `:p`; this is the reason the variable `:someone` is used in all the policy statement rule examples. The final rule, when used, will look for a resource to bind to `:someone`.

A PEDAL policy is used as the input to *cwm*. The policy is filtered through the generating rule, which the logic processor runs once and subsequently outputs only rule conclusions.

The result is a set of rules used to filter the policy-protected metadata and anything known about the requester. This information is gathered from the client and expressed using `:Requester` role to keep the identity under consideration consistent. The result again of this process is a set of statements about what the requester is allowed to access. When combined with the policy-protected metadata and filtered through one last set of rules, the final set of appropriately viewable metadata is returned.

The last set of rules simply restates the metadata if it is visible or marks the metadata as `log:Chaff` if not. *cwm* then purges anything marked as `log:Chaff`. By virtue of filtering, *cwm* won't output metadata not mentioned in the policy, meaning PEDAL acts as if metadata is hidden by default.

```
{{ :x pedal:visibleTo :y .
   :a :x :b . } log:implies
 { :a :x :b . }} a log:Truth ;
        log:forAll :x, :y, :a, :b .

{{ :x pedal:hiddenTo :y .
   :a :x :b . } log:implies
 { :x a log:Chaff }} a log:Truth ;
        log:forAll :x, :y, :a, :b .
```

One might note that metadata could be declared hidden and visible at the same time in a set of rules, given that the metadata requester combines the correct set of attributes. While in the end this would conclude the metadata is hidden, it would be avoidable by only writing `:visibleTo` statements. Only using `:visibleTo` statements isn't always possible, however, and one of the byproducts of negotiation is eliminating possible conflicts in access declaration. One could view negotiation as a sort of caching of conflict resolution.

5.2.9 Authentication

Perhaps the most pressing issue not fully explored in constructing PEDAL is the issue of authentication, that is, how can a service be assured that clients are truthfully representing their information while trying to make a decision on access control? Or that a policy author isn't trying to corrupt negotiation by lying about their identity? Troublesome clients could attempt to reverse engineer the rules governing a resource by some brute force tactics, resubmitting different personal information until the desired result is achieved, violating the rules trusted by the author to the service for enforcement.

Different solutions which immediately suggest themselves are briefly commented upon in section 7.1; however, no concrete solution was chosen or tested. More work and research may be invested here profitably.

5.3 PEDAL Software

The development of related PEDAL software was done on both a Mandrake Linux 8.2 and a Red Hat Linux 7.1 system running the Apache 1.3.23 web server, interfaced to the PostGreSQL 7.0.3 relational database. Software was written in Python 2.2.1, based on software developed at the W3C, primarily by Dan Connolly and Tim Berners-Lee. Their code is a package referred to as `cwm` (see section 4.5) which is constantly under development. PEDAL code is based on `cwm` version 1.82; whenever `cwm` reaches a stable major version, it would be beneficial to bring PEDAL code up to date as well, which should not be difficult since PEDAL makes use of `cwm`'s current API without modifying any `cwm` code.

PEDAL software should run on any web server which executes Common Gateway Interface (CGI) programs, runs a Python 2.2.1 interpreter, and has some sort of database drivers. The driver and interface used in this environment is PyGreSQL 3.2, obtained from <http://www.druid.net/pygresql/>. Database use is light at this point, consisting solely of record fetches.

5.3.1 Peddler: PEDAL Generator

A policy generator based on a service's own PEDAL policy called *Peddler* assists users in generating their own PEDAL policies. Peddler's appearance depends on the configuration in the service's PEDAL policy; Peddler does not generate generic policies. Rather, it parses the service's PEDAL policy to determine which vocabularies it can reliably offer as rule components. A tool that can compose generic policies would be less effective than a text editor since the data being entered could take on an overly large set of values while restricting data entry to unwieldy HTML forms.

Peddler can only be used when metadata for a resource has been provided. The generator is based on an HTML form interface to be used through a Web browser. It is broken into four components. The first gathers the name of the policy (figure 5), having been initiated through Bouncer (section

5.3.4).

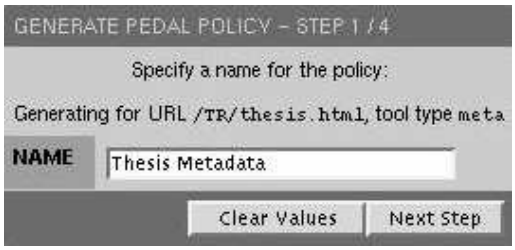

Figure 5: First step in PEDAL policy generation

The next step processes the provided metadata through cwm, then determines which namespaces it `:supports`, and finally offers an interface (figure 6) for composing rules.

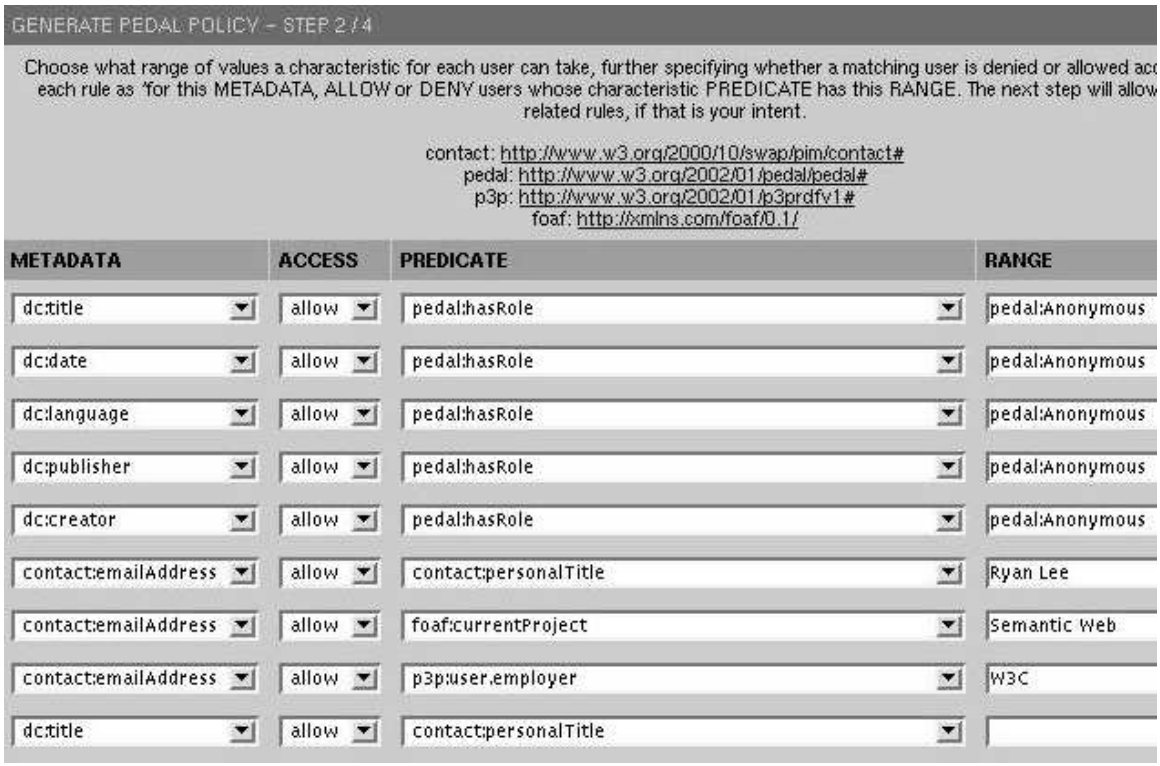

Figure 6: Second step in PEDAL policy generation

The input from this step goes into a rule combination form where rules for the same resource and visibility can be merged into a larger rule (figure 7). Rules that cannot be logically combined are not displayed.

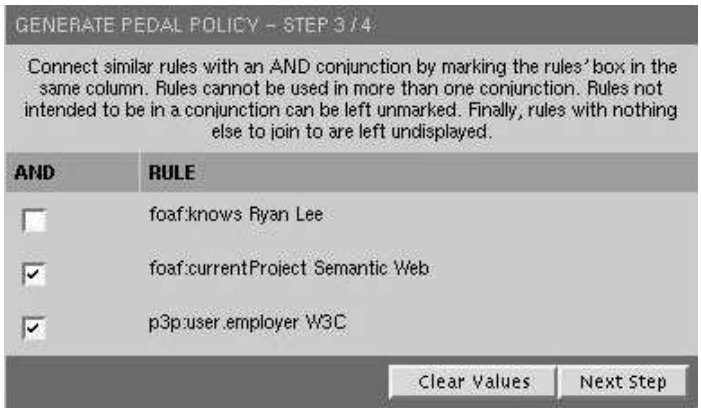

Figure 7: Third step in PEDAL policy generation

Finally, the combined (or uncombined) rules are assigned priorities (figure 8), submitted, and a policy is returned. The mechanisms for input are intended to avoid typing by users (and, by extension, typing mistakes by users) whenever possible. The result of the steps shown in the screenshots is in the appendix, section 11.7.

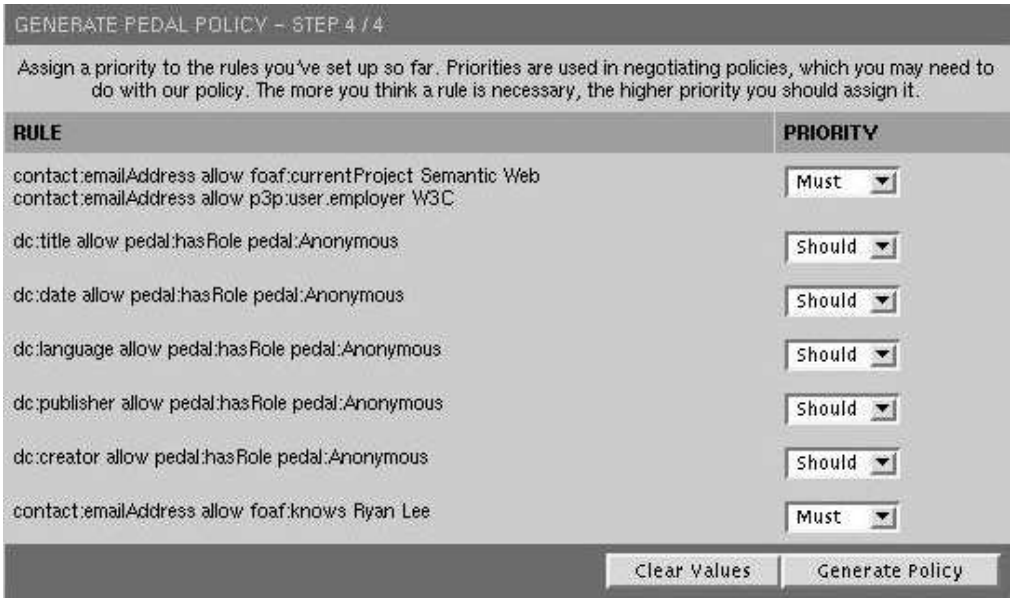

Figure 8: Last step in PEDAL policy generation

Once a policy is generated, it can be downloaded; it can additionally be submitted to services which understand PEDAL to negotiate a policy consistent with both the service and the user's desires.

This negotiation is handled by Bouncer.

5.3.2 Peddler: HTML Library

The user interface of Peddler is generated by an HTML library (section 12.8) which incorporates table and form building into objects. This library eliminates most of the messiness involved with mixing HTML and other code often found in creating web applications. For Peddler, it is a reasonable separation of content and presentation, though others may not find it quite as useful.

5.3.3 Peddler: PEDAL Library

The Peddler library (section 12.4 consists of an extension to `cwm`'s RDF parser. `cwm` is object oriented by design. The main contribution in the Peddler library is a subclass of `cwm`'s `RDFStore` class. In addition to the normal `RDFStore`, it also incorporates a negotiation module.

`cwm` implements the basics for turning an RDF document into a parsed entity. With that component abstracted out, the problem to solve focuses mainly on creating a PEDAL-compliant policy by implementing `cwm` plug-in for "understanding" negotiation. A handful examples for authoring `cwm` plug-ins exist, so the negotiation module is mostly an exercise in understanding `cwm`'s API and correctly implementing the semantics of PEDAL roles and imperatives.

5.3.4 Bouncer: Protection

Bouncer implements PEDAL policy rules (code in section 12.5). Requests for any metadata are performed using the comma tool concept originally used on the W3C's own site for access control and other functions, found at <http://www.w3.org/,tools>. As is implied by the name, a comma tool can be accessed by appending a comma and the name of the tool to a URI. Setting up the recognition of comma tools was accomplished by a simple configuration of Apache to use URL

rewriting (in section 12.1). Requests for comma tools are redirected to the Bouncer script. Two are available in this implementation, the `meta` and the `contact` tools.

Bouncer makes use of the PostGreSQL relational database to query whether a comma tool is available for resources. This is something of a stopgap measure. While RDF querying languages and tools are shaping up, there is currently no clear choice to integrate with a Python-based system. The querying is done with SQL, though the API should allow replacement with an RDF-based querying system when one emerges.

A resource can have no metadata and no policy, only metadata, or both. Having nothing is an uninteresting case (though in Bouncer's current implementation, one can upload metadata - this is a toy example only, a real implementation would guard the upload with a little more rigor). Given that a resource does not have its own policy, though, Bouncer checks if there is a broader policy under which the resource falls. Again, here a policy can be uploaded or generated via Peddler. Bouncer takes care of the negotiation between uploaded or generated policies. If there is no policy at all, then the metadata will default to being completely hidden. With a policy, Bouncer begins the data protection process.

Bouncer uses cwm to reason through the policy generation rules and presents an HTML form to the user to fill in (figure 9). Once the information is submitted, Bouncer matches filled in values against *cwm* generated rules to determine whether or not the user has access to the information provided by a comma tool, and the final set of metadata is returned as raw RDF to the user (figure 10).

Bouncer would perform faster as an Apache module, much like the way ChACL rules are enforced. This would create potential difficulties in interfacing easily with *cwm* and would require a good deal more of cross-platform support. It may be worth investigating should PEDAL ever grow in popularity.

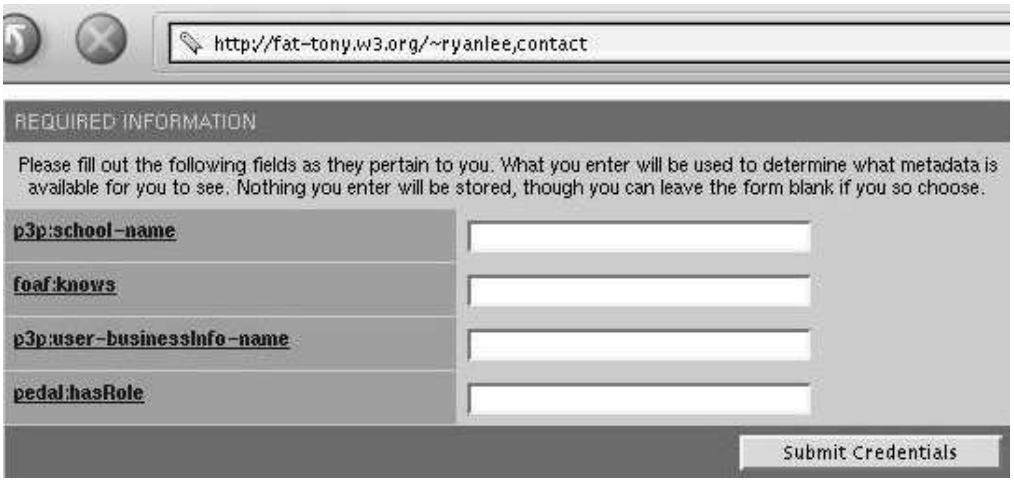

Figure 9: Bouncer Gateway Form

```
<?xml version="1.0"?>

<rdf:RDF xmlns="#"
    xmlns:acl="http://www.w3.org/2001/02/acls/ns#"
    xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#"
    xmlns:dc="http://www.purl.org/dc/elements/1.1/"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:log="http://www.w3.org/2000/10/swap/log#"
    xmlns:p3p="http://www.w3.org/2002/01/p3prdfv1#"
    xmlns:pedal="http://www.w3.org/2002/01/pedal/pedal#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

    <rdf:Description rdf:about="#RL">
        <contact:fullName>Ryan Lee</contact:fullName>
        <contact:publicHomePage rdf:resource="http://ryanlee.org/"/>
    </rdf:Description>
</rdf:RDF>
```

Figure 10: Bouncer Results

6 Results

Six tests were performed to exercise the PEDAL software's ability to protect data and negotiate policies, with a basic, advanced, and real-world application to check each function. The real-world applications are interesting cases culled from current Semantic Web activities where personal data privacy is a concern.

6.1 Basic Protection

Before implementing the data protection strategy described in Bouncer 12.5, simple data, protection, and input were run through *cwm*. `simple-policy` contains a PEDAL policy protecting `simple-metadata`. `simple-input` describes a user requesting `simple-metadata`. *cwm* [2] was then run in the following manner:

```
% cwm -n3 simple-policy -filter=pedal-generate-rules.n3 > rules
% cwm -n3 simple-metadata simple-input -filter=rules > visibility
% cwm -n3 visibility simple-metadata -filter=pedal-access-rules.n3 -purge
```

Running *cwm*

cwm can process either RDF or N3, so a flag (`-n3`) is explicitly added to indicate which file format is being used. An arbitrary number of filenames can be passed in, after which the `-filter` command indicates which file contains rules to filter the other files through. The result of a `-filter` is only the conclusions made by rules. Using the `-think` command and passing in the file as an argument will result in a repeated application of any rules until no more conclusions can be made; everything is then printed out. The `-purge` command removes anything marked as `log:Chaff` before printing out the result.

[2] On the command line, *cwm* is an alias for python2 cwm.py

Expected Results

The `simple-metadata` files are reproduced in full here to better illustrate what results are expected.

The policy allows everybody to see the title, date, language, and publisher metadata for the document <http://www.example.org/doc#>. The creator metadata, however, has been left out, so *cwm* should not come to the conclusion that it is visible to this particular requester, who is anonymous.

Simple Metadata

```
# simple-metadata
@prefix dc:      <http://www.purl.org/dc/elements/1.1/> .
@prefix doc:     <http://www.example.org/doc#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
@prefix :        <#> .

doc: dc:title ‘‘A Simple Test’’ ;
   dc:creator ‘‘Ryan Lee’’ ;
   dc:date ‘‘2002-08-22’’ ;
   dc:language ‘‘en’’ ;
   dc:publisher ‘‘example.org’’ .
```

Simple Input

```
# simple-input
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .

:Requester :hasRole :Anonymous .
```

Simple Policy

```
# simple-policy
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix dc:      <http://www.purl.org/dc/elements/1.1/> .
@prefix p3p:     <http://www.w3.org/2002/01/p3prdfv1#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
@prefix foaf:    <http://xmlns.com/foaf/0.1/> .

here:SimplePolicy a :Policy ;
    :policyName "Simple Metadata Policy" ;
    :forMetadataOf <http://www.example.org/doc#> ;
    :authoredBy :PolicyAuthority .

[ a :PolicyStatement ;
```

```
    :forPolicy here:SimplePolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :forResource dc:date ;
    :forResource dc:language ;
    :forResource dc:publisher ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone :hasRole :Anonymous . } ;
    :hasComponent [ a :Component ;
                    :withPredicate :hasRole ;
                    :withRange :Anonymous ] ] .
```

Actual Results

For the sake of illustration, the result of each step using *cwm* is reproduced to demonstrate rule functioning. The two intermediate results, `rules` and `visibility`, are internal constructs used for more *cwm* processing. In practice, users would only see the end result from the final run of *cwm*.

```
% cwm -n3 simple-policy -filter=pedal-generate-rules.n3 > rules
# rules
@prefix : <http://www.w3.org/2000/10/swap/log#> .
@prefix dc: <http://www.purl.org/dc/elements/1.1/> .
@prefix pedal: <http://www.w3.org/2002/01/pedal/pedal#> .

{{ pedal:someone pedal:hasRole pedal:Anonymous . } :implies
 { dc:publisher pedal:visibleTo pedal:someone . }}
    a :Truth,
      pedal:AccessRule;
    :forAll pedal:someone .

{{ pedal:someone pedal:hasRole pedal:Anonymous . } :implies
 { dc:title pedal:visibleTo pedal:someone . }}
    a :Truth,
       pedal:AccessRule;
    :forAll pedal:someone .

{{ pedal:someone pedal:hasRole pedal:Anonymous . } :implies
 { dc:date pedal:visibleTo pedal:someone . }}
    a :Truth,
       pedal:AccessRule;
    :forAll pedal:someone .

{{ pedal:someone pedal:hasRole pedal:Anonymous . } :implies
 { dc:language pedal:visibleTo pedal:someone . }}
```

```
    a :Truth,
      pedal:AccessRule;
    :forAll pedal:someone .
```

This looks to be correct so far. The rules generated from `simple-policy` all state that anyone should have access to all metadata save `dc:creator`.

```
% cwm -n3 simple-metadata simple-input -filter=rules > visibility
# visibility
@prefix : <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix dc: <http://www.purl.org/dc/elements/1.1/> .

dc:date :visibleTo :Requester .
dc:language :visibleTo :Requester .
dc:publisher :visibleTo :Requester .
dc:title  :visibleTo :Requester .
```

Again, when run against the actual metadata and the input, the generated rules conclude exactly that the requester can see the four pieces of metadata.

```
% cwm -n3 visibility simple-metadata -filter=pedal-access-rules.n3 -purge
@prefix doc: <http://www.example.org/doc#> .
@prefix dc: <http://www.purl.org/dc/elements/1.1/> .

doc: dc:date "2002-08-22";
    dc:language "en";
    dc:publisher "example.org";
    dc:title "A Simple Test" .
```

As the final output shows, the basic protection test worked as planned.

6.2 Advanced Protection

Moving beyond the simple test, the advanced test provides more metadata, more policy statements about the metadata, and more characteristics from the requester as input. The policy in particular demonstrates a number of cases which are exercised by different inputs:

Intentional conflict The `:hiddenTo` property is mostly useful when a set of users has been defined for which a resource is visible, but further refinement is necessary to define a set for which the resource is not visible. Policy statements may conflict with one another if the intent is to hide something that would otherwise be visible.

Compound rules Rules that have more than one requirement are tested.

Cross statement requirements Statements may share the same requirements for a rule to be satisfied.

The metadata, policy, and inputs with their corresponding results can be found in the appendix in section 11.2.

Expected Results

The policy used in this test expresses the following statements:

- The title, publication date, and language that describe the resource must be visible to everyone.
- The publisher's information is only visible to employees of the company.
- Furthermore, the creator's contact information is only visible to members of the same department of the same company.
- Lastly, anyone named "Ben Bitdiddle" is disbarred from any of the creator's contact information.

Six different inputs with different requester information are used to test the policy's eventual resolution. They are summarized in the following table.

User Charateristics	Expected Metadata
Anonymous	Title, language, and date
Employee of a different company	(same as above)
Employee of the same company	Title, language, date, publisher's information
Employee of the same company, different department	(same as above)
Employee of the same company, same department	All document, publisher, and creator information
Employee of the same company, same department, named "Ben Bitdiddle"	Only document and publisher information

Actual Results

Examination of the inputs and outputs in section 11.2 clearly show that inputs match up to their expected results.

6.3 Contact and Friend-of-a-Friend (Protection)

Protecting publicized contact information is an interesting application for protection. People who decide to publish their personal contact information and relationship expressions are likely to be the only parties interested in protecting the information, so no publisher negotiation over policy conflicts should be necessary.

The Contact schema, located at <http://www.w3.org/2000/10/swap/pim/contact.n3>, provides a vocabulary for describing general contact information. The Friend-of-a-Friend (FOAF) schema <http://xmlns.com/foaf/0.1/> also defines some vocabulary for contact information in addition to interrelations between people.

6.3.1 Contact

Contact is based off of the Microsoft Outlook contact information scheme. Many of the contact information items one might expect to find in a corporate address book such as phone number and department name can be found in the schema as properties. The Contact schema is frequently used

whenever such information needs to be shared and recorded, such as teleconference meetings. At W3C, however, most of the time this information is accessible only to those who belong to the right group.

6.3.2 FOAF

FOAF is used to describe relationships between people. Its authors have a list of interesting FOAF applications at <http://xmlns.com/foaf/> including the ability to find friends who have published FOAF documents and to trace who knows who through photograph description.

6.3.3 Test Description

This test is essentially a check that protection works in a real world application. Bouncer is a friendly interface for generating the inputs to protection decisions. Since Bouncer ties directly in to *cwm* and *cwm*'s success with protecting information was shown in the previous two test cases, this test is essentially a confirmation that integrating *cwm* with a Web server is feasible. The policy in section 11.6 was used for the Bouncer URL `/~ryanlee,contact`, which would provide contact information for the user `ryanlee`. Where document publication data access policies can be straightforward, policies regarding contact information access may take on a more eclectic set of statements. Some people are comfortable baring all their contact information to the Web, while some are much more protective. The policy governing `ryanlee`'s data, in section 11.5, tries to strike a balance.

Expected Results

The policy describes rules that only allow the email address to be given to those who know the user and attend his academic institution (admittedly, there is no schema at the moment for saying "this user attends this school," but this is mostly a theoretical exercise). The email address can also be

given to those who are employed at the same institution. The full name and public homepage are availble to anyone.

Actual Results

One can visit the implementation at a live demo, <http://www.w3.org/2002/01/pedal/demo>, to verify that Bouncer indeed does obey the policy correctly.

6.4 Basic Negotiation

Simple, contradictory policies were run through *cwm* to check negotiation rules. `policy` contains a PEDAL Author policy with one statement, while `opposing-policy` contains a PEDAL Policy Authority policy that disagrees with `policy`. Tests were run using:

```
% cwm -n3 policy opposing-policy pedal-negotiation-rules.n3 \
  -think -purge > negotiated
% cwm -n3 negotiated -filter=pedal-cleanup-negotiation-rules.n3 -purge
```

Expected Results

Again, the full policies are reproduced for illustrative purposes. The two conflict directly by stating opposite access policies for the same class of users for the resource `dc:title`. Since both parties give their policy statements the `:Must` priority, the Policy Authority's statement should win out over the Author's.

```
# policy
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix dc:      <http://www.purl.org/dc/elements/1.1/> .
@prefix p3p:     <http://www.w3.org/2002/01/p3prdfv1#> .
```

```
here:DocumentPolicy a :Policy ;
    :policyName "Overall Document Policy" ;
    :forMetadataOf <http://www.example.org/doc#> ;
    :authoredBy :Author .

[ a :PolicyStatement ;
    :forPolicy here:DocumentPolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :withVisibility :hiddenTo ;
    :hasComponent [ a :Component ;
                    :withPredicate :hasRole ;
                    :withRange :Anonymous ] ] .
```

```
# opposing-policy
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix dc:      <http://www.purl.org/dc/elements/1.1/> .
@prefix p3p:     <http://www.w3.org/2002/01/p3prdfv1#> .

here:DocumentPolicy a :Policy ;
    :policyName "Document Policy" ;
    :forMetadataOf <http://www.example.org/doc#> ;
    :supports dc: ;
    :supports p3p: ;
    :authoredBy :PolicyAuthority .

dc: a :Grammar ;
    :abbreviatedAs "dc" .

p3p: a :Grammar ;
    :abbreviatedAs "p3p" .

# Policy statements
[ a :PolicyStatement ;
    :forPolicy here:DocumentPolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone :hasRole :Anonymous . } ;
    :hasComponent [ a :Component ;
                    :withPredicate :hasRole ;
                    :withRange :Anonymous ] ] .
```

Actual Results

The intermediate, unpurged results are very verbose since a lot of things end up as chaff. The result of running both *cwm* commands is shown below, and the anticipated result of the Policy Authority's statement being chosen is reflected by looking at the visibility clause in the last line.

```
@prefix : <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix dc: <http://www.purl.org/dc/elements/1.1/> .
@prefix here: <policy#> .

here:DocumentPolicy a :NegotiatedPolicy,
                      :Policy;
        :authoredBy :Author;
        :contributedBy :PolicyAuthority;
        :forMetadataOf <http://www.example.org/doc#>;
        :policyName "Overall Document Policy" .

[ a :PolicyStatement;
        :forPolicy here:DocumentPolicy;
        :forResource dc:title;
        :hasComponent  [
            a :Component;
            :withPredicate :hasRole;
            :withRange :Anonymous ];
        :hasPriority :Must;
        :ruleSubject {:someone :hasRole :Anonymous . };
        :withVisibility :visibleTo ].
```

6.5 Advanced Negotiation

The advanced negotiation test takes two policies, each consistine of one policy statement at a time and negotiates between them. This examines the correctness of negotiation rules so far by stressing each combination of variables. The following table describes what happens in each situation when the author, priority, access, predicate, and range are in different states for two policies that are for the same resource. Uninteresting cases where no negotiation takes place are left out. Interpreting the table, when the author's priority is compared to the policy authority's (the author's being on the left of the comparator) and the other variables are either the same or different, the expected

action is either the policy authority (PA)'s statement being deleted or the author's (A).

The action is mostly determined by the priority and role, though the other variables mitigate whether or not a conflict is really detected or not.

Priority	Access	Predicate	Range	Expected
<	same	same	same	PA delete
=				PA delete
>				PA delete
<	diff			A delete
=				A delete
>				PA delete
<		diff		A delete
=				A delete
>				PA delete
<			diff	A delete
=				A delete
>				PA delete

Expected Results

A small Python script (section 12.9) was used to generate twenty-four files matching the truth table. For each of the twelve interesting cases above, the results from running *cwm* negoitation should match with the last column in the table.

Actual Results

Those who have desire to test their N3 and PEDAL comprehension can wade through each of the twelve cases to verify that negotiation did occur as intended. See section 11.3 for the files and their

negotiated outputs.

6.6 Document Publication (Negotiation)

The W3C hosts a large body of research pertaining to Web development from different points in the publication process, from notes to candidate recommendations. Metadata can be associated with each of these documents by their authors. They can protect any personal information contained in the metadata through PEDAL. This test case is meant to exercise the ability to negotiate between different parties about protecting a resource of common interest.

The Dublin Core (DC) is one ontology suggested for the Semantic Web for specifying document publication information.

6.6.1 Dublin Core (DC)

Metadata on the Web took cues from an older type of information repository, the libraries. The Dublin Core Metadata Initiative was founded by the Online Computer Library Center (OCLC) as an "organization dedicated to promoting the widespread adoption of interoperable metadata standards and developing specialized metadata vocabularies for describing resources that enable more intelligent information discovery systems." [23] The organization has produced the Dublin Core (DC) metadata vocabulary, found at <http://purl.org/dc/elements/1.1/>.

The DC vocabulary presents fifteen verbs for standard metadata used in describing a resource, such as the author, publisher, and date of publication, which experts consider to make up a central body of information about a document.

6.6.2 Test Description

In this case, there is one policy theoretically authored by the W3C which controls access to all metadata for its publications, reproduced in full in section 11.4. The W3C is considered the policy authority and has the final say in any negotiation conflicts. The theoretical policy is intended to allow as much disclosure as is sensible. Again, this real-world application test is meant more to insure the workings of the Peddler apparatus since it is simply built atop of *cwm*.

Expected Results

Negotiations with the given policy are expected to follow the defined negotiation rules. One can use the Peddler PEDAL policy generator to create a policy as was shown in screenshots throughout section 5.3.1. The next step after policy generation is negotiation, and Peddler should work with *cwm*.

Actual Results

Aagin, the best way to see Peddler at work is to watch it. Choosing a valid document from within the demo site at <http://www.w3.org/2002/01/pedal/demo> and appending the `,meta` comma tool to the URL would bring up the Bouncer metadata interface. After providing some metadata for a URL, users can then engage the Peddler policy generator. Depending on their inputs, the negotiated policy might look like the one generated in the Peddler screenshots, reproduced in section 11.7.

7 Further Work and Research

PEDAL is but a brief research project. It has its shortcomings and drawbacks, some of which are expanded upon in the following sections, particularly the question of authentication and the scope of PEDAL.

7.1 Authentication

As was mentioned above, the PEDAL design and implementation are incomplete when it comes to authentication. Modern solutions to the problem of authentication are fairly good, but the history of implementing authentication algorithms and protocols reveals a consistent set of repeated errors. Authentication can be a delicate field, so choosing a proven solution adaptable to fulfill PEDAL's authentication needs would be ideal. Some potential mechanisms based on proven public key technology are suggested below.

In brief, public key technology is based on the need to verify messages transmitted across insecure channels. Instead of attempting to secure the channel, keys are exchanged to mathematically verify that the sender of a message is who they claim to be. The sender is armed with two keys, a public and a private one. The private key, known only to the sender, is used to 'sign' a message (or, more appropriately, a 'hash' of the message - a unique and easily recomputable fingerprint of the message). The sender's identity can be verified with the sender's public key and the signed message hash. Public key technology is extended through certificates. A certificate is a signed statement from one entity saying that the public key and identifying information of another entity is valid. A Public Key Infrastructure (PKI) is a system built for certificate management and distribution.

Secure Socket Layer (SSL)

The Secure Socket Layer (SSL) protocol is frequently used by Web clients to verify the authenticity of services. Certificates identifying a service are signed by a Certification Authority, an entity which, in an ideal world, everybody should trust because of a rigorous signing process in which only verified services get their certificate signed. Client certificates are also available; user agents receive a certificate verifying that the user is who they claim to be. This architecture is implemented for student use at MIT: MIT, the certification authority in this case, signs verified services' certificates, such as online academic transcript access. Students are given certificates by the school to authenticate them for privately accessing services, an important part of an online transcript service.

Using SSL would be difficult to implement since SSL, hailed as a solution to the problem of authenticated identification on the Internet, has still not widely caught on when it comes to client certificates. Other difficulties include viable models for carrying out user agent certification authority responsibilities and advertising the benefits of widespread SSL use in the face of an Internet population which warily guards its anonymity.

Additionally, SSL authenticates a 'session,' a somewhat ambiguous term to describe some transaction between a user and a service, such as accessing a transcript. Only the session can be authenticated in SSL; documents and finer granularities of information cannot, so using SSL as is is not enough.

The certificate structure used in SSL is referred to as X.509 [27]. Because X.509 (version 1) has been around since 1988 and is used as the IETF's Internet PKI certificate standard, its implementations are generally well tested and could be used in a different context. Though most often used for Web browsers, X.509 is also used for other purposes such as securing email.

Simple Distributed Security Infrastructure (SDSI)

Where X.509 was intended to address the question of authentication, the Simple Distributed Security Infrastructure (SDSI) [28] is intended to prove authorization through certificates. Naming and authorization imperatives used in conjunction with the Simple Public Key Infrastructure (SPKI) [29] certificates can be used to both authenticate identity and verify authorized actions. The key difference between SPKI certificates and X.509 certificates are their intended uses. SPKI goes beyond identification to specify authorization, which is often why certificates are necessary: once correctly identified, a service must still determine if the certificate presenter is authorized to perform some action, a decision usually not based on the person's name.

This brings up an area of conflict between PEDAL and SPKI/SDSI as PEDAL is intended to specify authorization within the Semantic Web. SDSI naming asserts that local names can be used globally by chaining through others' names. The portion of SPKI/SDSI which authenticates these local SDSI names could be highly useful, but some middle ground between PEDAL and SPKI/SDSI authorization should be achieved before moving in this direction.

XML Signatures

The XML-Signature [30] standard, co-authored by both the Internet Engineering Task Force (IETF) and the W3C, provides a standard for signing documents. Its strength lies in the ability to indicate the signing of different parts of a document's XML tree and to be represented itself in XML. This produces a finer granularity than schemes like SSL for authenticating information.

Another of the key advantages of XML-Signature technology is the ability to specify which certificate scheme is in use. No *a priori* scheme is tied to XML-Signature, a clear advantage since explicitness about anything that matters is one of the first principles in security and authentication. XML-Signature currently provides for RSA and DSA key schemes and X.509, SPKI, and PGP certificate

schemes.

XML-Signature in combination with X.509 or SPKI/SDSI appears to be one of the best areas for further investigation.

7.2 Scope

PEDAL's scope is currently set on protecting user metadata. Applications using PEDAL can expand its uses, and some thoughts on useful PEDAL-based applications are also given below.

Web Services

A web service is essentially an application available through the Web. The original scope of this project was to negotiate the breadth of a web service depending on the client. This proved to be too broad since the amount of background work leading up to a web service negotiation application has already proved to be extensive. This is perhaps the next logical step for PEDAL. It already protects data, it can probably be used to protect service offerings given the proper vocabulary and application.

Some work was done with the Web Services Description Language (WSDL) [31], a markup language originally designed in XMLSchema. The purpose of WSDL is to provide a standard method for describing web services. WSDL is a work in progress, being refined by the W3C Web Services Working Group. For the original purposes of this project, WSDL 1.1 was rewritten in N3 and named WSDLRDF to provide statements about web services *cwm* could reason through.

The WSDL schema is formally described in [31]; future version upgrades are likely. The translation process in writing WSDLRDF deviated as little as possible from the meanings in [31], though WSDLRDF does suffer from a lack of ability to describe data types, a function XMLSchema is designed for.

WSDLRDF and its associated extensions can be found at <http://www.w3.org/2002/01/pedal/wsdl/>.

Beyond Information Protection

PEDAL currently uses the binary access method of `:visibleTo` and `: hiddenTo`. It may be worthwhile in other cases to provide finer granularities of access control beyond simple allowance or denial. Operating systems, for example, provide read, write, and execution privileges. This type of access scheme could tie in well with web service negotiation where information can be manipulated and not merely viewed.

7.3 PEDAL as Protocol

Like HTTP and P3P, PEDAL is a protocol. Protocols only function when involved parties follow the protocol's guidelines. The implementation of PEDAL with this research, however, is an application on top of HTTP. An infrastructure which could guarantee PEDAL policies being properly followed would be a vast improvement. Simply including the policy within a document and publishing it while being assured that the publishing server would obey the policy could be achieved with some sort of HTTP/PEDAL hybrid server. This PEDAL-aware server must obey PEDAL policies as they are written. The two protocols must succeed or fail together; nobody should be able to access the full contents of a file by somehow bypassing the policy rule checking process.

Using the current PEDAL software as an application as shown in this research should be considered an experiment and not representative of any real-world implementation.

Temporal Concerns

Caching and the lifespan of information are important to take into account. Caching documents can vastly reduce the amount of time for multiple users to obtain a document since it reduces the load

on the original server and moves the document closer in the network topology to users. In section 13 of the HTTP RFC [32], the authors discuss caching in HTTP. Of interest are outlines of HTTP caching correctness, document expiration, and cache validity.

Much like HTTP, PEDAL controlled information may end up using caches to improve performance. It is important for the cache both to properly follow the PEDAL policy controlling that information and to eventually purge the information lest it serve stale data; information and policies both may change over time.

Some additional vocabulary is needed for declaring how long a cached or general piece of information should be kept before expiring. Lessons can be taken from HTTP as well as general network caching and hardware memory cache designs.

7.4 PEDAL Software

The set of software written for this research is essentially a bare minimum for demonstrating PEDAL's viability. Additional tools which could greatly enhance further research and general PEDAL use could include:

PEDAL validator Verifies that both server and client policies use the PEDAL grammar properly.

PEDAL server Either modifying an HTTP server or writing a module for an HTTP server to handle requests for documents under a PEDAL policy, improving performance.

User-agent plug-in A user-agent plug-in for automatic handling of PEDAL authorization requests. The plug-in could adapt to a user's preferences as they encounter unfamiliar policies and metadata schemas.

8 Conclusion

As progress continues on developing the Semantic Web for popular use, the problem of preserving personal privacy can be addressed using that which makes the Semantic Web so powerful. Logical rules written using the PEDAL vocabulary can be reasoned through using existing Semantic Web tools. The rules can express more by describing the set of users who are being allowed or denied access to personal information while also incorporating the ability to use rules relating to existing prior-agreement user bases. The granularity of data to which these rules can be applied is at the resource level, so PEDAL scales from a simple protection of one piece of personal data to an entire collection of documents.

PEDAL also includes a negotiation component for different parties to automatically come to an agreement on any opposing rule statements within their PEDAL policies regarding published meta-data. While a deployable PEDAL scheme is incomplete without an authentication component, authentication would provide prevent malicious users from making harmful policies and from brute forcing the discovery of a private policy. As PEDAL is based on RDF, it is also easily extensible.

This is a first step in examining privacy protection within the Semantic Web. Building a global infrastructure where guarantees of privacy can be made is one compelling reason for further research into completing PEDAL and investigating its uses as well as general research towards creating a Semantic Web.

9 Appendix: Acronyms

All acronyms used throughout the above text are expanded here.

ACL Access Control List

APPEL A P3P Preferences Exchange Language

DTD Document Type Definition

FOAF Friend-of-a-Friend

FTC (United States) Federal Trade Commission

FTP File Transfer Protocol

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IETF Internet Engineering Task Force

MIT Massachusetts Institute of Technology

N3 Notation3

P3P Platform for Privacy Preferences Project

PEDAL Personal Data Access Language

PGP Pretty Good Privacy

PICS Platform for Internet Content Selection

PKI Public Key Infrastructure

RDF Resource Description Framework

SOAP Simple Object Access Protocol

SPKI/SDSI Simple Public Key Infrastructure/Simple Distributed Security Infrastructure

TCP/IP Transmission Control Protocol/Internet Protocol

URI Uniform Resource Identifier

URL Uniform Resource Locator

W3C World Wide Web Consortium

WSDL Web Services Description Language

XML Extensible Markup Language

10 Appendix: Terms

Definitions of terms as used in this paper.

client A machine requesting resources. User agents are normally the software used by a user on client machines to make requests from servers.

Internet The resources available through the TCP/IP communications layer, beyond just the Web, including e-mail, gopher, IRC, FTP, and instant messaging.

metadata Data about data; i.e., publication data about books, such as author and publisher, could be considered metadata.

namespace A context for understanding the definition of terms; i.e., "keyboard" in a musical namespace describes an instrument, while the same word in a computing namespace describes a text input device.

policy In the context of PEDAL, a policy is a set of statements declaring one party's criteria for others to access the metadata about a resource.

resource A resource is anything that can be defined by a URI; examples of resources are web pages, email addreses, people, and organizations. Just about anything can be assigned a URI, meaning just about anything, physical or abstract, can be a resource.

server A machine providing resources; accessed by clients.

service (or *web service*) An application made available through the Web.

Semantic Web A vision for the future of the Web in which information is not only machine-readable but also machine-understandable, allowing for greater automation and intelligent use of data.

user A person interacting with the Internet.

user agent The software used by a person to connect to the Internet; common user agents are referred to as "web browsers," such as Netscape/Mozilla or Internet Explorer. User agents act on behalf of a user and can perform tasks beyond simple resource acquisition on the Web.

vocabulary Also *grammar*, *data model*, *schema*. A vocabulary provides the definitions for a related set of terms, rooted at a namespace.

Web The World Wide Web, the interconnected resources available through HTTP, as it stands today.

11 Appendix: Grammars

All of the grammars written in RDF Schema along with examples of them in use are provided in this appendix.

11.1 PEDAL

```
# $Id: pedal.n3,v 1.10 2002/08/21 15:49:27 ryanlee Exp $
# Schema for Personal Data Access Logic (PEDAL)

@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix log:    <http://www.w3.org/2000/10/swap/log#> .
@prefix acl:    <http://www.w3.org/2001/02/acls/ns#> .
@prefix dc:     <http://www.purl.org/dc/elements/1.1/> .
@prefix :       <#> .

<> dc:title "Personal Data Access Logic (PEDAL) Schema" ;
    dc:creator "Ryan Lee" ;
    dc:date "2001-12-20" ;
    dc:language "en" ;
    dc:publisher "The World Wide Web Consortium (W3C)" .

###
# Policy classes and properties
###

:Policy a rdfs:Class ;
    rdfs:label "Policy" ;
    rdfs:comment """A metadata policy concerning a document.""" ;
    rdfs:subClassOf rdfs:Resource ;
    rdfs:isDefinedBy <> .

:policyName a rdf:Property ;
    rdfs:label "Policy Name" ;
    rdfs:comment """Name of a policy.""" ;
    rdfs:range rdfs:Literal ;
    rdfs:domain :Policy ;
    rdfs:isDefinedBy <> .

:forMetadataOf a rdf:Property ;
    rdfs:label "For Metadata Of" ;
    rdfs:comment """Which resource's metadata a policy governs.""" ;
    rdfs:range rdfs:Resource ;
    rdfs:domain :Policy ;
```

```
    rdfs:isDefinedBy <> .

:authoredBy a rdf:Property ;
    rdfs:label "Authored By" ;
    rdfs:comment """Who authored a policy.""" ;
    rdfs:range :Role ;
    rdfs:domain :Policy ;
    rdfs:isDefinedBy <> .

:contributedBy a rdf:Property ;
    rdfs:label "Contributed By" ;
    rdfs:comment """Who contributed to a policy.""" ;
    rdfs:range :Role ;
    rdfs:domain :Policy ;
    rdfs:isDefinedBy <> .

:forPolicy a rdf:Property ;
    rdfs:label "For Policy" ;
    rdfs:comment """Which policy a policy statement relates to.""" ;
    rdfs:range :Policy ;
    rdfs:domain :PolicyStatement ;
    rdfs:isDefinedBy <> .

:NegotiatedPolicy a rdfs:Class ;
    rdfs:label "Negotiated Policy" ;
    rdfs:comment """A policy that is the result of negotiation.""" ;
    rdfs:subClassOf :Policy ;
    rdfs:isDefinedBy <> .

:PolicyStatement a rdfs:Class ;
    rdfs:label "Policy Statement" ;
    rdfs:comment """Part of a policy dealing with one resource.""" ;
    rdfs:subClassOf :Policy ;
    rdfs:isDefinedBy <> .

:Component a rdfs:Class ;
    rdfs:label "Component" ;
    rdfs:comment """Part of a policy statement dealing with its rule""" ;
    rdfs:subClassOf :PolicyStatement ;
    rdfs:isDefinedBy <> .

:hasComponent a rdf:Property ;
    rdfs:label "Has Component" ;
    rdfs:comment """Connecting statment to component.""" ;
    rdfs:range :Component ;
    rdfs:domain :PolicyStatement ;
    rdfs:isDefinedBy <> .

:withPredicate a rdf:Property ;
    rdfs:label "With Predicate" ;
    rdfs:comment """Connecting a component to whatever rule component
```

```
  it describes."""  ;
    rdfs:range rdf:Property ;
    rdfs:domain :Component ;
    rdfs:isDefinedBy <> .

:withRange a rdf:Property ;
    rdfs:label "With Range" ;
    rdfs:comment """Connecting a component to whatever rule component's
  range it describes."""  ;
    rdfs:range rdfs:Resource ;
    rdfs:domain :Component ;
    rdfs:isDefinedBy <> .

:Priority a rdfs:Class ;
    rdfs:label "Priority" ;
    rdfs:comment """A set of priorities."""  ;
    rdfs:subClassOf rdfs:Resource ;
    rdfs:isDefinedBy <> .

:hasPriority a rdf:Property ;
    rdfs:label "Intended Behavior" ;
    rdfs:comment """The priority the author gives to a policy staetment
  when it comes to negotiating policies."""  ;
    rdfs:domain :PolicyStatement ;
    rdfs:range :Priority ;
    rdfs:isDefinedBy <> .

:forResource a rdf:Property ;
    rdfs:label "For Resource" ;
    rdfs:comment """Relating which resource(s) a policy statement is
  about."""  ;
    rdfs:range rdfs:Resource ;
    rdfs:domain :PolicyStatement ;
    rdfs:isDefinedBy <> .

:withVisibility a rdf:Property ;
    rdfs:label "With Visibility" ;
    rdfs:comment """Relating a statement to whether it is hiding or
  revealing."""  ;
    rdfs:range :visibility ;
    rdfs:domain :PolicyStatement ;
    rdfs:isDefinedBy <> .

:Grammar a rdfs:Class ;
    rdfs:label "Grammar" ;
    rdfs:comment """A schema."""  ;
    rdfs:subClassOf rdfs:Resource ;
    rdfs:isDefinedBy <> .

:supports a rdf:Property ;
    rdfs:label "Supports" ;
```

```
    rdfs:comment """Allow the service to specify which namespace of
  grammars it understands""" ;
    rdfs:range :Grammar ;
    rdfs:domain :Policy ;
    rdfs:isDefinedBy <> .

:abbreviatedAs a rdf:Property ;
    rdfs:label "Abbreviated As" ;
    rdfs:comment """A helpful, brief name for a grammar; unnecessary, but
  more friendly looking than a generated name.""" ;
    rdfs:range rdfs:Literal ;
    rdfs:domain :Grammar ;
    rdfs:isDefinedBy <> .

:ruleSubject a rdf:Property ;
    rdfs:label "Rule Subject" ;
    rdfs:comment """This is a hack.  Because DAML lists don't sit well
  cwm yet, list processing of a component list to nicely generate rules
  with multiple predicates can't function.  So instead of generating rule
  subjects, just list the rule subject as the object of this.  Sad.""" ;
    rdfs:range rdfs:Resource ;
    rdfs:domain :PolicyStatement ;
    rdfs:isDefinedBy <> .

###
# Role classes and properties
###

:Role a rdfs:Class ;
    rdfs:label "Role" ;
    rdfs:comment """An abstraction over identities to detail what
  part an identity plays.""" ;
    rdfs:subClassOf rdfs:Resource ;
    rdfs:isDefinedBy <> .

:ClientRole a rdfs:Class ;
    rdfs:label "Client Role" ;
    rdfs:comment """Roles take on by clients""" ;
    rdfs:subClassOf :Role ;
    rdfs:isDefinedBy <> .

:ServerRole a rdfs:Class ;
    rdfs:label "Server Role" ;
    rdfs:comment """Roles taken on by a service""" ;
    rdfs:subClassOf :Role ;
    rdfs:isDefinedBy <> .

:hasRole a rdf:Property ;
    rdfs:label "Has Role" ;
    rdfs:comment """Relating roles to identities""" ;
    rdfs:domain acl:Identity ;
```

```
    rdfs:range :Role ;
    rdfs:isDefinedBy <> .

:Author a :ServerRole ;
    rdfs:label "Author" ;
    rdfs:comment """The author of a document or service.""" ;
    rdfs:isDefinedBy <> .

:PolicyAuthority a :ServerRole ;
    rdfs:label "Policy Authority" ;
    rdfs:comment """The role with the final word about policies for a
  service.""" ;
    rdfs:isDefinedBy <> .

:Anonymous a :ClientRole ;
    rdfs:label "Anonymous" ;
    rdfs:comment """The anonymous, unknown client using a service.""" ;
    rdfs:isDefinedBy <> .

:Requester a :ClientRole ;
    rdfs:lable "Requester" ;
    rdfs:comment """The role making a request.""" ;
    rdfs:isDefinedBy <> .

###
# Priorities (imperatives) for negotiation
# Taken from RFC 2119
###

:Must a :Priority ;
    rdfs:label "Must" ;
    rdfs:comment """A policy statement that is an absolute
  requirement.""" ;
    rdfs:isDefinedBy <> .
:Required = :Must ;
    rdfs:label "Required" ;
    rdfs:isDefinedBy <> .
:Shall = :Must ;
    rdfs:label "Shall" ;
    rdfs:isDefinedBy <> .

:Should a :Priority ;
    rdfs:label "Should" ;
    rdfs:comment """A policy statement that is strongly requested, but
  a service which has other overriding requirements may ignore it.""" ;
    rdfs:isDefinedBy <> .
:Recommended = :Should ;
    rdfs:label "Recommended" ;
    rdfs:isDefinedBy <> .

:May a :Priority ;
```

```
    rdfs:label "May" ;
    rdfs:comment """A policy statement that can be followed or
  ignored.""" ;
    rdfs:isDefinedBy <> .
:Optional = :May ;
    rdfs:label "Optional" ;
    rdfs:isDefinedBy <> .

:hasWarning a rdf:Property ;
    rdfs:label "Has Warning" ;
    rdfs:comment """Flags policy statments as not having any nicely
  resolvable conflicts with someobdy else's statements.""" ;
    rdfs:range rdfs:Literal ;
    rdfs:domain :PolicyStatement ;
    rdfs:isDefinedBy <> .

:warnAuthor a rdf:Property ;
    rdfs:label "Warn Author" ;
    rdfs:comment """Which policy author to flag with a warning.""" ;
    rdfs:range :Role ;
    rdfs:domain :PolicyStatement ;
    rdfs:isDefinedBy <> .

###
# Data element visibility
###

:visibility a rdf:Property ;
    rdfs:label "Visibility" ;
    rdfs:comment """Group the visibilities into one class.""" ;
    rdfs:subClassOf rdfs:Resource ;
    rdfs:isDefinedBy <> .

:hiddenTo a rdf:Property ;
    rdfs:label "Hidden To" ;
    rdfs:comment """A resource is hidden to a role.""" ;
    rdfs:range :Role ;
    rdfs:domain :PolicyStatement ;
    rdfs:subPropertyOf :visibility ;
    rdfs:isDefinedBy <> .

:visibleTo a rdf:Property ;
    rdfs:label "Visible To" ;
    rdfs:comment """A resource is visible to a role.""" ;
    rdfs:range :Role ;
    rdfs:domain :PolicyStatement ;
    rdfs:subPropertyOf :visibility ;
    rdfs:isDefinedBy <> .

###
# Core of identity properties, partly taken from Eric Prud'hommeaux's
```

```
# ACL namespace
###

:Identity a rdfs:Class ;
    rdfs:label "Identity" ;
    = acl:Identity ;
    rdfs:isDefinedBy acl: .

:Group a rdfs:Class ;
    rdfs:label "Group" ;
    = acl:Group ;
    rdfs:isDefinedBy acl: .

:memberOf a rdf:Property ;
    rdfs:label "Member Of" ;
    = acl:memberOf ;
    rdfs:range :Group ;
    rdfs:domain :Identity ;
    rdfs:isDefinedBy acl: .

:Meeting a rdfs:Class ;
    rdfs:label "Meeting" ;
    rdfs:comment """Class of meetings""" ;
    rdfs:subClassOf rdfs:Resource ;
    rdfs:isDefinedBy <> .

:attended a rdf:Property ;
    rdfs:label "Attended" ;
    rdfs:comment """Predicate indicating a user was present at a
  meeting.""" ;
    rdfs:range :Meeting ;
    rdfs:domain :Identity ;
    rdfs:isDefinedBy <> .

:Occupation a rdfs:Class ;
    rdfs:label "Occupation" ;
    rdfs:comment """Class of jobs""" ;
    rdfs:subClassOf rdfs:Resource ;
    rdfs:isDefinedBy <> .

:employedAs a rdf:Property ;
    rdfs:label "Employed As" ;
    rdfs:comment """Predicate indicating a user's occupation""" ;
    rdfs:range :Occupation ;
    rdfs:domain :Identity ;
    rdfs:isDefinedBy <> .

:hasUsername a rdf:Property ;
    rdfs:label "Has Username" ;
    rdfs:comment """Predicate indicating a user's username""" ;
    rdfs:range rdfs:Literal ;
```

```
    rdfs:domain :Identity ;
    rdfs:isDefinedBy <> .
```

Rule Generation Definition Rules

```
# $Id: pedal-generate-rules.n3,v 1.3 2002/08/18 17:52:40 ryanlee Exp $

@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix log:    <http://www.w3.org/2000/10/swap/log#> .
@prefix dc:     <http://www.purl.org/dc/elements/1.1/> .
@prefix pedal:  <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix daml:   <http://www.daml.org/2001/03/daml+oil#> .
@prefix ont:   <http://www.daml.org/2000/10/daml-ont#> .
@prefix :       <#> .

# Generate access rules based on the policy
{{ :x a pedal:PolicyStatement .
   :x pedal:forResource :y .
   :x pedal:withVisibility :z .
   :x pedal:ruleSubject :p .
   :p log:includes { :a :b :c . } . } log:implies
 { {  :p log:implies
    { :y :z :a . }} a pedal:AccessRule ;
                         a log:Truth ;
                         log:forAll :a . }} a log:Truth ;
    log:forAll :x, :y, :z, :p, :a, :b, :c .

# ENDS
```

Access Definition Rules

```
# $Id: pedal-access-rules.n3,v 1.1 2002/08/17 01:36:26 ryanlee Exp $

@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix log:    <http://www.w3.org/2000/10/swap/log#> .
@prefix acl:    <http://www.w3.org/2001/02/acls/ns#> .
@prefix dc:     <http://www.purl.org/dc/elements/1.1/> .
@prefix pedal:  <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix :       <#> .

<> dc:title "Personal Data Access Logic (PEDAL) Schema" ;
    dc:creator "Ryan Lee" ;
    dc:date "2001-12-20" ;
    dc:language "en" ;
    dc:publisher "The World Wide Web Consortium (W3C)" .

# Rules to apply to things generated by the rule generator
{{ :x pedal:visibleTo :y .
   :a :x :b . } log:implies
```

```
 { :a :x :b . }} a log:Truth ;
    log:forAll :x, :y, :a, :b .

{{ :x pedal:hiddenTo :y .
   :a :x :b . } log:implies
 { :x a log:Chaff }} a log:Truth ;
    log:forAll :x, :y, :a, :b .

# ENDS
```

Negotiation Rules

```
# $Id: pedal-negotiate-rules.n3,v 1.3 2002/08/21 19:36:02 ryanlee Exp $

@prefix log:    <http://www.w3.org/2000/10/swap/log#> .
@prefix pedal:  <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix :       <#> .

# Note that declaring something a log:Chaff means any references to it
# are internally deleted from a store and externally not shown

{{ :x a pedal:Grammar . } log:implies
 { :x a log:Chaff . }}
    a log:Truth ;
    log:forAll :x .

{{ :x pedal:authoredBy pedal:Author . } log:implies
 { :x a pedal:NegotiatedPolicy . }}
    a log:Truth ;
    log:forAll :x .

{{ :x a pedal:PolicyStatement .
   :x a log:Chaff .
   :x pedal:hasComponent :y . } log:implies
 { :y a log:Chaff . }}
    a log:Truth ;
    log:forAll :x, :y .

# Negotiation rules

# PolicyAuthority overrules Author when they give :Must priority
# with differing visibility measures and withPredicate's
{{ :a pedal:authoredBy pedal:PolicyAuthority .
   :z pedal:authoredBy pedal:Author .
   :b pedal:forPolicy :a .
   :y pedal:forPolicy :z .
   :b pedal:forResource :m .
   :y pedal:forResource :m .
   :b pedal:withVisibility :l .
   :y pedal:withVisibility :n .
```

```
   :l log:notEqualTo :n .
   :b pedal:hasPriority pedal:Must .
   :y pedal:hasPriority pedal:Must .
   :b pedal:hasComponent :c .
   :c pedal:withPredicate :d .
   :y pedal:hasComponent :x .
   :x pedal:withPredicate :w .
   :w log:notEqualTo :d . } log:implies
 { :b pedal:forPolicy :z .
   :x a log:Chaff .
   :y a log:Chaff .
   :y pedal:hasWarning """Statement was superseded (both had priority
  Must for different visibility conclusions)""" .
   :y pedal:warnAuthor pedal:Author . }} a log:Truth ;
    log:forAll :a, :b, :c, :d, :l, :m, :n, :w, :x, :y, :z .

# ... or they match on everything except visibility for one statement
{{ :a pedal:authoredBy pedal:PolicyAuthority .
   :z pedal:authoredBy pedal:Author .
   :b pedal:forPolicy :a .
   :y pedal:forPolicy :z .
   :b pedal:forResource :m .
   :y pedal:forResource :m .
   :b pedal:withVisibility :l .
   :y pedal:withVisibility :n .
   :l log:notEqualTo :n .
   :b pedal:hasPriority pedal:Must .
   :y pedal:hasPriority pedal:Must .
   :b pedal:hasComponent :c .
   :c pedal:withPredicate :d .
   :c pedal:withRange :e .
   :y pedal:hasComponent :x .
   :x pedal:withPredicate :d .
   :x pedal:withRange :e . } log:implies
 { :b pedal:forPolicy :z .
   :y a log:Chaff .
   :x a log:Chaff .
   :y pedal:hasWarning """Statement was superseded (both had priority
  Must for different visibility conclusions)""" .
   :y pedal:warnAuthor pedal:Author . }} a log:Truth ;
    log:forAll :a, :b, :c, :d, :e, :l, :m, :n, :v, :w, :x, :y, :z .

# ... or they completely match, so the PolicyAuthority one is tossed
{{ :a pedal:authoredBy pedal:PolicyAuthority .
   :z pedal:authoredBy pedal:Author .
   :b pedal:forPolicy :a .
   :y pedal:forPolicy :z .
   :b log:notEqualTo :y .
   :a log:notEqualTo :z .
   :b pedal:forResource :m .
   :y pedal:forResource :m .
```

```
   :b pedal:withVisibility :l .
   :y pedal:withVisibility :n .
   :l log:equalTo :n .
   :b pedal:hasComponent :c .
   :c pedal:withPredicate :d .
   :c pedal:withRange :e .
   :y pedal:hasComponent :x .
   :x pedal:withPredicate :d .
   :x pedal:withRange :e . } log:implies
 { :b a log:Chaff .
   :c a log:Chaff .
   :b pedal:hasWarning "Exact same statement as Author, removing." .
   :b pedal:warnAuthor pedal:PolicyAuthority . }} a log:Truth ;
    log:forAll :a, :b, :c, :d, :e, :l, :m, :n, :x, :y, :z .

# Anybody supersedes with :Must over anything the other person
# gives non-:Must priority to (again with the two different types of
# predicate conflicts - this one with equal everything)
{{ :a a pedal:Policy .
   :b a pedal:Policy .
   :o a pedal:NegotiatedPolicy .
   :a log:notEqualTo :b .
   :a pedal:authoredBy :m .
   :b pedal:authoredBy :n .
   :m log:notEqualTo :n .
   :c pedal:forPolicy :a .
   :d pedal:forPolicy :b .
   :c pedal:forResource :x .
   :d pedal:forResource :x .
   :c pedal:withVisibility :y .
   :d pedal:withVisibility :z .
   :y log:notEqualTo :z .
   :c pedal:hasPriority pedal:Must .
   :d pedal:hasPriority :p .
   :p log:notEqualTo pedal:Must .
   :c pedal:hasComponent :e .
   :d pedal:hasComponent :f .
   :e pedal:withPredicate :g .
   :f pedal:withPredicate :g .
   :e pedal:withRange :h .
   :f pedal:withRange :h . } log:implies
 { :c pedal:forPolicy :o .
   :d a log:Chaff .
   :f a log:Chaff .
   :d pedal:hasWarning """Statement was superseded (the other party
  conflicted over a visibility conclusion but gave higher priority)""" .
   :d pedal:warnAuthor :n . }} a log:Truth ;
    log:forAll :a, :b, :c, :d, :e, :f, :g, :h, :m, :n, :o, :p, :x, :y, :z .

# and this one with different predicates
{{ :a a pedal:Policy .
```

```
   :b a pedal:Policy .
   :o a pedal:NegotiatedPolicy .
   :a log:notEqualTo :b .
   :a pedal:authoredBy :m .
   :b pedal:authoredBy :n .
   :m log:notEqualTo :n .
   :c pedal:forPolicy :a .
   :d pedal:forPolicy :b .
   :c pedal:forResource :x .
   :d pedal:forResource :x .
   :c pedal:withVisibility :y .
   :d pedal:withVisibility :z .
   :y log:notEqualTo :z .
   :c pedal:hasPriority pedal:Must .
   :d pedal:hasPriority :p .
   :p log:notEqualTo pedal:Must .
   :c pedal:hasComponent :e .
   :d pedal:hasComponent :f .
   :e pedal:withPredicate :g .
   :f pedal:withPredicate :h .
   :g log:notEqualTo :h . } log:implies
 { :c pedal:forPolicy :o .
   :d a log:Chaff .
   :f a log:Chaff .
   :d pedal:hasWarning """Statement was superseded (the other party
  conflicted over a visibility conclusion but gave higher priority)""" .
   :d pedal:warnAuthor :n . }} a log:Truth ;
    log:forAll :a, :b, :c, :d, :e, :f, :g, :h, :m, :n, :o, :p, :x, :y, :z .

# The policy authority always wins in a clash (like the dealer)
{{ :a pedal:authoredBy pedal:PolicyAuthority .
   :b pedal:authoredBy pedal:Author .
   :o a pedal:NegotiatedPolicy .
   :a log:notEqualTo :b .
   :c pedal:forPolicy :a .
   :d pedal:forPolicy :b .
   :c pedal:forResource :x .
   :d pedal:forResource :x .
   :c pedal:withVisibility :y .
   :d pedal:withVisibility :z .
   :y log:notEqualTo :z .
   :c pedal:hasPriority :p .
   :d pedal:hasPriority :p .
   :p log:notEqualTo pedal:Must .
   :c pedal:hasComponent :e .
   :d pedal:hasComponent :f .
   :e pedal:withPredicate :g .
   :f pedal:withPredicate :g .
   :e pedal:withRange :h .
   :f pedal:withRange :h . } log:implies
 { :d a log:Chaff .
```

```
   :f a log:Chaff .
   :d pedal:hasWarning "Statement was superseded (Matching priorities)" .
   :d pedal:warnAuthor pedal:Author . }} a log:Truth ;
    log:forAll :a, :b, :c, :d, :e, :f, :g, :h, :o, :p, :x, :y, :z .

# other (differing predicates)
{{ :a pedal:authoredBy pedal:PolicyAuthority .
   :b pedal:authoredBy pedal:Author .
   :o a pedal:NegotiatedPolicy .
   :a log:notEqualTo :b .
   :a pedal:authoredBy :m .
   :b pedal:authoredBy :n .
   :m log:notEqualTo :n .
   :c pedal:forPolicy :a .
   :d pedal:forPolicy :b .
   :c pedal:forResource :x .
   :d pedal:forResource :x .
   :c pedal:withVisibility :y .
   :d pedal:withVisibility :z .
   :y log:notEqualTo :z .
   :c pedal:hasPriority :p .
   :d pedal:hasPriority :p .
   :p log:notEqualTo pedal:Must .
   :c pedal:hasComponent :e .
   :d pedal:hasComponent :f .
   :e pedal:withPredicate :g .
   :f pedal:withPredicate :h .
   :g log:notEqualTo :h . } log:implies
 { :d a log:Chaff .
   :f a log:Chaff .
   :d pedal:hasWarning "Statement was superseded (Matching priorities)" .
   :d pedal:warnAuthor :n . }} a log:Truth ;
    log:forAll :a, :b, :c, :d, :e, :f, :g, :h, :m, :n, :o, :p, :x, :y, :z .

# Anybody supersedes anybody with :Should
# over anything non-:Must and non-:Should
{{ :a a pedal:Policy .
   :b a pedal:Policy .
   :o a pedal:NegotiatedPolicy .
   :a log:notEqualTo :b .
   :a pedal:authoredBy :m .
   :b pedal:authoredBy :n .
   :m log:notEqualTo :n .
   :c pedal:forPolicy :a .
   :d pedal:forPolicy :b .
   :c pedal:forResource :x .
   :d pedal:forResource :x .
   :c pedal:withVisibility :y .
   :d pedal:withVisibility :z .
   :y log:notEqualTo :z .
   :c pedal:hasPriority pedal:Should .
```

```
   :d pedal:hasPriority :p .
   :p log:notEqualTo pedal:Must, pedal:Should .
   :c pedal:hasComponent :e .
   :d pedal:hasComponent :f .
   :e pedal:withPredicate :g .
   :f pedal:withPredicate :g .
   :e pedal:withRange :h .
   :f pedal:withRange :h . } log:implies
 { :c pedal:forPolicy :o .
   :d a log:Chaff .
   :f a log:Chaff .
   :d pedal:hasWarning """Statement was superseded (the other party gave
  a higher priority to their rules)""" .
   :d pedal:warnAuthor :n . }} a log:Truth ;
    log:forAll :a, :b, :c, :d, :e, :f, :g, :h, :m, :n, :o, :p, :x, :y, :z .

{{ :a a pedal:Policy .
   :b a pedal:Policy .
   :o a pedal:NegotiatedPolicy .
   :a log:notEqualTo :b .
   :a pedal:authoredBy :m .
   :b pedal:authoredBy :n .
   :m log:notEqualTo :n .
   :c pedal:forPolicy :a .
   :d pedal:forPolicy :b .
   :c pedal:forResource :x .
   :d pedal:forResource :x .
   :c pedal:withVisibility :y .
   :d pedal:withVisibility :z .
   :y log:notEqualTo :z .
   :c pedal:hasPriority pedal:Should .
   :d pedal:hasPriority :p .
   :p log:notEqualTo pedal:Must, pedal:Should .
   :c pedal:hasComponent :e .
   :d pedal:hasComponent :f .
   :e pedal:withPredicate :g .
   :f pedal:withPredicate :h . } log:implies
 { :c pedal:forPolicy :o .
   :d a log:Chaff .
   :f a log:Chaff .
   :d pedal:hasWarning """Statement was superseded (the other party gave
  a higher priority to their rules)""" .
   :d pedal:warnAuthor :n . }} a log:Truth ;
    log:forAll :a, :b, :c, :d, :e, :f, :g, :h, :m, :n, :o, :p, :x, :y, :z .

# These rules are cwm -think'd, so get rid of them with -purge
# when -think'ing is done; also rids other policy's and useless policy
# statements
log:forAll a log:Chaff .
log:Truth a log:Chaff .
```

```
# ENDS
```

Negotiation Cleanup Rules

```
# $Id: pedal-cleanup-negotiate-rules.n3,v 1.1 2002/08/21 15:48:32 ryanlee Exp $

@prefix log:    <http://www.w3.org/2000/10/swap/log#> .
@prefix pedal:  <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix :       <#> .

# Cleaning up after negotiation rules
# Get rid of the policy authority's policy ID
{{ :x pedal:authoredBy pedal:PolicyAuthority .
   :y a pedal:NegotiatedPolicy . } log:implies
 { :x a log:Chaff .
   :y pedal:contributedBy pedal:PolicyAuthority . }}
    a log:Truth ;
    log:forAll :x , :y ;
    a log:Chaff .

# Statements which didn't conflict with anything should be merged with
# the negotiated policy, which should be the only policy left.
{{ :x a pedal:Policy .
   :y a pedal:PolicyStatement . } log:implies
 { :y pedal:forPolicy :x . }}
    a log:Truth ;
    log:forAll :x, :y ;
    a log:Chaff .

# ENDS
```

11.2 Advanced Protection Test

Advanced Metadata

```
# simple-metadata
@prefix dc:      <http://www.purl.org/dc/elements/1.1/> .
@prefix doc:     <http://www.example.org/doc#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

doc: dc:title "A Complex Test" ;
        dc:creator doc:creator ;
        dc:date "2002-08-22" ;
        dc:language "en" ;
        dc:publisher doc:publisher .

doc:creator contact:fullName "Ryan Lee" ;
    contact:emailAddress <mailto:ryanlee@mit.edu> ;
    <http://www.w3.org/2002/01/p3prdfv1#user.employer> doc:publisher .
```

```
doc:publisher <http://www.w3.org/2002/01/p3prdfv1#business.name> "Examples" ;
    <http://www.w3.org/2002/01/p3prdfv1#business.contact-info.online.uri> <http:
//www.example.org> .

# ENDS
```

Advanced Policy

```
# complex-policy
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix dc:      <http://www.purl.org/dc/elements/1.1/> .
@prefix p3p:     <http://www.w3.org/2002/01/p3prdfv1#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
@prefix foaf:    <http://xmlns.com/foaf/0.1/> .

here:SimplePolicy a :Policy ;
    :policyName "Complex Metadata Policy" ;
    :forMetadataOf <http://www.example.org/doc#> ;
    :authoredBy :Author .

# Policy statements
[ a :PolicyStatement ;
    :forPolicy here:SimplePolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :forResource dc:date ;
    :forResource dc:language ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone :hasRole :Anonymous . } ;
    :hasComponent [ a :Component ;
                    :withPredicate :hasRole ;
                    :withRange :Anonymous ] ] .

[ a :PolicyStatement ;
    :forPolicy here:SimplePolicy ;
    :hasPriority :Must ;
    :forResource dc:publisher ;
    :forResource <http://www.w3.org/2002/01/p3prdfv1#business.name> ;
    :forResource <http://www.w3.org/2002/01/p3prdfv1#business.contact-info.onlin
e.uri> ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone <http://www.w3.org/2002/01/p3prdfv1#user.employer> "
Examples" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate <http://www.w3.org/2002/01/p3prdfv1#user.empl
oyer> ;
                    :withRange "Examples"  ] ] .
```

```
[ a :PolicyStatement ;
    :forPolicy here:SimplePolicy ;
    :hasPriority :Must ;
    :forResource dc:creator ;
    :forResource contact:fullName ;
    :forResource contact:emailAddress ;
    :forResource <http://www.w3.org/2002/01/p3prdfv1#user.employer> ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone <http://www.w3.org/2002/01/p3prdfv1#user.employer> "
Examples" ; <http://www.w3.org/2002/01/p3prdfv1#user.department> "Example Writer
s" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate <http://www.w3.org/2002/01/p3prdfv1#user.empl
oyer> ;
                    :withRange "Examples" ] ;
    :hasComponent [ a :Component ;
                    :withPredicate <http://www.w3.org/2002/01/p3prdfv1#user.depa
rtment> ;
                    :withRange "Example Writers" ] ] .

[ a :PolicyStatement ;
    :forPolicy here:SimplePolicy ;
    :hasPriority :Must ;
    :forResource contact:fullName ;
    :forResource contact:emailAddress ;
    :forResource <http://www.w3.org/2002/01/p3prdfv1#user.employer> ;
    :withVisibility :hiddenTo ;
    :ruleSubject { :someone contact:fullName "Ben Bitdiddle" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:fullName ;
                    :withRange "Ben Bitdiddle" ] ] .

# ENDS
```

Inputs and Corresponding Outputs

```
# complex-input-1
@prefix :     <http://www.w3.org/2002/01/pedal/pedal#> .

:Requester :hasRole :Anonymous .
```

```
# output-1
@prefix : <http://www.example.org/doc#> .

:    <http://www.purl.org/dc/elements/1.1/date> "2002-08-22";
    <http://www.purl.org/dc/elements/1.1/language> "en";
    <http://www.purl.org/dc/elements/1.1/title> "A Complex Test" .
```

```
# complex-input-2
@prefix :     <http://www.w3.org/2002/01/pedal/pedal#> .
```

```
:Requester :hasRole :Anonymous .
:Requester <http://www.w3.org/2002/01/p3prdfv1#user.employer> "Counterexamples"
.

@prefix : <http://www.example.org/doc#> .

: <http://www.purl.org/dc/elements/1.1/date> "2002-08-22";
    <http://www.purl.org/dc/elements/1.1/language> "en";
    <http://www.purl.org/dc/elements/1.1/title> "A Complex Test" .

# complex-input-3
@prefix :     <http://www.w3.org/2002/01/pedal/pedal#> .

:Requester :hasRole :Anonymous .
:Requester <http://www.w3.org/2002/01/p3prdfv1#user.employer> "Examples" .

# output-3
@prefix : <http://www.example.org/doc#> .

: <http://www.purl.org/dc/elements/1.1/date> "2002-08-22";
    <http://www.purl.org/dc/elements/1.1/language> "en";
    <http://www.purl.org/dc/elements/1.1/publisher> :publisher;
    <http://www.purl.org/dc/elements/1.1/title> "A Complex Test" .

:publisher <http://www.w3.org/2002/01/p3prdfv1#business.contact-info.online.uri>
 <http://www.example.org>;
    <http://www.w3.org/2002/01/p3prdfv1#business.name> "Examples" .

# complex-input-4
@prefix :     <http://www.w3.org/2002/01/pedal/pedal#> .

:Requester :hasRole :Anonymous .
:Requester <http://www.w3.org/2002/01/p3prdfv1#user.employer> "Examples" .
:Requester <http://www.w3.org/2002/01/p3prdfv1#user.department> "Example Readers
" .

# output-4
@prefix : <http://www.example.org/doc#> .

: <http://www.purl.org/dc/elements/1.1/date> "2002-08-22";
    <http://www.purl.org/dc/elements/1.1/language> "en";
    <http://www.purl.org/dc/elements/1.1/publisher> :publisher;
    <http://www.purl.org/dc/elements/1.1/title> "A Complex Test" .

:publisher <http://www.w3.org/2002/01/p3prdfv1#business.contact-info.online.uri>
 <http://www.example.org>;
    <http://www.w3.org/2002/01/p3prdfv1#business.name> "Examples" .

# complex-input-5
@prefix :     <http://www.w3.org/2002/01/pedal/pedal#> .
```

```
:Requester :hasRole :Anonymous .
:Requester <http://www.w3.org/2002/01/p3prdfv1#user.employer> "Examples" .
:Requester <http://www.w3.org/2002/01/p3prdfv1#user.department> "Example Writers
" .

# output-5
@prefix : <http://www.example.org/doc#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

: <http://www.purl.org/dc/elements/1.1/creator> :creator;
    <http://www.purl.org/dc/elements/1.1/date> "2002-08-22";
    <http://www.purl.org/dc/elements/1.1/language> "en";
    <http://www.purl.org/dc/elements/1.1/publisher> :publisher;
    <http://www.purl.org/dc/elements/1.1/title> "A Complex Test" .

:creator contact:emailAddress <mailto:ryanlee@mit.edu>;
    contact:fullName "Ryan Lee";
    <http://www.w3.org/2002/01/p3prdfv1#user.employer> :publisher .

:publisher <http://www.w3.org/2002/01/p3prdfv1#business.contact-info.online.uri>
 <http://www.example.org>;
    <http://www.w3.org/2002/01/p3prdfv1#business.name> "Examples" .

# complex-input-6
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

:Requester :hasRole :Anonymous .
:Requester <http://www.w3.org/2002/01/p3prdfv1#user.employer> "Examples" .
:Requester <http://www.w3.org/2002/01/p3prdfv1#user.department> "Example Writers
" .
:Requester contact:fullName "Ben Bitdiddle" .

# output-6
@prefix : <http://www.example.org/doc#> .

: <http://www.purl.org/dc/elements/1.1/creator> :creator;
    <http://www.purl.org/dc/elements/1.1/date> "2002-08-22";
    <http://www.purl.org/dc/elements/1.1/language> "en";
    <http://www.purl.org/dc/elements/1.1/publisher> :publisher;
    <http://www.purl.org/dc/elements/1.1/title> "A Complex Test" .

:publisher <http://www.w3.org/2002/01/p3prdfv1#business.contact-info.online.uri>
 <http://www.example.org>;
    <http://www.w3.org/2002/01/p3prdfv1#business.name> "Examples" .
```

11.3 Advanced Negotiation Test

```
# input-1-1
```

```
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:MyDocumentPolicy a :Policy ;
    :authoredBy :Author .

[ a :PolicyStatement ;
    :forPolicy here:MyDocumentPolicy ;
    :hasPriority :Should ;
    :forResource dc:title ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone contact:familyName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:familyName ;
                    :withRange "Allen" ] ] .

# input-1-2
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:DocumentPolicy a :Policy ;
    :authoredBy :PolicyAuthority .

[ a :PolicyStatement ;
    :forPolicy here:DocumentPolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone contact:familyName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:familyName ;
                    :withRange "Allen" ] ] .

# out-1
     @prefix : <http://www.w3.org/2002/01/pedal/pedal#> .
     @prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
     @prefix here: <input-1-1#> .
     @prefix hereg1: <input-1-2#> .
     @prefix log: <http://www.w3.org/2000/10/swap/log#> .

    here:MyDocumentPolicy     a :NegotiatedPolicy,
                :Policy;
          :authoredBy :Author;
          :contributedBy :PolicyAuthority .
       [      a :PolicyStatement;
              :forPolicy here:MyDocumentPolicy;
              :forResource <http://www.purl.org/dc/elements/1.1/title>;
```

```
             :hasComponent  [
                 a :Component;
                 :withPredicate contact:familyName;
                 :withRange "Allen" ];
             :hasPriority :Should;
             :ruleSubject {:someone     contact:familyName "Allen" .
            };
             :withVisibility :visibleTo ].

# input-2-1
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:MyDocumentPolicy a :Policy ;
    :authoredBy :Author .

[ a :PolicyStatement ;
    :forPolicy here:MyDocumentPolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone contact:familyName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:familyName ;
                    :withRange "Allen" ] ] .

# input-2-2
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:DocumentPolicy a :Policy ;
    :authoredBy :PolicyAuthority .

[ a :PolicyStatement ;
    :forPolicy here:DocumentPolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone contact:familyName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:familyName ;
                    :withRange "Allen" ] ] .

# out-2
     @prefix : <http://www.w3.org/2002/01/pedal/pedal#> .
     @prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
     @prefix here: <input-2-1#> .
```

```
     @prefix hereg1: <input-2-2#> .
     @prefix log: <http://www.w3.org/2000/10/swap/log#> .

    here:MyDocumentPolicy     a :NegotiatedPolicy,
                :Policy;
          :authoredBy :Author;
          :contributedBy :PolicyAuthority .
       [      a :PolicyStatement;
              :forPolicy here:MyDocumentPolicy;
              :forResource <http://www.purl.org/dc/elements/1.1/title>;
              :hasComponent  [
                  a :Component;
                  :withPredicate contact:familyName;
                  :withRange "Allen" ];
              :hasPriority :Must;
              :ruleSubject {:someone     contact:familyName "Allen" .
             };
              :withVisibility :visibleTo ].

# input-3-1
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:MyDocumentPolicy a :Policy ;
    :authoredBy :Author .

[ a :PolicyStatement ;
    :forPolicy here:MyDocumentPolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone contact:familyName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:familyName ;
                    :withRange "Allen" ] ] .

# input-3-2
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:DocumentPolicy a :Policy ;
    :authoredBy :PolicyAuthority .

[ a :PolicyStatement ;
    :forPolicy here:DocumentPolicy ;
    :hasPriority :Should ;
    :forResource dc:title ;
```

```
    :withVisibility :visibleTo ;
    :ruleSubject { :someone contact:familyName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:familyName ;
                    :withRange "Allen" ] ] .

# out-3
     @prefix : <http://www.w3.org/2002/01/pedal/pedal#> .
     @prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
     @prefix here: <input-3-1#> .
     @prefix hereg1: <input-3-2#> .
     @prefix log: <http://www.w3.org/2000/10/swap/log#> .

    here:MyDocumentPolicy     a :NegotiatedPolicy,
                :Policy;
          :authoredBy :Author;
          :contributedBy :PolicyAuthority .
       [      a :PolicyStatement;
              :forPolicy here:MyDocumentPolicy;
              :forResource <http://www.purl.org/dc/elements/1.1/title>;
              :hasComponent  [
                  a :Component;
                  :withPredicate contact:familyName;
                  :withRange "Allen" ];
              :hasPriority :Must;
              :ruleSubject {:someone     contact:familyName "Allen" .
             };
              :withVisibility :visibleTo ].

# input-4-1
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:MyDocumentPolicy a :Policy ;
    :authoredBy :Author .

[ a :PolicyStatement ;
    :forPolicy here:MyDocumentPolicy ;
    :hasPriority :Should ;
    :forResource dc:title ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone contact:familyName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:familyName ;
                    :withRange "Allen" ] ] .

# input-4-2
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
```

```
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:DocumentPolicy a :Policy ;
    :authoredBy :PolicyAuthority .

[ a :PolicyStatement ;
    :forPolicy here:DocumentPolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :withVisibility :hiddenTo ;
    :ruleSubject { :someone contact:familyName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:familyName ;
                    :withRange "Allen" ] ] .

# out-4
     @prefix : <http://www.w3.org/2002/01/pedal/pedal#> .
     @prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
     @prefix here: <input-4-1#> .
     @prefix hereg1: <input-4-2#> .
     @prefix log: <http://www.w3.org/2000/10/swap/log#> .

    here:MyDocumentPolicy     a :NegotiatedPolicy,
                :Policy;
            :authoredBy :Author;
            :contributedBy :PolicyAuthority .
       [      a :PolicyStatement;
              :forPolicy here:MyDocumentPolicy;
              :forResource <http://www.purl.org/dc/elements/1.1/title>;
              :hasComponent  [
                  a :Component;
                  :withPredicate contact:familyName;
                  :withRange "Allen" ];
              :hasPriority :Must;
              :ruleSubject {:someone     contact:familyName "Allen" .
             };
              :withVisibility :hiddenTo ].

# input-5-1
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:MyDocumentPolicy a :Policy ;
    :authoredBy :Author .

[ a :PolicyStatement ;
    :forPolicy here:MyDocumentPolicy ;
    :hasPriority :Must ;
```

```
    :forResource dc:title ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone contact:familyName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:familyName ;
                    :withRange "Allen" ] ] .

# input-5-2
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:DocumentPolicy a :Policy ;
    :authoredBy :PolicyAuthority .

[ a :PolicyStatement ;
    :forPolicy here:DocumentPolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :withVisibility :hiddenTo ;
    :ruleSubject { :someone contact:familyName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:familyName ;
                    :withRange "Allen" ] ] .

# out-5
     @prefix : <http://www.w3.org/2002/01/pedal/pedal#> .
     @prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
     @prefix here: <input-5-1#> .
     @prefix hereg1: <input-5-2#> .
     @prefix log: <http://www.w3.org/2000/10/swap/log#> .

    here:MyDocumentPolicy     a :NegotiatedPolicy,
                :Policy;
         :authoredBy :Author;
         :contributedBy :PolicyAuthority .
       [      a :PolicyStatement;
              :forPolicy here:MyDocumentPolicy;
              :forResource <http://www.purl.org/dc/elements/1.1/title>;
              :hasComponent  [
                  a :Component;
                  :withPredicate contact:familyName;
                  :withRange "Allen" ];
              :hasPriority :Must;
              :ruleSubject {:someone     contact:familyName "Allen" .
             };
              :withVisibility :hiddenTo ].

# input-6-1
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
```

```
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:MyDocumentPolicy a :Policy ;
    :authoredBy :Author .

[ a :PolicyStatement ;
    :forPolicy here:MyDocumentPolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone contact:familyName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:familyName ;
                    :withRange "Allen" ] ] .

# input-6-2
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:DocumentPolicy a :Policy ;
    :authoredBy :PolicyAuthority .

[ a :PolicyStatement ;
    :forPolicy here:DocumentPolicy ;
    :hasPriority :Should ;
    :forResource dc:title ;
    :withVisibility :hiddenTo ;
    :ruleSubject { :someone contact:familyName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:familyName ;
                    :withRange "Allen" ] ] .

# out-6
     @prefix : <http://www.w3.org/2002/01/pedal/pedal#> .
     @prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
     @prefix foaf: <http://xmlns.com/foaf/0.1/> .
     @prefix here: <input-6-1#> .
     @prefix hereg1: <input-6-2#> .

    here:MyDocumentPolicy     a :NegotiatedPolicy,
                :Policy;
          :authoredBy :Author;
          :contributedBy :PolicyAuthority .
       [      a :PolicyStatement;
              :forPolicy here:MyDocumentPolicy;
              :forResource <http://www.purl.org/dc/elements/1.1/title>;
              :hasComponent  [
```

```
                a :Component;
                :withPredicate contact:familyName;
                :withRange "Allen" ];
             :hasPriority :Must;
             :ruleSubject {:someone     contact:familyName "Allen" .
            };
             :withVisibility :visibleTo ].

# input-7-1
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:MyDocumentPolicy a :Policy ;
    :authoredBy :Author .

[ a :PolicyStatement ;
    :forPolicy here:MyDocumentPolicy ;
    :hasPriority :Should ;
    :forResource dc:title ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone contact:familyName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:familyName ;
                    :withRange "Allen" ] ] .

# input-7-2
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix p3p:     <http://www.w3.org/2002/01/p3prdfv1#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:DocumentPolicy a :Policy ;
    :authoredBy :PolicyAuthority .

[ a :PolicyStatement ;
    :forPolicy here:DocumentPolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :withVisibility :hiddenTo ;
    :ruleSubject { :someone contact:firstName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:firstName ;
                    :withRange "Allen" ] ] .
# out-7
     @prefix : <http://www.w3.org/2002/01/pedal/pedal#> .
     @prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
     @prefix here: <input-7-1#> .
     @prefix hereg1: <input-7-2#> .
```

```
     @prefix log: <http://www.w3.org/2000/10/swap/log#> .

    here:MyDocumentPolicy     a :NegotiatedPolicy,
                :Policy;
          :authoredBy :Author;
          :contributedBy :PolicyAuthority .
       [      a :PolicyStatement;
              :forPolicy here:MyDocumentPolicy;
              :forResource <http://www.purl.org/dc/elements/1.1/title>;
              :hasComponent  [
                  a :Component;
                  :withPredicate contact:firstName;
                  :withRange "Allen" ];
              :hasPriority :Must;
              :ruleSubject {:someone     contact:firstName "Allen" .
             };
              :withVisibility :hiddenTo ].

# input-8-1
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:MyDocumentPolicy a :Policy ;
    :authoredBy :Author .

[ a :PolicyStatement ;
    :forPolicy here:MyDocumentPolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone contact:familyName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:familyName ;
                    :withRange "Allen" ] ] .

# input-8-2
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:DocumentPolicy a :Policy ;
    :authoredBy :PolicyAuthority .

[ a :PolicyStatement ;
    :forPolicy here:DocumentPolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :withVisibility :hiddenTo ;
```

```
    :ruleSubject { :someone contact:firstName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:firstName ;
                    :withRange "Allen" ] ] .

# out-8
     @prefix : <http://www.w3.org/2002/01/pedal/pedal#> .
     @prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
     @prefix here: <input-8-1#> .
     @prefix hereg1: <input-8-2#> .
     @prefix log: <http://www.w3.org/2000/10/swap/log#> .

    here:MyDocumentPolicy     a :NegotiatedPolicy,
                :Policy;
         :authoredBy :Author;
         :contributedBy :PolicyAuthority .
       [       a :PolicyStatement;
               :forPolicy here:MyDocumentPolicy;
               :forResource <http://www.purl.org/dc/elements/1.1/title>;
               :hasComponent  [
                   a :Component;
                   :withPredicate contact:firstName;
                   :withRange "Allen" ];
               :hasPriority :Must;
               :ruleSubject {:someone     contact:firstName "Allen" .
              };
               :withVisibility :hiddenTo ].

# input-9-1
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:MyDocumentPolicy a :Policy ;
    :authoredBy :Author .

[ a :PolicyStatement ;
    :forPolicy here:MyDocumentPolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone contact:familyName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:familyName ;
                    :withRange "Allen" ] ] .

# input-9-2
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
```

```
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:DocumentPolicy a :Policy ;
    :authoredBy :PolicyAuthority .

[ a :PolicyStatement ;
    :forPolicy here:DocumentPolicy ;
    :hasPriority :Should ;
    :forResource dc:title ;
    :withVisibility :hiddenTo ;
    :ruleSubject { :someone contact:firstName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:firstName ;
                    :withRange "Allen" ] ] .

# out-9
     @prefix : <http://www.w3.org/2002/01/pedal/pedal#> .
     @prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
     @prefix here: <input-9-1#> .
     @prefix hereg1: <input-9-2#> .
     @prefix log: <http://www.w3.org/2000/10/swap/log#> .

    here:MyDocumentPolicy     a :NegotiatedPolicy,
                :Policy;
          :authoredBy :Author;
          :contributedBy :PolicyAuthority .
       [      a :PolicyStatement;
              :forPolicy here:MyDocumentPolicy;
              :forResource <http://www.purl.org/dc/elements/1.1/title>;
              :hasComponent  [
                  a :Component;
                  :withPredicate contact:familyName;
                  :withRange "Allen" ];
              :hasPriority :Must;
              :ruleSubject {:someone     contact:familyName "Allen" .
             };
              :withVisibility :visibleTo ].

# input-10-1
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:MyDocumentPolicy a :Policy ;
    :authoredBy :Author .

[ a :PolicyStatement ;
    :forPolicy here:MyDocumentPolicy ;
    :hasPriority :Should ;
    :forResource dc:title ;
```

```
    :withVisibility :visibleTo ;
    :ruleSubject { :someone contact:familyName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:familyName ;
                    :withRange "Allen" ] ] .

# input-10-2
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:DocumentPolicy a :Policy ;
    :authoredBy :PolicyAuthority .

[ a :PolicyStatement ;
    :forPolicy here:DocumentPolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :withVisibility :hiddenTo ;
    :ruleSubject { :someone contact:firstName "Bishop" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:firstName ;
                    :withRange "Bishop" ] ] .

# out-10
     @prefix : <http://www.w3.org/2002/01/pedal/pedal#> .
     @prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
     @prefix here: <input-10-1#> .
     @prefix hereg1: <input-10-2#> .
     @prefix log: <http://www.w3.org/2000/10/swap/log#> .

    here:MyDocumentPolicy     a :NegotiatedPolicy,
                :Policy;
         :authoredBy :Author;
         :contributedBy :PolicyAuthority .
       [      a :PolicyStatement;
              :forPolicy here:MyDocumentPolicy;
              :forResource <http://www.purl.org/dc/elements/1.1/title>;
              :hasComponent  [
                  a :Component;
                  :withPredicate contact:firstName;
                  :withRange "Bishop" ];
              :hasPriority :Must;
              :ruleSubject {:someone     contact:firstName "Bishop" .
             };
              :withVisibility :hiddenTo ].

# input-11-1
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
```

```
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:MyDocumentPolicy a :Policy ;
    :authoredBy :Author .

[ a :PolicyStatement ;
    :forPolicy here:MyDocumentPolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone contact:familyName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:familyName ;
                    :withRange "Allen" ] ] .

# input-11-2
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:DocumentPolicy a :Policy ;
    :authoredBy :PolicyAuthority .

[ a :PolicyStatement ;
    :forPolicy here:DocumentPolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :withVisibility :hiddenTo ;
    :ruleSubject { :someone contact:firstName "Bishop" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:firstName ;
                    :withRange "Bishop" ] ] .

# out-11
     @prefix : <http://www.w3.org/2002/01/pedal/pedal#> .
     @prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
     @prefix here: <input-11-1#> .
     @prefix hereg1: <input-11-2#> .
     @prefix log: <http://www.w3.org/2000/10/swap/log#> .

    here:MyDocumentPolicy     a :NegotiatedPolicy,
                :Policy;
          :authoredBy :Author;
          :contributedBy :PolicyAuthority .
      [       a :PolicyStatement;
              :forPolicy here:MyDocumentPolicy;
              :forResource <http://www.purl.org/dc/elements/1.1/title>;
              :hasComponent  [
                  a :Component;
```

```
                :withPredicate contact:firstName;
                :withRange "Bishop" ];
             :hasPriority :Must;
             :ruleSubject {:someone     contact:firstName "Bishop" .
            };
             :withVisibility :hiddenTo ].

# input-12-1
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:MyDocumentPolicy a :Policy ;
    :authoredBy :Author .

[ a :PolicyStatement ;
    :forPolicy here:MyDocumentPolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone contact:familyName "Allen" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:familyName ;
                    :withRange "Allen" ] ] .

# input-12-1
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .

here:DocumentPolicy a :Policy ;
    :authoredBy :PolicyAuthority .

[ a :PolicyStatement ;
    :forPolicy here:DocumentPolicy ;
    :hasPriority :Should ;
    :forResource dc:title ;
    :withVisibility :hiddenTo ;
    :ruleSubject { :someone contact:firstName "Bishop" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate contact:firstName ;
                    :withRange "Bishop" ] ] .

# out-12
     @prefix : <http://www.w3.org/2002/01/pedal/pedal#> .
     @prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
     @prefix here: <input-12-1#> .
     @prefix hereg1: <input-12-2#> .
```

```
    here:MyDocumentPolicy     a :NegotiatedPolicy,
                :Policy;
         :authoredBy :Author;
         :contributedBy :PolicyAuthority .
       [      a :PolicyStatement;
              :forPolicy here:MyDocumentPolicy;
              :forResource <http://www.purl.org/dc/elements/1.1/title>;
              :hasComponent  [
                  a :Component;
                  :withPredicate contact:familyName;
                  :withRange "Allen" ];
              :hasPriority :Must;
              :ruleSubject {:someone     contact:familyName "Allen" .
             };
              :withVisibility :visibleTo ].
```

11.4 Document Publication Service Policy

```
# $Id: meta-policy.n3.txt,v 1.1 2002/08/22 07:01:01 ryanlee Exp $

@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix dc:      <http://www.purl.org/dc/elements/1.1/> .
@prefix p3p:     <http://www.w3.org/2002/01/p3prdfv1#> .

here:DocumentPolicy a :Policy ;
    :policyName "Document Metadata Policy" ;
    :forMetadataOf <http://www.example.org/documents> ;
    :supports dc: ;
    :supports p3p: ;
    :authoredBy :PolicyAuthority .

dc: a :Grammar ;
    :abbreviatedAs "dc" .

p3p: a :Grammar ;
    :abbreviatedAs "p3p" .

# Policy statements
[ a :PolicyStatement ;
    :forPolicy here:DocumentPolicy ;
    :hasPriority :Must ;
    :forResource dc:title ;
    :forResource dc:language ;
    :forResource dc:publisher ;
    :forResource dc:date ;
    :forResource dc:format ;
    :forResource dc:identifier ;
```

```
    :forResource dc:contributor ;
    :forResource dc:relation ;
    :forResource dc:rights ;
    :forResource dc:source ;
    :forResource dc:subject ;
    :forResource dc:type ;
    :forResource dc:creator ;
    :forResource contact:fullName ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone :hasRole :Anonymous . } ;
    :hasComponent [ a :Component ;
                    :withPredicate :hasRole ;
                    :withRange :Anonymous ] ] .

[ a :PolicyStatement ;
    :forPolicy here:DocumentPolicy ;
    :hasPriority :Should ;
    :forResource contact:emailAddress ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone :memberOf "Member" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate :memberOf ;
                    :withRange user:Member ] ] .

[ a :PolicyStatement ;
    :forPolicy here:DocumentPolicy ;
    :hasPriority :Should ;
    :forResource foaf:phone ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone p3p:business-name "W3C" .
                   :someone :employedAs "Research Scientist" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate p3p:business-name ;
                    :withRange "W3C" ] ;
    :hasComponent [ a :Component ;
            :withPredicate :employedAs ;
            :withRange "Research Scientist" ] ] .
```

11.5 Contact Information

```
# $Id: contact-info.n3.txt,v 1.1 2002/08/22 06:58:20 ryanlee Exp $
# Schema for Personal Data Access Logic (PEDAL)

@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
@prefix :        <#> .

:RL contact:emailAddress <mailto:ryanlee@mit.edu> ;
    contact:publicHomePage <http://ryanlee.org/> ;
    contact:fullName "Ryan Lee" .
```

11.6 Contact Information Policies

```
# $Id: contact-policy.n3.txt,v 1.1 2002/08/22 06:58:20 ryanlee Exp $

@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix dc:      <http://www.purl.org/dc/elements/1.1/> .
@prefix p3p:     <http://www.w3.org/2002/01/p3prdfv1#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
@prefix foaf:    <http://xmlns.com/foaf/0.1/> .

here: dc:title "Contact Information Policy" ;
    dc:creator "Ryan Lee" ;
    dc:date "2002-08-15" ;
    dc:language "en" ;
    dc:publisher "The World Wide Web Consortium (W3C)" .

here:ContactPolicy a :Policy ;
    :policyName "Contact Information Policy" ;
    :forMetadataOf <http://www.example.org/ryanlee> ;
    :authoredBy :Author .

# Policy statements
[ a :PolicyStatement ;
    :forPolicy here:ContactPolicy ;
    :hasPriority :Must ;
    :forResource contact:emailAddress ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone :school-name "MIT" .
                :someone foaf:knows "Ryan Lee" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate p3p:school-name ;
                    :withValue "MIT" ] ;
    :hasComponent [ a :Component ;
            :withPredicate foaf:knows ;
            :withValue "Ryan Lee" ] ] .

[ a :PolicyStatement ;
    :forPolicy here:ContactPolicy ;
    :hasPriority :Must ;
    :forResource contact:emailAddress ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone <http://www.w3.org/2002/01/p3prdfv1#user.employer> "
W3C" . } ;
    :hasComponent [ a :Component ;
                    :withPredicate <http://www.w3.org/2002/01/p3prdfv1#user.empl
oyer> ;
                    :withValue "W3C" ] ] .
```

```
[ a :PolicyStatement ;
    :forPolicy :ContactPolicy ;
    :hasPriority here:Must ;
    :forResource contact:publicHomePage ;
    :forResource contact:fullName ;
    :withVisibility :visibleTo ;
    :ruleSubject { :someone :hasRole :Anonymous . } ;
    :hasComponent [ a :Component ;
                    :withPredicate :hasRole ;
                    :withValue :Anonymous ] ] .
```

11.7 Sample Generated PEDAL Policy

```
@prefix : <#> .
@prefix acl: <http://www.w3.org/2001/02/acls/ns#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
@prefix dc: <http://www.purl.org/dc/elements/1.1/> .
@prefix doc: <http://www.example.org/thesis.html> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix here: <> .
@prefix hereg1: <online:meta#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix p3p: <http://www.w3.org/2002/01/p3prdfv1#> .
@prefix pedal: <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

 [      a pedal:PolicyStatement;
        pedal:forPolicy <metaThesisMetadata>;
        pedal:forResource <http://www.purl.org/dc/elements/1.1/language>;
        pedal:hasComponent  [
            a pedal:Component;
            pedal:withPredicate pedal:hasRole;
            pedal:withRange pedal:Anonymous ];
        pedal:hasPriority pedal:Should;
        pedal:ruleSubject {pedal:someone     pedal:hasRole pedal:Anonymous
.
       };
        pedal:withVisibility pedal:visibleTo ].

 [      a pedal:PolicyStatement;
        pedal:forPolicy <metaThesisMetadata>;
        pedal:forResource <http://www.purl.org/dc/elements/1.1/date>;
        pedal:hasComponent  [
            a pedal:Component;
            pedal:withPredicate pedal:hasRole;
            pedal:withRange pedal:Anonymous ];
        pedal:hasPriority pedal:Must;
        pedal:ruleSubject {pedal:someone     pedal:hasRole pedal:Anonymous
.
       };
```

```
            pedal:withVisibility pedal:visibleTo ].

      [     a pedal:PolicyStatement;
            pedal:forPolicy <metaThesisMetadata>;
            pedal:forResource <http://www.purl.org/dc/elements/1.1/title>;
            pedal:hasComponent  [
                a pedal:Component;
                pedal:withPredicate pedal:hasRole;
                pedal:withRange pedal:Anonymous ];
            pedal:hasPriority pedal:Should;
            pedal:ruleSubject {pedal:someone     pedal:hasRole pedal:Anonymous
.
           };
            pedal:withVisibility pedal:visibleTo ].

      [     a pedal:PolicyStatement;
            pedal:forPolicy <metaThesisMetadata>;
            pedal:forResource <http://www.purl.org/dc/elements/1.1/publisher>;
            pedal:hasComponent  [
                a pedal:Component;
                pedal:withPredicate pedal:hasRole;
                pedal:withRange pedal:Anonymous ];
            pedal:hasPriority pedal:Should;
            pedal:ruleSubject {pedal:someone     pedal:hasRole pedal:Anonymous
.
           };
            pedal:withVisibility pedal:visibleTo ].

      [     a pedal:PolicyStatement;
            pedal:forPolicy <metaThesisMetadata>;
            pedal:forResource contact:emailAddress;
            pedal:hasComponent  [
                a pedal:Component;
                pedal:withPredicate <http://xmlns.com/foaf/0.1/knows>;
                pedal:withRange "Ryan Lee" ];
            pedal:hasPriority pedal:Must;
            pedal:ruleSubject {pedal:someone     <http://xmlns.com/foaf/0.1/kno
ws> "Ryan Lee" .
           };
            pedal:withVisibility pedal:visibleTo ].

      [     a pedal:PolicyStatement;
            pedal:forPolicy <metaThesisMetadata>;
            pedal:forResource <http://www.purl.org/dc/elements/1.1/creator>;
            pedal:hasComponent  [
                a pedal:Component;
                pedal:withPredicate pedal:hasRole;
                pedal:withRange pedal:Anonymous ];
            pedal:hasPriority pedal:Should;
            pedal:ruleSubject {pedal:someone     pedal:hasRole pedal:Anonymous
.
```

```
            };
             pedal:withVisibility pedal:visibleTo ].

      [      a pedal:PolicyStatement;
             pedal:forPolicy <metaThesisMetadata>;
             pedal:forResource contact:emailAddress;
             pedal:hasPriority pedal:Must;
             pedal:ruleSubject {pedal:someone     <http://www.w3.org/2002/01/p3p
rdfv1#user.employer> "W3C";
                 <http://xmlns.com/foaf/0.1/currentProject> "Semantic Web" .
            };
             pedal:withVisibility pedal:visibleTo ].

    <>     <http://www.purl.org/dc/elements/1.1/creator> "Peddler Generator";
         <http://www.purl.org/dc/elements/1.1/date> "8/22/2002";
         <http://www.purl.org/dc/elements/1.1/language> "en";
         <http://www.purl.org/dc/elements/1.1/title> "Thesis Metadata PEDAL Poli
cy" .

    <metaThesisMetadata>     a pedal:NegotiatedPolicy,
                pedal:Policy;
         pedal:authoredBy pedal:Author;
         pedal:contributedBy pedal:PolicyAuthority;
         pedal:forMetadataOf <http://www.example.org/thesis.html>;
         pedal:policyName "Thesis Metadata" .
       [      a pedal:PolicyStatement;
             pedal:forPolicy <metaThesisMetadata>;
             pedal:forResource <http://xmlns.com/foaf/0.1/phone>;
             pedal:hasComponent  [
                 a pedal:Component;
                 pedal:withPredicate p3p:business-name;
                 pedal:withRange "W3C" ],
                     [
                 a pedal:Component;
                 pedal:withPredicate pedal:employedAs;
                 pedal:withRange "Research Scientist" ];
             pedal:hasPriority pedal:Should;
             pedal:ruleSubject {pedal:someone     p3p:business-name "W3C";
                 pedal:employedAs "Research Scientist" .
            };
             pedal:withVisibility pedal:visibleTo ].
```

12 Appendix: Source Code

All of the code written for PEDAL is done in Python (see section 4.7 for language references).

12.1 Apache Configuration

The following exercept is taken from `/etc/httpd/conf/httpd.conf` from a standard Apache 1.3.23 installation using the Red Hat Package Manager on Red Hat 7.1. For any given HTTP server software, Bouncer requires something similar to `mod_rewrite` and `mod_proxy` to be enabled and configured in order to handle comma tools correctly.

Note `localhost` in the rewrite rule should be the location of the Bouncer CGI script; in this instance, it just happens to reside on the same machine.

```
LoadModule rewrite_module     modules/mod_rewrite.so
LoadModule proxy_module       modules/libproxy.so

AddModule mod_rewrite.c
AddModule mod_proxy.c

<IfModule mod_rewrite.c>
RewriteEngine On
RewriteOptions inherit
RewriteLog logs/rewrite_log
RewriteLogLevel 0

RewriteRule ^(.+),([^/]+)$  http://localhost/cgi-bin/bouncer?type=$2\&resource=
$1  [P,L,QSA]
</IfModule>
```

12.2 Bouncer SQL Data Model

```
-- PEDAL's Bouncer data model
-- $Id: pedal.sql,v 1.5 2002/08/21 01:09:43 ryanlee Exp $

create table resource_meta (
       resource_url          varchar(4000) not null,
       data             lztext not null,
       type             varchar(20) not null,
```

```
       constraint resource_meta_type_chk
                     check (type in ('meta','contact')),
       constraint resource_meta_pk primary key (resource_url, type)
);

create table meta_policy (
       resource_url         varchar(4000) not null,
       type             varchar(20) not null,
       data             lztext not null,
       constraint meta_policy_ref foreign key (resource_url, type) references re
source_meta(resource_url, type)
);

create table tool_policy (
       type             varchar(20) not null primary key,
       data             lztext not null
);
```

12.3 Peddler

Naming

```
#!/usr/bin/python2
# $Id: peddler,v 1.10 2002/08/20 23:59:05 ryanlee Exp $

import cgi, sys, os, urllib2
sys.path.insert(0, "/var/www/lib")
import htmllib, peddlerlib, bouncerlib
from htmllib import *

def main():
    # Appearance / content configuration
    title = "Peddler PEDAL Generate - Step 1 / 4"
    head = """
  <link rel="stylesheet" type="text/css" href="/style.css">"""

    # Local variables
    doc  = XHTMLDoc()
    form = cgi.FieldStorage()

    type = form.getvalue("type", None)
    resource = form.getvalue("resource", None)

    if type is None:
        doc.makeError("Missing input <code>type</code>, please correct\n")
        doc.serveDoc()
        return

    if resource is None:
        doc.makeError("Missing input <code>resource</code>, please correct\n")
```

```
        doc.serveDoc()
        return

    if not (type == "contact" or type == "meta"):
        doc.makeError("Invalid comma tool <code>type</code>: "+type+"\n")
        doc.serveDoc()
        return

    # Check if the URL is valid - see bouncer for why
    try:
        t = urllib2.urlopen("http://"+os.environ.get("HTTP_HOST")+resource)
    except urllib2.HTTPError, urllib2.HTTPError.code:
        stat = "%s" % urllib2.HTTPError.code
        if stat.find("404") > 0:
            doc.makeNotFound(resource)
            doc.serveDoc()
            return
        else:
            doc.makeError(stat)
            doc.serveDoc()
            return

    # Check if policy already exists, reject if it does
    b = bouncerlib.Bouncer()
    b.openDB()
    if b.checkResourceTypeInDB(resource, "pedal") > 0:
        b.closeDB()
        doc.makeError("Policy already exists, cannot generate new policy\n")
        doc.serveDoc()
        return
    b.closeDB()
    del(b)

    generateform = TableForm("/cgi-bin/peddler-2")
    generateform.addComponent("", "hidden", ["resource", resource])
    generateform.addComponent("", "hidden", ["type", type])
    generateform.addTitle("GENERATE PEDAL POLICY - STEP 1 / 4")
    generateform.setSubmit("Next Step")
    generateform.setReset("Clear Values")
    generateform.addLegend("Specify a name for the policy:")
    generateform.addLegend("Generating for URL <code>%s</code>, tool type <code>
%s</code>" % (resource, type))
    generateform.addComponent("NAME","text","policyname")
    generateform.addButtons()

    # Serve it up
    doc.makeDoc(title, generateform.getvalue(), extra = head)
    doc.serveDoc()

main()
```

Composing Rules

```
#!/usr/bin/python2
# $Id: peddler-2,v 1.5 2002/08/20 23:59:06 ryanlee Exp $

import cgi, sys, time
sys.path.insert(0, "/var/www/lib")
import htmllib, peddlerlib, bouncerlib, llyn, notation3
from htmllib import *
from types import ListType, StringType

def main():
    # Appearance config
    title = "Peddler PEDAL Generate - Step 2 / 4"
    head = """
  <link rel="stylesheet" type="text/css" href="/style.css">"""

    form    = cgi.FieldStorage()
    doc     = XHTMLDoc()

    # Form processing
    type = form.getvalue("type", None)
    resource = form.getvalue("resource", None)
    policyname = form.getvalue("policyname", None)

    if type is None:
        doc.makeError("Missing input <code>type</code>, please correct\n")
        doc.serveDoc()
        return

    if resource is None:
        doc.makeError("Missing input <code>resource</code>, please correct\n")
        doc.serveDoc()
        return

    if policyname is None:
        doc.makeError("Missing input <code>policyname</code>, please correct\n")
        doc.serveDoc()
        return

    # Generate form - first, use all known namespaces for predicates
    namespaces = []
    allatts = []
    for knownNS in peddlerlib.KNOWN_ABOUT_USERS:
        ns   = knownNS[peddlerlib.NAMESPACE]
        pre  = knownNS[peddlerlib.NICKNAME]
        atts = knownNS[peddlerlib.PROPERTIES]
        namespaces.append([pre, makeAnchor(ns)])
        preatts = [peddlerlib.prefixns(pre, att) for att in atts]
        allatts += preatts
    namespaces = [": ".join(el) for el in namespaces]
```

```
    # Then grab the metadata for the resource and figure out what
    # valid values can be selected
    metadataURI = "online:"+resource+"/"+type
    _metaURI = metadataURI+"/RUN"+`time.time()`
    b = bouncerlib.Bouncer()
    store = llyn.RDFStore(metaURI = _metaURI)
    b.openDB()
    metadata = b.getData(resource, type)
    b.closeDB()
    del(b)
    doctype = bouncerlib.guessType(metadata)
    if doctype is None:
        raise IOError
    elif doctype == "rdf":
        import sax2rdf, xml.sax._exceptions
        p = sax2rdf.RDFXMLParser(store, metadataURI)
        p.feed(metadata)
        del(p)
    elif doctype == "n3":
        p = notation3.SinkParser(store, metadataURI)
        p.startDoc()
        p.feed(metadata)
        p.endDoc()
    del(p)
    metadataContext = store.intern((notation3.RESOURCE,
                                    metadataURI+"#_formula"))
    matches = store.each((metadataContext,
                          None,
                          None,
                          None))
    del(store)
    # Only want the predicates out of the store
    preds = []
    thing.progress(matches)
    for match in matches:
        typ, uriref = match[0].asPair()
        if uriref.find("http://www.w3.org/2000/10/swap/log#") >= 0:
            continue
        preds.append(peddlerlib.abbreviateResource(uriref))

    generateform = RepeatedForm("/cgi-bin/peddler-3", width="50%")
    generateform.addComponent("", "hidden", ["resource", resource])
    generateform.addComponent("", "hidden", ["type", type])
    generateform.addComponent("", "hidden", ["policyname", policyname])
    generateform.addTitle("GENERATE PEDAL POLICY - STEP 2 / 4", cols = "4")
    generateform.setSubmit("Next Step")
    generateform.setReset("Clear Values")
    generateform.addLegend("Choose what range of values a characteristic for eac
h user can take, further specifying whether a matching user is denied or allowed
 access; you can read each rule as 'for this METADATA, ALLOW or DENY users whose
```

```
 characteristic PREDICATE has this RANGE.  The next step will allow you to combi
ne related rules, if that is your intent.", cols = "4")
    generateform.addLegend("<br />".join(namespaces), cols = "4")
    generateform.addHeaders(["METADATA", "ACCESS", "PREDICATE", "RANGE"])
    generateform.addTemplateComponent("metadata", "select", preds)
    generateform.addTemplateComponent("allow", "select", ["allow", "deny"])
    generateform.addTemplateComponent("predicate", "select", allatts)
    generateform.addTemplateComponent("range", "text", [])
    generateform.computeForm()
    generateform.addButtons(cols = "4")

    doc.makeDoc(title, generateform.getvalue(), extra = head)
    doc.serveDoc()

main()
```

Grouping Rules

```
#!/usr/bin/python2
# $Id: peddler-3,v 1.5 2002/08/21 23:10:53 ryanlee Exp $

# Add an option for just generating versus generating + negotiating

import cgi, sys
sys.path.insert(0, "/var/www/lib")
import htmllib, peddlerlib, thing
from htmllib import *
from types import ListType, StringType

def main():
    # Appearance config
    title = "Peddler PEDAL Generate - Step 3 / 4"
    head = """
  <link rel="stylesheet" type="text/css" href="/style.css">"""

    form    = cgi.FieldStorage()
    doc     = XHTMLDoc()

    type = form.getvalue("type", None)
    resource = form.getvalue("resource", None)
    policyname = form.getvalue("policyname", None)

    if type is None:
        doc.makeError("Missing input <code>type</code>, please correct\n")
        doc.serveDoc()
        return

    if resource is None:
        doc.makeError("Missing input <code>resource</code>, please correct\n")
        doc.serveDoc()
```

```
        return

    if policyname is None:
        doc.makeError("Missing input <code>policyname</code>, please correct\n")
        doc.serveDoc()
        return

    # Take a break from key processing to setup a form
    generateform = TableForm("/cgi-bin/peddler-4", width="50%")
    generateform.addComponent("", "hidden", ["resource", resource])
    generateform.addComponent("", "hidden", ["type", type])
    generateform.addComponent("", "hidden", ["policyname", policyname])

    # Check rule input
    keys = form.keys()
    rules = []
    oldnumber = -1
    for key in keys:
        if key.find(".") > -1:
            kname, knumber = key.split(".")
            if not (form.has_key("range."+knumber)):
                continue
            elif knumber != oldnumber and oldnumber != -1:
                rules.append(onerule)
                onerule = []
                onerule.append(form[key].value)
                oldnumber = knumber
            elif knumber != oldnumber:
                onerule = []
                onerule.append(form[key].value)
                oldnumber = knumber
            else:
                onerule.append(form[key].value)
            generateform.addComponent("", "hidden", [key, form[key].value])
    rules.append(onerule)

    if len(rules) == 0:
        doc.makeError("Missing rules, please correct\n")
        doc.serveDoc()
        return

    # Take care of resubmitting rules
    generateform.addTitle("GENERATE PEDAL POLICY - STEP 3 / 4")
    generateform.addLegend("Connect similar rules with an AND conjunction by mar
king the rules' box in the same column.  Rules cannot be used in more than one c
onjunction.  Rules not intended to be in a conjunction can be left unmarked.  Fi
nally, rules with nothing else to join to are left undisplayed.")
    generateform.setSubmit("Next Step")
    generateform.setReset("Clear Values")
    row = TableRow()
    row.addCell("AND", align="left", cssclass="tablehighlight")
```

```
    row.addCell("RULE", align="left", cssclass="tablehighlight")
    generateform.addRow(row)

    # Now that existence is established, sort it a little more nicely
    ruleDict = {}
    for rule in rules:
        metadata  = rule[0]
        access    = rule[1]
        predicate = rule[2]
        rrange    = rule[3]
        if not ruleDict.has_key((metadata,access)):
            ruleDict[(metadata,access)] = [(predicate, rrange)]
        else:
            ruleDict[(metadata,access)].append((predicate, rrange))

    # Calculate the size of each (metadata,access) key's list
    # That number / 2 then truncated is the number of connection boxes
    # needed per key
    rkeys = ruleDict.keys()
    asize = 1
    first = 1
    j     = 0
    for rkey in rkeys:
        rkeyList = ruleDict[rkey]
        rsize = len(rkeyList)
        # nothing to join with if size of key's list is 1
        if rsize == 1:
            continue
        else:
            asize = rsize
        checkboxsize = rsize / 2 # automatically truncated (floor)
        # don't add the 'next set' advert if the first time
        if first:
            first = 0
        else:
            row = TableRow()
            row.addCell("<b>NEXT SET</b>", cssclass="small", colspan="2")
            j += checkboxsize+1
            generateform.addRow(row)
        for rkeyPair in rkeyList:
            # generate a checkbox
            txt = ""
            for i in range(1, checkboxsize+1):
                a, b = rkey
                c, d = rkeyPair
                txt += "<input type=\"checkbox\" name=\"%s|%s|%s|%s\" value=\"%s
\">" \
                       % (a, b, c, d, i+j)
            row = TableRow()
            row.addCell(txt, cssclass="small")
            row.addCell("%s %s" % rkeyPair, cssclass="small")
```

```
            generateform.addRow(row)

    if asize == 1:
        generateform.addLegend("No rules to combine, just continue to assigning
priorities.")

    # Finish off that form
    generateform.addButtons()

    doc.makeDoc(title, generateform.getvalue(), extra = head)
    doc.serveDoc()

main()
```

Assigning Priorities

```
#!/usr/bin/python2
# $Id: peddler-4,v 1.1 2002/08/21 23:10:09 ryanlee Exp $

# Add an option for just generating versus generating + negotiating

import cgi, sys
sys.path.insert(0, "/var/www/lib")
import htmllib, peddlerlib, thing
from htmllib import *
from types import ListType, StringType

__revision__ = '$Revision: 1.1 $'

def main():
    # Appearance config
    title = "Peddler PEDAL Generate - Step 4 / 4"
    head = """
  <link rel="stylesheet" type="text/css" href="/style.css">"""

    form    = cgi.FieldStorage()
    doc     = XHTMLDoc()

    type = form.getvalue("type", None)
    resource = form.getvalue("resource", None)
    policyname = form.getvalue("policyname", None)

    if type is None:
        doc.makeError("Missing input <code>type</code>, please correct\n")
        doc.serveDoc()
        return

    if resource is None:
        doc.makeError("Missing input <code>resource</code>, please correct\n")
        doc.serveDoc()
```

```
    return

if policyname is None:
    doc.makeError("Missing input <code>policyname</code>, please correct\n")
    doc.serveDoc()
    return

# Take a break from key processing to setup a form
generateform = TableForm("/cgi-bin/peddler-generate", width="70%")
generateform.addComponent("", "hidden", ["resource", resource])
generateform.addComponent("", "hidden", ["type", type])
generateform.addComponent("", "hidden", ["policyname", policyname])

# Check rule input
keys = form.keys()
rules = []
rulesDict = {}
oldnumber = -1
for key in keys:
    if key.find(".") > -1:
        k = key.split(".")
        if len(k) == 2: kname, knumber = k
        if not (form.has_key("range."+knumber)):
            continue
        elif knumber != oldnumber and oldnumber != -1:
            rules.append(onerule)
            rulesDict[oldnumber] = onerule
            onerule = []
            onerule.append(form[key].value)
            oldnumber = knumber
        elif knumber != oldnumber:
            onerule = []
            onerule.append(form[key].value)
            oldnumber = knumber
        else:
            onerule.append(form[key].value)
        generateform.addComponent("", "hidden", [key, form[key].value])
rules.append(onerule)
rulesDict[oldnumber] = onerule

if len(rules) == 0:
    doc.makeError("Missing rules, please correct\n")
    doc.serveDoc()
    return

GroupedDict = {}
groupedrules = []
# Now check rule grouping
for key in keys:
    r = key.split("|")
    if len(r) == 4:
```

```
            gkey = form.getvalue(key)
            groupedrules.append(r)
            if GroupedDict.has_key(gkey):
                GroupedDict[gkey].append(r)
            else:
                GroupedDict[gkey] = [r]
            generateform.addComponent("", "hidden", [key, form[key].value])

    # Now that existence is established, sort it a little more nicely
    ungroupedrules = []
    for rule in rules:
        if rule not in groupedrules:
            ungroupedrules.append(rule)

    generateform.addTitle("GENERATE PEDAL POLICY - STEP 4 / 4")
    generateform.addLegend("Assign a priority to the rules you've set up so far.
  Priorities are used in negotiating policies, which you may need to do with our
 policy.  The more you think a rule is necessary, the higher priority you should
 assign it.")
    generateform.setSubmit("Generate Policy")
    generateform.setReset("Clear Values")
    row = TableRow()
    row.addCell("RULE", align="left", cssclass="tablehighlight")
    row.addCell("PRIORITY", align="left", cssclass="tablehighlight")
    generateform.addRow(row)

    # Now generate the form
    sel = "<select name=\"%s\">\n<option value=\"Must\">Must</option>\n<option v
alue=\"Should\">Should</option>\n<option value=\"May\">May</option>\n</select>\n
"
    gkeys = GroupedDict.keys()
    for gkey in gkeys:
        selname = "group."+ gkey
        row = TableRow()
        txt = ""
        for rule in GroupedDict[gkey]:
            txt += "%s %s %s %s <br />\n" % tuple(rule)
        row.addCell(txt, cssclass="small")
        row.addCell(sel % selname)
        generateform.addRow(row)

    i = 0
    rkeys = rulesDict.keys()
    for rule in ungroupedrules:
        for rkey in rkeys:
            if rulesDict[rkey] == rule:
                i = rkey
        selname = "priority."+i
        row = TableRow()
        txt = "%s %s %s %s <br />\n" % tuple(rule)
        row.addCell(txt, cssclass="small")
```

```
        row.addCell(sel % selname)
        generateform.addRow(row)

    # Finish off that form
    generateform.addButtons()

    doc.makeDoc(title, generateform.getvalue(), extra = head)
    doc.serveDoc()

main()
```

Generation

```
#!/usr/bin/python2
# $Id: peddler-generate,v 1.5 2002/08/21 23:10:29 ryanlee Exp $

import cgi, sys, StringIO, os
sys.path.insert(0, "/var/www/lib")
import htmllib, peddlerlib, thing, notation3
from htmllib import *
from peddlerlib import PedalDoc

# Debugging
import cgitb; cgitb.enable()
thing.setVerbosity(0)

######################
def main():
   title = "Peddler PEDAL Generate - Complete"
   head = """
  <link rel="stylesheet" type="text/css" href="/style.css">"""

   form    = cgi.FieldStorage()
   doc     = XHTMLDoc()

   # Form processing
   type = form.getvalue("type", None)
   resource = form.getvalue("resource", None)
   policyname = form.getvalue("policyname", None)

   if type is None:
      doc.makeError("Missing input <code>type</code>, please correct\n")
      doc.serveDoc()
      return

   if resource is None:
      doc.makeError("Missing input <code>resource</code>, please correct\n")
      doc.serveDoc()
      return
```

```
   if policyname is None:
      doc.makeError("Missing input <code>policyname</code>, please correct\n")
      doc.serveDoc()
      return

   keys = form.keys()
   rulesDict = {}
   groupAllowsDict = {}
   oldnumber = -1
   for key in keys:
      if key.find(".") > -1 and key.find("|") == -1:
         args = key.split(".")
         if len(args) > 2:
            continue
         kname, knumber = args
         if kname == "group": groupAllowsDict[knumber] = form.getvalue(key)
         if rulesDict.has_key(knumber):
            rulesDict[knumber].append(form.getvalue(key))
         else:
            rulesDict[knumber] = [form.getvalue(key)]

   groupedrules = []
   GroupedDict = {}
   for key in keys:
      args = key.split("|")
      if len(args) == 4:
         groupedrules.append(args)
         gkey = form[key].value
         if GroupedDict.has_key(gkey):
            GroupedDict[gkey].append(args)
         else:
            GroupedDict[gkey] = []
            GroupedDict[gkey].append(args)

   policy  = PedalDoc(policyname.strip(), "http://"+os.environ.get("HTTP_HOST")+
resource.strip())
   for rkey in rulesDict.keys():
      if rulesDict[rkey][:4] in groupedrules: continue
      policy.addRule(rulesDict[rkey])

   for gkey in GroupedDict.keys():
      policy.addCompoundRule((GroupedDict[gkey], groupAllowsDict[gkey]))

   policyStr = StringIO.StringIO()
   policy.dumpNested(policy.context, notation3.ToN3(policyStr.write, base = poli
cy.uri, flags = "qa"))

   negotiateform = TableForm("/cgi-bin/peddler-negotiate")
   negotiateform.addComponent("", "hidden", ["resource", resource])
   negotiateform.addComponent("", "hidden", ["type", type])
   negotiateform.addTitle("POLICY GENERATED")
```

```
   negotiateform.addLegend("You can highlight the area and copy the policy somew
here else, or you can continue and negotiate this policy with us to see if we ca
n agree on a merged policy.")
   row = TableRow()
   row.addCell("<textarea name=\"policy\" cols=\"90\" rows=\"30\">%s</textarea>"
%policyStr.getvalue(), colspan="2")
   negotiateform.addRow(row)
   negotiateform.setSubmit("Negotiate Policy")
   negotiateform.addButtons()
   doc.makeDoc(title, negotiateform.getvalue(), extra = head)
   doc.serveDoc()

main()
```

Negotiation

```
#!/usr/bin/python2
# $Id: peddler-negotiate,v 1.5 2002/08/21 19:38:42 ryanlee Exp $

import cgi, sys, StringIO, time, os
sys.path.insert(0, "/var/www/lib")
import htmllib, peddlerlib, llyn, bouncerlib, notation3, thing
from htmllib import *

__revision__ = '$Revision: 1.5 $'

# Negotiation rule files
NEG = '/var/www/lib/negotiate-rules.n3'
CLN = '/var/www/lib/cleanup-negotiate-rules.n3'

# Takes policy as either a textarea or file upload, both of which will look
# the same to it.

def main():
    title = "Peddler PEDAL Negotiation"
    head = """
  <link rel="stylesheet" type="text/css" href="/style.css">"""

    # Local variables
    doc  = XHTMLDoc()
    form = cgi.FieldStorage()

    negotiated = form.getvalue("negotiated", None)
    accept   = form.getvalue("accept", "Reject")
    type     = form.getvalue("type", None)
    resource = form.getvalue("resource", None)
    policy   = form.getvalue("policy", None)

    if type is None:
        doc.makeError("Missing input <code>type</code>, please correct\n")
```

```
        doc.serveDoc()
        return

    if resource is None:
        doc.makeError("Missing input <code>resource</code>, please correct\n")
        doc.serveDoc()
        return

    if policy is None:
        doc.makeError("Missing input <code>policy</code>, please correct\n")
        doc.serveDoc()
        return

    # Data input is ok, dispatch
    if negotiated is None:
        contents = negotiate(resource, type, policy)
        doc.makeDoc(title, contents, extra = head)
    elif accept == "Reject":
        b = bouncerlib.Bouncer()
        b.openDB()
        b.rejectNegotiation(resource, type)
        b.closeDB()
        del(b)
        doc.makeDoc("Policy Rejected", """The policy was rejected and the metada
ta removed.  You can <a href="%s,%s">go back</a> where you started and try somet
hing else.""" % (resource, type), extra = head)
    else:
        b = bouncerlib.Bouncer()
        b.openDB()
        b.addPolicy(resource, type, policy)
        b.closeDB()
        del(b)
        doc.makeRedirect(resource+","+type)

    doc.serveDoc()

def negotiate(resource, type, policy):
    contents = ""
    negotiatedPolicy = StringIO.StringIO()
    policyURI = "online:"+resource+"/"+type
    _metaURI  = policyURI+"/RUN"+`time.time()`
    store = llyn.RDFStore(metaURI = _metaURI)

    b = bouncerlib.Bouncer()
    b.openDB()
    generalPolicy = b.getGeneralPolicy(type)
    b.closeDB()
    del(b)

    # Read in generate rules
    if not os.path.exists(NEG):
```

```
        raise IOError
    else:
        f = file(NEG, 'r')
        negRules = f.read()
        f.close()

    # Read in access rules
    if not os.path.exists(CLN):
        raise IOError
    else:
        f = file(CLN, 'r')
        clnRules = f.read()
        f.close()

    pdoctype = bouncerlib.guessType(policy)
    gdoctype = bouncerlib.guessType(generalPolicy)

    # Set up all the URIS for contexts used
    generalPolicyURI = "online:"+type
    negRulesURI = "online:"+NEG
    clnRulesURI = "online:"+CLN

    # Set up the contexts
    frag = "#_formula"
    policyContext   = store.intern((notation3.FORMULA, policyURI+frag))
    generalPolicyContext = store.intern((notation3.FORMULA, generalPolicyURI+fra
g))
    negRulesContext = store.intern((notation3.FORMULA, negRulesURI+frag))
    clnRulesContext = store.intern((notation3.FORMULA, clnRulesURI+frag))

    # TO NEGOTIATE POLICIES
    # cwm policy general-policy negotiate-rules -think -purge > negotiated

    # Add policy to store
    if pdoctype is None:
        raise IOError
    elif pdoctype == "rdf":
        import sax2rdf, xml.sax._exceptions
        p = sax2rdf.RDFXMLParser(store, policyURI)
        p.feed(policy)
        del(p)
    elif pdoctype == "n3":
        p = notation3.SinkParser(store, policyURI)
        p.startDoc()
        p.feed(policy)
        p.endDoc()
        del(p)
    # Add general policy to store
    if gdoctype is None:
        raise IOError
    elif gdoctype == "rdf":
```

```
        import sax2rdf, xml.sax._exceptions
        p = sax2rdf.RDFXMLParser(store, generalPolicyURI)
        p.feed(generalPolicy)
        del(p)
    elif pdoctype == "n3":
        p = notation3.SinkParser(store, generalPolicyURI)
        p.startDoc()
        p.feed(generalPolicy)
        p.endDoc()
        del(p)
    # Add negotiation rules
    pr = notation3.SinkParser(store, negRulesURI)
    pr.startDoc()
    pr.feed(negRules)
    pr.endDoc()
    del(pr)
    store.moveContext(generalPolicyContext, policyContext)
    store.moveContext(negRulesContext, policyContext)
    store.think(policyContext)

    matches = []
    resourceDict = {}  # {subject} = [resources]
    warningDict = {}   # {subject} = warning text
    # Get any warnings, remember them
    matches = store.each((policyContext,
                          store.intern((notation3.RESOURCE,
                                        peddlerlib.PEDAL_NS + "hasWarning")),
                          None,
                          None))
    used = matches[:]
    if len(matches) > 0:
        for match in matches:
            subject, object = match
            typ, uriref = subject.asPair()
            keep = store.contains((policyContext,
                                   store.intern((notation3.RESOURCE,
                                                 peddlerlib.PEDAL_NS+"warnAuthor
")),
                                   subject,
                                   store.intern((notation3.RESOURCE,
                                                 peddlerlib.PEDAL_NS+"Author")))
)
            if keep is None:
                used.remove(match)
            else:
                warningDict[subject] = object
                res = store.each((policyContext,
                                  store.intern((notation3.RESOURCE,
                                                peddlerlib.PEDAL_NS+"forResource
")),
                                  subject,
```

```
                                None))
                for r in res:
                    if resourceDict.has_key(subject):
                        resourceDict[subject].append(r)
                    else:
                        resourceDict[subject] = [r]
    # Purge
    store.purge(policyContext)

    # TO CLEAN POLICIES
    # cwm negotiated cleanup-negotiate-rules.n3 -think -purge

    p = notation3.SinkParser(store, clnRulesURI)
    p.startDoc()
    p.feed(clnRules)
    p.endDoc()
    del(p)
    store.moveContext(clnRulesContext, policyContext)
    store.think(policyContext)
    store.purge(policyContext)

    # Errors
    error_txt = ""
    resource_list = []
    keys = resourceDict.keys()
    if len(keys) > 0:
        error_txt += "<h4>WARNINGS</h4>\n<p class=\"small\">Parts of your policy
 were superseded by our overall policy, please check these warnings to see if th
e following is still acceptable to you</p>\n"
        for key in keys:
            error_txt += "<p class=\"small\" align=\"center\">\"<font color=\"re
d\">%s</font>\" for a policy statement about resource(s) %s</p>\n" % (warningDic
t[key], ", ".join([i.asPair()[1] for i in resourceDict[key]]))
    contents += error_txt

    # Output
    store.dumpNested(policyContext, notation3.ToN3(negotiatedPolicy.write, polic
yURI, quiet=1))

    simple = TableForm("/cgi-bin/peddler-negotiate", width="50%")
    simple.addComponent("", "hidden", ["resource", resource])
    simple.addComponent("", "hidden", ["type", type])
    simple.addComponent("", "hidden", ["negotiated", "1"])
    row = TableRow()
    row.addCell("POLICY", cssclass="tablehighlight")
    row.addCell("<textarea wrap=\"natural\" cols=\"90\" rows=\"20\" name=\"polic
y\">%s</textarea>" % negotiatedPolicy.getvalue())
    simple.addRow(row)
    simple.addLegend("You can either accept or reject the negotiated policy.  Re
jecting it will also remove metadata associated with this resource since you as
the author retain direct control over whether the information is published or no
```

```
t.")
    row = TableRow()
    row.addCell("<input type=\"submit\" name=\"accept\" value=\"Accept\"> <input
 type=\"submit\" name=\"accept\" value=\"Reject\">", colspan = "2", cssclass="ta
bleend", align="right")
    simple.addRow(row)
    contents += simple.getvalue()
    return contents

main()
```

12.4 Peddler Library

```
#!/usr/bin/python2
# $Id: peddlerlib.py,v 1.9 2002/08/21 23:11:24 ryanlee Exp $

__version__ = "$Revision: 1.9 $"

import sys, llyn, RDFSink, notation3, thing, StringIO, os, types, time, knownNS
from llyn import RDFStore
from RDFSink import FORMULA, LITERAL, ANONYMOUS, VARIABLE, SYMBOL, Logic_NS
from types import ListType, StringType
from notation3 import N3_forSome_URI, N3_forAll_URI
from knownNS import *

###### Configuration
#
thing.setVerbosity(0)   # How much should Peddler say in the logs

policyfile = "/var/www/lib/meta-policy"
                        # Location of the site-wide PEDAL policy file
#
###### End configuration


###### RDFStore Constants
FORMULA =RDFSink.FORMULA
LITERAL =RDFSink.LITERAL
ANONYMOUS = RDFSink.ANONYMOUS
VARIABLE=RDFSink.VARIABLE
RESOURCE=RDFSink.SYMBOL

PEDAL_NS = "http://www.w3.org/2002/01/pedal/pedal#"
Logic_NS = RDFSink.Logic_NS
RDF_type_URI = notation3.RDF_type_URI
DC_NS    = "http://www.purl.org/dc/elements/1.1/"

# Indexes into a rule list
METADATA   = 0
ACCESS     = 1
```

```
PREDICATE  = 2
RANGE      = 3
IMPERATIVE = 4

###### GenerateForm Constants
PARTNAME = 0
PARTHTML = 1

######
###### Code
######

class PedalDoc(RDFStore):
    def __init__(self, name, refuri, genPrefix=None, metaURI=None):
        RDFStore.__init__(self)
        self.bind("here", (RESOURCE, "#"))
        self.bind("pedal", (RESOURCE, PEDAL_NS))
        self.bind("log", (RESOURCE, Logic_NS))
        self.bind("dc", (RESOURCE, DC_NS))
        self.bind("foaf", (RESOURCE, "http://xmlns.com/foaf/0.1/"))
        self.bind("acl", (RESOURCE, "http://www.w3.org/2001/02/acls/ns#"))
        self.bind("p3p", (RESOURCE, "http://www.w3.org/2002/01/p3prdfv1#"))
        self.bind("contact", (RESOURCE, "http://www.w3.org/2000/10/swap/pim/cont
act#"))
        self.bind("rdf", (RESOURCE, "http://www.w3.org/1999/02/22-rdf-syntax-ns#
"))
        self.role = "Author"
        self.uri  = "#"
        self.name = name
        self.id = self.name.replace(" ", "")
        self.context = self.intern((FORMULA, "#_formula"))
        self.intern((RESOURCE, self.uri))
        self.refuri = refuri
        self.bind("doc", (RESOURCE, self.refuri))
        self.intern((RESOURCE, self.refuri))
        self.storeQuad((self.context,
                        self.intern((RESOURCE, RDF_type_URI)),
                        self.intern((RESOURCE, self.uri + self.id)),
                        self.intern((RESOURCE, PEDAL_NS + "Policy"))))
        self.storeQuad((self.context,
                        self.intern((RESOURCE, PEDAL_NS + "policyName")),
                        self.intern((RESOURCE, self.uri + self.id)),
                        self.intern((LITERAL, self.name))))
        self.storeQuad((self.context,
                        self.intern((RESOURCE, PEDAL_NS + "forMetadataOf")),
                        self.intern((RESOURCE, self.uri + self.id)),
                        self.intern((RESOURCE, self.refuri))))
        self.storeQuad((self.context,
                        self.intern((RESOURCE, DC_NS + "creator")),
                        self.intern((RESOURCE, self.uri)),
                        self.intern((LITERAL, "Peddler Generator"))))
```

```
        now  = time.localtime(time.time())[0:3]
        date = "%d/%d/%d" % (now[1], now[2], now[0])
        self.storeQuad((self.context,
                        self.intern((RESOURCE, DC_NS + "date")),
                        self.intern((RESOURCE, self.uri)),
                        self.intern((LITERAL, date))))
        self.storeQuad((self.context,
                        self.intern((RESOURCE, DC_NS + "language")),
                        self.intern((RESOURCE, self.uri)),
                        self.intern((LITERAL, "en"))))
        self.storeQuad((self.context,
                        self.intern((RESOURCE, DC_NS + "title")),
                        self.intern((RESOURCE, self.uri)),
                        self.intern((LITERAL, self.name + " PEDAL Policy"))))
        self.storeQuad((self.context,
                        self.intern((RESOURCE, PEDAL_NS + "authoredBy")),
                        self.intern((RESOURCE, self.uri + self.id)),
                        self.intern((RESOURCE, PEDAL_NS + self.role))))
        self.anonymousIDCount = 0

    def nextID(self):
        self.anonymousIDCount += 1
        return "a" + repr(self.anonymousIDCount)

    def addRule(self, rule, id = None, rsid = None):
        if id is None:
            subj = None
        else:
            subj = id

        if rsid is None:
            rulesubj = None
        else:
            rulesubj = rsid

        if not subj:
            subj = self.nextID()
            self.storeQuad((self.context,
                            self.intern((RESOURCE, N3_forSome_URI)),
                            self.context,
                            self.intern((RESOURCE, self.uri + subj))))
            self.storeQuad((self.context,
                            self.intern((RESOURCE, RDF_type_URI)),
                            self.intern((RESOURCE, self.uri + subj)),
                            self.intern((RESOURCE, PEDAL_NS + "PolicyStatement")
))))
            self.storeQuad((self.context,
                            self.intern((RESOURCE, PEDAL_NS + "forPolicy")),
                            self.intern((RESOURCE, self.uri + subj)),
                            self.intern((RESOURCE, self.uri + self.id))))
        if rule[RANGE].find(":") >= 0:
```

```
    pre, val = rule[RANGE].split(":")
    ns = self.namespaces.get(pre, (RESOURCE, self.uri))
    range = RESOURCE, ns[1] + val
else:
    range = LITERAL, rule[RANGE]

if rule[PREDICATE].find(":") >= 0:
    pre, val = rule[PREDICATE].split(":")
    ns = self.namespaces.get(pre, (RESOURCE, self.uri))
    predicate = RESOURCE, ns[1] + val
else:
    predicate = RESOURCE, self.uri + rule[PREDICATE]

if rule[METADATA].find(":") >= 0:
    pre, val = rule[METADATA].split(":")
    ns = self.namespaces.get(pre, (RESOURCE, self.uri))
    metadata = RESOURCE, ns[1] + val
else:
    metadata = RESOURCE, self.uri + rule[METADATA]

if rule[ACCESS] == "allow":
    visibility = RESOURCE, PEDAL_NS + "visibleTo"
elif rule[ACCESS] == "deny":
    visibility = RESOURCE, PEDAL_NS + "hiddenTo"

imperative = RESOURCE, PEDAL_NS + rule[IMPERATIVE]

subject = RESOURCE, self.uri + subj

# Get some more IDs for the ruleSubject and Component
comp = RESOURCE, self.uri + self.nextID()

if rulesubj is None:
    rulesubj = FORMULA, self.uri + self.nextID()
    self.storeQuad((self.context,
                    self.intern((RESOURCE, N3_forSome_URI)),
                    self.context,
                    self.intern(rulesubj)))
    self.storeQuad((self.context,
                    self.intern((RESOURCE, PEDAL_NS + "ruleSubject")),
                    self.intern(subject),
                    self.intern(rulesubj)))
else:
    rulesubj = FORMULA, self.uri + rulesubj

self.storeQuad((self.intern(rulesubj),
                self.intern(predicate),
                self.intern((RESOURCE, PEDAL_NS + "someone")),
                self.intern(range)))
self.storeQuad((self.context,
                self.intern((RESOURCE, N3_forSome_URI)),
```

```
                    self.context,
                    self.intern(comp)))
        self.storeQuad((self.context,
                    self.intern((RESOURCE, PEDAL_NS + "hasComponent")),
                    self.intern(subject),
                    self.intern(comp)))
        self.storeQuad((self.context,
                    self.intern((RESOURCE, RDF_type_URI)),
                    self.intern(comp),
                    self.intern((RESOURCE, PEDAL_NS + "Component"))))
        self.storeQuad((self.context,
                    self.intern((RESOURCE, PEDAL_NS + "withPredicate")),
                    self.intern(comp),
                    self.intern(predicate)))
        self.storeQuad((self.context,
                    self.intern((RESOURCE, PEDAL_NS + "withRange")),
                    self.intern(comp),
                    self.intern(range)))
        self.storeQuad((self.context,
                    self.intern((RESOURCE, PEDAL_NS + "hasPriority")),
                    self.intern(subject),
                    self.intern(imperative)))
        self.storeQuad((self.context,
                    self.intern((RESOURCE, PEDAL_NS + "forResource")),
                    self.intern(subject),
                    self.intern(metadata)))
        self.storeQuad((self.context,
                    self.intern((RESOURCE, PEDAL_NS + "withVisibility")),
                    self.intern(subject),
                    self.intern(visibility)))
        return

    def addCompoundRule(self, rule):
        subj = self.nextID()
        rulesubj = self.nextID()
        rules, allows = rule
        subject = RESOURCE, self.uri + subj
        rulesubject = FORMULA, self.uri + rulesubj
        self.storeQuad((self.context,
                    self.intern((RESOURCE, N3_forSome_URI)),
                    self.context,
                    self.intern(subject)))
        self.storeQuad((self.context,
                    self.intern((RESOURCE, RDF_type_URI)),
                    self.intern(subject),
                    self.intern((RESOURCE, PEDAL_NS + "PolicyStatement"))))
        self.storeQuad((self.context,
                    self.intern((RESOURCE, PEDAL_NS + "forPolicy")),
                    self.intern(subject),
                    self.intern((RESOURCE, self.uri + self.id))))
        self.storeQuad((self.context,
```

```
                        self.intern((RESOURCE, N3_forSome_URI)),
                        self.context,
                        self.intern(rulesubject)))
        self.storeQuad((self.context,
                        self.intern((RESOURCE, PEDAL_NS + "ruleSubject")),
                        self.intern(subject),
                        self.intern(rulesubject)))
        for r in rules:
            r.append(allows)
            self.addRule(r, id = subj, rsid = rulesubj)
        return

def policyExists():
    if os.path.exists(policyfile):
        return 1
    else:
        return 0

def prefixns(ns, term):
    return ns + ":" + term

def abbreviate(ns):
    for knownns in KNOWN_NS_LIST:
        if knownns[0] == ns:
            return knownns[1]
    return None

def lookupPrefix(prefix):
    for knownns in KNOWN_NS_LIST:
        if knownns[1] == prefix:
            return knownns[0]
    return None

def abbreviateResource(resource):
    i = resource.find("#")
    l = len(resource)
    if i > 0:
        abbr = abbreviate(resource[0:i+1])
        if not abbr is None:
            return abbr+":"+resource[i+1:l]

    i = resource.rfind("/")
    if i > 0:
        abbr = abbreviate(resource[0:i+1])
        if not abbr is None:
            return abbr+":"+resource[i+1:l]
    return resource

def expandAbbreviation(resource):
    i = resource.find(":")
    l = len(resource)
```

```
    if i == 0:
        return resource
    if i > 0:
        ns = lookupPrefix(resource[0:i])
        if not ns is None:
            return ns+resource[i+1:l]
    return resource
```

12.5 Bouncer

```
#!/usr/bin/python2
# $Id: bouncer,v 1.6 2002/08/21 15:52:30 ryanlee Exp $

import cgi, os, urllib2, sys
sys.path.insert(0, "/var/www/lib")
import htmllib, peddlerlib, bouncerlib, thing
from htmllib import *
from bouncerlib import *

__revision__ = '$Revision: 1.6 $'

head = """
  <link rel="stylesheet" type="text/css" href="/style.css">"""

def main():
    form = cgi.FieldStorage()
    keys = form.keys()
    doc  = XHTMLDoc()
    info = []

    type = form.getvalue("type", None)
    resource = form.getvalue("resource", None)
    pfile = form.getfirst("policyfile", None)
    mfile = form.getfirst("metadatafile", None)
    keys = form.keys()
    for key in keys:
        if key.find("info.") >= 0:
            l = len(key)
            info.append(key)

    if type is None:
        doc.makeError("Missing input <code>type</code>, please correct\n")
        doc.serveDoc()
        return

    if resource is None:
        doc.makeError("Missing input <code>resource</code>, please correct\n")
        doc.serveDoc()
        return
```

```
if not (type == "contact" or type == "meta"):
    doc.makeError("Invalid comma tool <code>type</code>: "+type+"\n")
    doc.serveDoc()
    return

if not mfile is None:
    if mfile.strip() == "":
        doc.makeError("Uploaded an empty metadata file\n")
        doc.serveDoc()
        return

if not pfile is None:
    if pfile.strip() == "":
        doc.makeError("Uploaded an empty policy file\n")
        doc.serveDoc()
        return

# Lastly, check if the URL is valid - Apache won't match the rewrite
# condition correctly for doing this in httpd.conf, so it's been
# shunted to here until RewriteCond does matching for -F right
try:
    t = urllib2.urlopen("http://"+os.environ.get("HTTP_HOST")+resource)
except urllib2.HTTPError, urllib2.HTTPError.code:
    stat = "%s" % urllib2.HTTPError.code
    if stat.find("404") > 0:
        doc.makeNotFound(resource)
        doc.serveDoc()
        return
    else:
        doc.makeError(stat)
        doc.serveDoc()
        return

# Everything should be fine with the input by this point...

# Figure out what to do, dispatch as needed
if len(info) == 0 and mfile is None and pfile is None:
    # nothing submitted, run through Bouncer guard
    checkResource(resource, type)
    return
elif len(info) > 0:
    # information submitted, check against policy and show any data
    executePolicy(resource, type, form, info)
    return
else:
    # file upload
    if not (mfile is None):
        # upload metadata
        uploadFile(resource, type, mfile)
    else:
        # upload policy
```

```
            uploadPolicy(resource, type, pfile)

    # redirect back to the requested resource
    doc.makeRedirect(resource+","+type)
    doc.serveDoc()
    return

def checkResource(resource, type):
    # Now it's a matter of seeing if there's metadata for this resource,
    # and if there's a pedal policy governing the resource
    b   = Bouncer()
    doc = XHTMLDoc()

    b.openDB()
    if b.checkResourceTypeInDB(resource, type) == 0:
        # Offer to upload a resource file
        uploadform = TableForm(resource+","+type, enctype="multipart/form-data")
        uploadform.addComponent("", "hidden", ["resource", resource])
        uploadform.addComponent("", "hidden", ["type", type])
        uploadform.addTitle("UPLOAD METADATA")
        uploadform.setSubmit("Upload File")
        uploadform.addLegend("Upload metadata for <code>%s</code> for tool <code
>%s</code>" % (resource, type))
        uploadform.addComponent("Missing Metadata File", "file", "metadatafile")
        uploadform.addButtons()
        doc.makeDoc("Add Files", uploadform.getvalue(), extra = head)
        doc.serveDoc()
        return
    if b.checkPolicyForResourceTypeInDB(resource, type) == 0:
        # Offer to upload a policy file
        form_p = 1
        uploadform = TableForm("/cgi-bin/peddler-negotiate", enctype="multipart/
form-data", width="50%")
        uploadform.addTitle("UPLOAD POLICY")
        uploadform.addComponent("", "hidden", ["resource", resource])
        uploadform.addComponent("", "hidden", ["type", type])
        uploadform.setSubmit("Upload File")
        uploadform.addLegend("Upload direct policy for <code>%s</code> for tool
<code>%s</code> (another general policy already exists).  You can also %s if you
 don't have one." % (resource, type, makeAnchor("/cgi-bin/peddler?resource="+res
ource+"&type="+type, "generate a policy")))
        uploadform.addComponent("Policy File", "file", "policy")
        uploadform.addButtons()
    else:
        form_p = 0

    # Look it up
    if form_p:
        content = uploadform.getvalue()
    else:
        content = ""
```

```
    content += b.generateBouncerForm(resource,type)
    b.closeDB()

    doc.makeDoc("Metadata Gateway for %s" % resource, content, extra = head)
    doc.serveDoc()
    return

def executePolicy(resource, type, form, keys):
    b   = Bouncer()
    doc = XMLDoc()
    requesterInputs = []

    for key in keys:
        requesterInputs.append([key, form.getvalue(key)])

    b.openDB()
    metadata = b.filterMetadata(resource, type, requesterInputs)
    b.closeDB()
    doc.makeDoc(metadata)
    doc.serveDoc()
    return

def uploadFile(resource, type, file):
    b = Bouncer()
    doc = XHTMLDoc()
    b.openDB()
    try:
        b.addData(resource, type, file)
    except pgdb.DatabaseError:
        b.closeDB()
        doc.makeError("Database error - oops.  Sorry.")
        doc.serveDoc()
        raise
    b.closeDB()
    return

def uploadPolicy(resource, type, file):
    b = Bouncer()
    doc = XHTMLDoc()
    b.openDB()
    # NEGOTIATE HERE!
    try:
        b.addPolicy(resource, type, file)
    except pgdb.DatabaseError:
        b.closeDB()
        doc.makeError("Database error - oops.  Sorry.")
        doc.serveDoc()
        raise
    b.closeDB()
    return
```

```
main()
```

12.6 Bouncer Library

```
#!/usr/bin/python2
# $Id: bouncerlib.py,v 1.7 2002/08/21 15:50:35 ryanlee Exp $

__version__ = "$Revision: 1.7 $"

import os, sys, peddlerlib, cwm, sax2rdf, xml.sax._exceptions, llyn
import thing, notation3, htmllib, StringIO, time
sys.path.append('/usr/local/lib/python2.2/site-packages')
import pgdb

###### Configuration
#
thing.setVerbosity(0)   # How much should Peddler say in the logs

# For connecting through pgdb
HOSTNAME = 'localhost'
DATABASE = 'ryanlee'
USER     = 'ryanlee'

policyfile = peddlerlib.policyfile
GEN = '/var/www/lib/generate-rules.n3'
ACC = '/var/www/lib/access-rules.n3'
#
###### End configuration

def initialize():
    try:
        db = pgdb.connect(host     = HOSTNAME,
                          database = DATABASE,
                          user     = USER)
    except:
        print "Could not connect to database!"
        raise

    return db

def quote(str):
    return str.replace("\'", "\'\'")

def guessType(doc):
    if doc[0] == "#" or doc.strip()[0:7] == "@prefix":
        type = "n3"
    elif doc.find('xmlns') > 0:
        type = "rdf"
    else:
```

```
        type = None
    return type

class Bouncer:
    def __init__(self):
        pass

    def openDB(self):
        self.conn = initialize()
        return

    def closeDB(self):
        self.conn.close()
        return

    def checkResourceInDB(self, resource):
        # http://www.redhat.com/docs/manuals/database/RHDB-7.1.3-Manual/prog/exa
mplepy.html
        c = self.conn.cursor()
        query = "SELECT \
                   count(*) as n \
                 FROM \
                   resource_uri \
                 WHERE \
                   resource_uri = '%s'" % resource
        c.execute(query)
        n = c.fetchone()
        c.close()
        return n[0]

    def checkResourceTypeInDB(self, resource, type):
        c = self.conn.cursor()
        query = """
                 SELECT
                   count(*) as n
                 FROM
                   resource_meta
                 WHERE
                   resource_url = '%s'
                   AND type = '%s'""" % (resource, type)
        c.execute(query)
        n = c.fetchone()
        c.close()
        return n[0]

    def checkPolicyForResourceTypeInDB(self, resource, type):
        c = self.conn.cursor()
        query = """
                 SELECT
                   count(*) as n
                 FROM
```

```
            meta_policy
          WHERE
            resource_url = '%s'
            AND type = '%s'""" % (resource, type)
    c.execute(query)
    n = c.fetchone()
    c.close()
    return n[0]

def addData(self, resource, type, data):
    c = self.conn.cursor()
    chk_query = """
           SELECT
             count(*)
           FROM
             resource_meta
           WHERE
             resource_url = '%s'
             AND type = '%s'""" % (resource, type)
    dml = """
             INSERT INTO resource_meta
               (resource_url, type, data)
             VALUES
               ('%s', '%s', '%s') """ % (resource, type, quote(data))
    c.execute(chk_query)
    n = c.fetchone()
    if n[0] > 0:
        c.close()
        return
    else:
        try:
            c.execute(dml)
        except pgdb.DatabaseError:
            self.conn.rollback()
            raise
    self.conn.commit()
    c.close()
    return

def addPolicy(self, resource, type, data):
    c = self.conn.cursor()
    chk_query = """
           SELECT
             count(*)
           FROM
             meta_policy
           WHERE
             resource_url = '%s'
             AND type = '%s'""" % (resource, type)
    dml = """
             INSERT INTO meta_policy
```

```
                (resource_url, type, data)
              VALUES
                ('%s', '%s', '%s') """ % (resource, type, quote(data))
        c.execute(chk_query)
        n = c.fetchone()
        if n[0] > 0:
            c.close()
            return
        else:
            try:
                c.execute(dml)
            except pgdb.DatabaseError:
                self.conn.rollback()
                raise
        self.conn.commit()
        c.close()
        return

    def getData(self, resource, type):
        c = self.conn.cursor()
        query = """
                 SELECT
                   data
                 FROM
                   resource_meta
                 WHERE
                   resource_url = '%s'
                   AND type = '%s'""" % (resource, type)
        c.execute(query)
        data = c.fetchone()
        c.close()
        return data[0]

    def getPolicy(self, resource, type):
        # Get either the exact policy or the general policy
        if self.checkPolicyForResourceTypeInDB(resource, type) > 0:
            return self.getExactPolicy(resource, type)
        else:
            return self.getGeneralPolicy(type)

    def getExactPolicy(self, resource, type):
        # The policy for a resource, assume someone's checked it exists
        c = self.conn.cursor()
        query = """
                 SELECT
                   data
                 FROM
                   meta_policy
                 WHERE
                   resource_url = '%s'
                   AND type = '%s'""" % (resource, type)
```

```
        c.execute(query)
        data = c.fetchone()
        c.close()
        return data[0]

    def getGeneralPolicy(self, type):
        # One general policy for everything of one type
        c = self.conn.cursor()
        query = """
                 SELECT
                   data
                 FROM
                   tool_policy
                 WHERE
                   type = '%s'""" % type
        c.execute(query)
        data = c.fetchone()
        c.close()
        return data[0]

    def getFormArgs(self, resource, type):
        store = llyn.RDFStore()
        policy = self.getPolicy(resource, type)

        doctype = guessType(policy)
        thisURI = "online:"+type
        if doctype is None:
            raise IOError
        elif doctype == "rdf":
            import sax2rdf, xml.sax._exceptions
            p = sax2rdf.RDFXMLParser(store, thisURI)
            p.feed(policy)
            del(p)
        elif doctype == "n3":
            p = notation3.SinkParser(store, thisURI)
            p.startDoc()
            p.feed(policy)
            p.endDoc()
            del(p)
        thisContext = store.intern((notation3.FORMULA, thisURI + "#_formula"))
        matches = store.each((thisContext,
                              store.intern((notation3.RESOURCE, peddlerlib.PEDAL
_NS + "withPredicate")),
                              None,
                              None))
        argsList = []
        for match in matches:
            # only interested in the object
            argsList.append(match[1])
            if argsList.count(match[1]) > 1:
                argsList.pop()
```

```
        return argsList

    def generateBouncerForm(self, resource, type):
        form = htmllib.TableForm(resource+","+type, width="70%")
        args = self.getFormArgs(resource, type)
        form.addTitle("REQUIRED INFORMATION")
        form.addComponent("", "hidden", ["resource", resource])
        form.addComponent("", "hidden", ["type", type])
        form.addComponent("", "hidden", ["info.pedal:hasRole", "pedal:Anonymous"
])
        form.setSubmit("Submit Credentials")
        form.addLegend("Please fill out the following fields as they pertain to
you.  What you enter will be used to determine what metadata is available for yo
u to see.  Nothing you enter will be stored, though you can leave the form blank
 if you so choose.")
        for arg in args:
            type, uriref = arg.asPair()
            cleanabbrev = peddlerlib.abbreviateResource(uriref)
            form.addComponent(htmllib.makeAnchor(uriref, name=cleanabbrev),
                              "text", "info."+cleanabbrev)
        form.addButtons()
        return form.getvalue()

    def filterMetadata(self, resource, type, inputList):
        metadataURI = "online:"+resource+"/"+type
        _metaURI  = metadataURI+"/RUN"+`time.time()`
        final = StringIO.StringIO()
        store = llyn.RDFStore(metaURI = _metaURI)
        policy = self.getPolicy(resource, type)
        metadata = self.getData(resource, type)

        # Read in generate rules
        if not os.path.exists(GEN):
            raise IOError
        else:
            f = file(GEN, 'r')
            genRules = f.read()
            f.close()

        # Read in access rules
        if not os.path.exists(ACC):
            raise IOError
        else:
            f = file(ACC, 'r')
            accRules = f.read()
            f.close()

        pdoctype = guessType(policy)
        mdoctype = guessType(metadata)

        # Set up all the URIS for contexts used
```

```
policyURI   = "online:"+resource+"/pedal"
genRulesURI = "online:"+GEN
accRulesURI = "online:"+ACC
play1URI = metadataURI+"/PLAY1"
play2URI = metadataURI+"/PLAY2"
play3URI = metadataURI+"/PLAY3"
play4URI = metadataURI+"/PLAY4"
play5URI = metadataURI+"/PLAY5"

# Set up the contexts
frag = "#_formula"
metadataContext = store.intern((notation3.FORMULA, metadataURI+frag))
policyContext   = store.intern((notation3.FORMULA, policyURI+frag))
genRulesContext = store.intern((notation3.FORMULA, genRulesURI+frag))
accRulesContext = store.intern((notation3.FORMULA, accRulesURI+frag))
play1Context = store.intern((notation3.FORMULA, play1URI+frag))
play2Context = store.intern((notation3.FORMULA, play2URI+frag))
play3Context = store.intern((notation3.FORMULA, play3URI+frag))
play4Context = store.intern((notation3.FORMULA, play4URI+frag))
play5Context = store.intern((notation3.FORMULA, play5URI+frag))


# TO GENERATE ACCESS RULES FROM A POLICY
# cwm meta-policy -filter=pedal-generate-rules.n3 > rules
#  metadataContext        genRulesContext           play1Context
if pdoctype is None:
    raise IOError
elif pdoctype == "rdf":
    import sax2rdf, xml.sax._exceptions
    p = sax2rdf.RDFXMLParser(store, policyURI)
    p.feed(policy)
    del(p)
elif pdoctype == "n3":
    p = notation3.SinkParser(store, policyURI)
    p.startDoc()
    p.feed(policy)
    p.endDoc()
    del(p)
pr = notation3.SinkParser(store, genRulesURI)
pr.startDoc()
pr.feed(genRules)
pr.endDoc()
del(pr)
store.applyRules(policyContext, genRulesContext, play1Context)

# TO GENERATE VISIBILITY STATEMENTS BASED ON METADATA AND USER INPUT
# cwm test-meta.n3 test-meta-input.n3 -filter=rules > visibility
#     metadataContext   inputContext     play1Context  play2Context
if mdoctype is None:
    raise IOError
elif mdoctype == "rdf":
```

```
            import sax2rdf, xml.sax._exceptions
            p = sax2rdf.RDFXMLParser(store, play4URI)
            p.feed(metadata)
            del(p)
            p = sax2rdf.RDFXMLParser(store, play1URI)
            p.feed(metadata)
            del(p)
        elif mdoctype == "n3":
            p = notation3.SinkParser(store, play4URI)
            p.startDoc()
            p.feed(metadata)
            p.endDoc()
            del(p)
            p = notation3.SinkParser(store, play1URI)
            p.startDoc()
            p.feed(metadata)
            p.endDoc()
            del(p)
        self._makeInputN3(store, play1Context, inputList)
        store.applyRules(play1Context, play1Context, play4Context)

        # TO GENERATE FINAL OUTPUT OF METADATA
        # cwm visibility test-meta.n3 -filter=pedal-access-rules.n3 -purge
        #     play2Context metadataContext     accRulesContext
        #  ends in metadataContext
        pr = notation3.SinkParser(store, accRulesURI)
        pr.startDoc()
        pr.feed(accRules)
        pr.endDoc()
        del(pr)
        store.applyRules(play4Context, accRulesContext, play5Context)
        store.purge(play5Context)

        # Output
        store.dumpNested(play5Context, notation3.ToRDF(final, play5URI))
        return final.getvalue()

    def _makeInputN3(self, store, context, inputs):
        PEDAL_req_URI = peddlerlib.PEDAL_NS + "Requester"
        for input in inputs:
            predicate, range = input
            predicate = predicate[5:len(predicate)]
            uriref = peddlerlib.expandAbbreviation(predicate)
            i = range.find(":")
            j = range.find("/")
            l = len(range)
            if i >= 0:
                if j < 0:
                    range = peddlerlib.expandAbbreviation(range)
                store.storeQuad((context,
                                 store.intern((notation3.RESOURCE, uriref)),
```

```
                                store.intern((notation3.RESOURCE,
                                              PEDAL_req_URI)),
                                store.intern((notation3.RESOURCE, range))))
            else:
                store.storeQuad((context,
                                store.intern((notation3.RESOURCE, uriref)),
                                store.intern((notation3.RESOURCE,
                                              PEDAL_req_URI)),
                                store.intern((notation3.LITERAL, range))))
        return

    def rejectNegotiation(self, resource, type):
        # Delete metadata
        c = self.conn.cursor()
        dml = """
                 DELETE FROM resource_meta
                 WHERE resource_url = '%s'
                 AND   type = '%s' """ % (resource, type)
        try:
            c.execute(dml)
        except pgdb.DatabaseError:
            self.conn.rollback()
            raise
        self.conn.commit()
        c.close()
        return
```

12.7 HTTP Library[3]

```
#!/usr/bin/python2

import os

class Response:
    def __init__(self, status = "200 OK", contentType = "text/html"):
        self.status = status
        self.contentType = contentType
        self.location = None

    def respond(self):
        response = ""
        if self.status != "200 OK":
            response += "Status: " + self.status + "\n"
        if not (self.location is None):
            response += "Location: " + self.location + "\n"
        response += "Content-type: " + self.contentType + "\n\n"
        return response
```

[3]**Not to be confused with Python's built in httplib module**

12.8 HTML Library

```
#!/usr/bin/python2

# CAVEATS:
# Pretty sure adding a radio or checkbox type won't work as advertised
#  in TableForm and RepeatedForm, haven't tested yet.

from types import ListType, DictType
import thing, libhttp
#### CONTSTANTS
# For the size of a repeated form

DEFAULTSIZE = 20

#
#### END CONSTANTS

class XHTMLDoc:
    def __init__(self):
        self.doc = ""
        self.r = libhttp.Response()

    def docType(self):
        txt = """<?xml version="1.0"?>
<!DOCTYPE HTML "
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">\n"""
        return txt

    def xhtmlTitle(self, title, extra=""):
        txt = """<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="
en">
 <head>""" + extra + """
  <title>""" + title + """</title>
 </head>
 <body>\n"""
        return txt
#" my emacs python mode dislikes the way things look here cuz of quotes@@@

    def xhtmlFooter(self):
        txt = """ </body>
</html>\n"""
        return txt

    def makeDoc(self, title, content, extra=""):
        self.doc = self.r.respond()
        self.doc += self.docType()
        self.doc += self.xhtmlTitle(title, extra)
        self.doc += content
        self.doc += self.xhtmlFooter()
```

```
        return

    def makePlainDoc(self, content):
        self.doc = self.r.respond()
        self.doc += content
        return

    def makeError(self, errormessage):
        self.r.status = "500 Internal Server Error"
        self.doc = self.r.respond()
        self.doc += self.docType()
        self.doc += self.xhtmlTitle("Error")
        self.doc += "<h1>Internal Server Error</h1>\n"
        self.doc += errormessage
        self.doc += self.xhtmlFooter()
        return

    def makeNotFound(self, url):
        self.r.status = "404 Not Found"
        self.doc = self.r.respond()
        self.doc += self.docType()
        self.doc += self.xhtmlTitle("404 Not Found")
        self.doc += "<h1>Not Found</h1>\n"
        self.doc += "The requested URL "+url+" was not found"
        self.doc += self.xhtmlFooter()
        return

    def makeRedirect(self, url):
        self.r.status = "302 Found"
        self.r.location = url
        self.doc = self.r.respond()
        self.doc += self.docType()
        self.doc += self.xhtmlTitle("Redirect")
        self.doc += "<h1>Redirect</h1>\n"
        self.doc += "The requested URL has moved to"+url
        self.doc += self.xhtmlFooter()
        return

    def serveDoc(self):
        print self.doc

class XMLDoc(XHTMLDoc):
    def __init__(self):
        self.doc = ""
        self.r = libhttp.Response()
        self.r.contentType = "application/xml"

    def docType(self):
        txt = "<?xml version=\"1.0\"?>\n"
        return txt
```

```
    def makeDoc(self, content):
        self.doc = self.r.respond()
        self.doc += self.docType()
        self.doc += content
        return

class Table:
    def __init__(self,
                 pad = 2,
                 border = "0",
                 cellpadding = "5",
                 cellspacing = "1",
                 bgcolor = "#cccccc",
                 align = "center",
                 width = None,
                 extra = None):
        atts = []
        if border != None:
            atts.append("border=\""+border+"\"")
        if cellpadding != None:
            atts.append("cellpadding=\""+cellpadding+"\"")
        if cellspacing != None:
            atts.append("cellspacing=\""+cellspacing+"\"")
        if bgcolor != None:
            atts.append("bgcolor=\""+bgcolor+"\"")
        if align != None:
            atts.append("align=\""+align+"\"")
        if width != None:
            atts.append("width=\""+width+"\"")
        if extra != None:
            atts.append(extra)
        if len(atts) > 0:
            self.head = " "*pad+"<table "+" ".join(atts)+">\n"
        else:
            self.head = " "*pad+"<table>\n"
        self.tail = " "*pad+"</table>\n"
        self.table = []
        self.value = ""

    def addRow(self, row):
        self.table.append(row)
        return

    def getvalue(self):
        content = ""
        for row in self.table:
            content += row.getvalue()
        self.value = self.head + content + self.tail
        return self.value

class TableRow:
```

```
def __init__(self,
             pad = 3,
             bgcolor = None,
             style = None,
             cssclass = None,
             extra = None):
    atts = []
    if bgcolor != None:
        atts.append("bgcolor=\""+bgcolor+"\"")
    if style != None:
        atts.append("style=\""+style+"\"")
    if cssclass != None:
        atts.append("class=\""+cssclass+"\"")
    if extra != None:
        atts.append(extra)

    if len(atts) > 0:
        self.head = " "*pad+"<tr "+" ".join(atts)+">\n"
    else:
        self.head = " "*pad+"<tr>\n"

    self.tail = " "*pad+"</tr>\n"
    self.row = []
    self.value = ""

def addCell(self, content,
             pad = 4,
             align = "left",
             valign = "top",
             bgcolor = None,
             colspan = None,
             style = None,
             cssclass = None,
             extra = None):
    atts = []
    if align != None:
        atts.append("align=\""+align+"\"")
    if valign != None:
        atts.append("valign=\""+valign+"\"")
    if bgcolor != None:
        atts.append("bgcolor=\""+bgcolor+"\"")
    if colspan != None:
        atts.append("colspan=\""+colspan+"\"")
    if style != None:
        atts.append("style=\""+style+"\"")
    if cssclass != None:
        atts.append("class=\""+cssclass+"\"")
    if extra != None:
        atts.append(extra)
    if len(atts) > 0:
        value = " "*pad+"<td "+" ".join(atts)+">"+content+"\n"+" "*pad+"</td
```

```
>\n"
        else:
            value = " "*pad+"<td>"+content+"\n"+" "*pad+"</td>\n"
        self.row.append(value)
        return

    def reset(self):
        self.vale = ""
        self.row = []

    def getvalue(self):
        content = ""
        for cell in self.row:
            content += cell
        self.value = self.head + content + self.tail
        return self.value

def vertspace(pad = 0):
    return " "*pad+"<br />"

class Form:
    def __init__(self, action,
                 method = "POST",
                 enctype = None):
        atts = []
        atts.append("action=\""+action+"\"")
        atts.append("method=\""+method+"\"")
        if enctype != None:
            atts.append("enctype=\""+enctype+"\"")
        self.formhead = "<form "+" ".join(atts)+">\n"
        self.formtail = "</form>\n"
        self.components = []
        self.value = ""
        self.setSubmit()
        self.setReset()

    def addHidden(self, name, value):
        atts = []
        atts.append("name=\""+name+"\"")
        atts.append("type=\"hidden\"")
        atts.append("value=\""+value+"\"")
        self.components.append("<input "+" ".join(atts)+">\n")
        return

    def addInput(self, name, type,
                 size = "30",
                 maxlength = None,
                 value = None):
        atts = []
        atts.append("name=\""+name+"\"")
        atts.append("type=\""+type+"\"")
```

```
        if size != None:
            atts.append("size=\""+size+"\"")
        if maxlength != None:
            atts.append("maxlength=\""+maxlength+"\"")
        if value != None:
            atts.append("value=\""+value+"\"")
        self.components.append("<input "+" ".join(atts)+">")
        return

    def addTextarea(self, name,
                    cols = "30",
                    rows = "5",
                    wrap = "natural",
                    value = None):
        atts = []
        atts.append("name=\""+name+"\"")
        atts.append("cols=\""+cols+"\"")
        atts.append("rows=\""+rows+"\"")
        atts.append("wrap=\""+wrap+"\"")
        txt = "<textarea "+" ".join(atts)+">"
        if value != None:
            txt += value+"</textarea>"
        else:
            txt += "</textarea>"
        self.components.append(txt)
        return

    def addSelect(self, name, optionElements,
                  selectedElement = None):
        head = "<select name=\""+name+"\">\n"
        tail = "</select>\n"
        txt  = head
        if isinstance(optionElements, ListType):
            for element in optionElements:
                txt += "<option value=\""+element+"\">"+element+"</option>\n"
        elif isinstance(optionElements, DictType):
            vals = optionElements.keys()
            for val in vals:
                txt += "<option value=\""+val+"\">"+optionElements[val]+"</optio
n>\n"
        else:
            raise InputError, "optionElements either a list or a hash"
        txt += tail
        self.components.append(txt)
        return

    def addCheckbox(self, name, optionElements,
                    checkedElements = None):
        txt = ""
        if isinstance(optionElements, ListType):
            for element in optionElements:
```

```
                txt += "<input type=\"checkbox\" name=\""+name+"\" value=\""+ele
ment+"\">"+element+"\n"
            self.components.append(txt)
        elif isinstance(optionElements, DictType):
            vals = optionElements.keys()
            for val in vals:
                txt += "<input type=\"checkbox\" name=\""+name+"\" value=\""+val
+"\">"+optionElements[val]+"\n"
            self.components.append(txt)
        else:
            raise InputError, "optionElements either a list or a hash"
        return

    def addRadio(self, name, optionElements,
                 checkedElement = None):
        txt = ""
        if isinstance(optionElements, ListType):
            for element in optionElements:
                txt += "<input type=\"radio\" name=\""+name+"\" value=\"+element
+\">"+element+"\n"
            self.components.append(txt)
        elif isinstance(optionElements, DictType):
            vals = optionElements.keys()
            for val in vals:
                txt += "<input type=\"radio\" name=\""+name+"\" value=\""+val+"\
">"+optionElements[val]+"\n"
            self.components.append(txt)
        else:
            raise InputError, "optionElements either a list or a hash"
        return

    def setSubmit(self, text = "Submit"):
        if text == None:
            self.submit = ""
        else:
            self.submit = "<input type=\"submit\" value=\""+text+"\">"

    def setReset(self, text = None):
        if text == None:
            self.reset = ""
        else:
            self.reset = "<input type=\"reset\" value=\""+text+"\">"

    def getvalue(self):
        content = ""
        for element in self.components:
            content += element
        content += self.reset + self.submit
        self.value = self.formhead + content + self.formtail
        return self.value
```

```
class TableForm(Table, Form):
    def __init__(self, action,
                 pad = 2,
                 method = "POST",
                 enctype = None,
                 border = "0",
                 cellpadding = "5",
                 cellspacing = "1",
                 bgcolor = "#cccccc",
                 align = "center",
                 width = None,
                 extra = None):
        formatts = []
        formatts.append("action=\""+action+"\"")
        formatts.append("method=\""+method+"\"")
        if enctype != None:
            formatts.append("enctype=\""+enctype+"\"")
        self.formhead = " "*pad+"<form "+" ".join(formatts)+">\n"
        self.formtail = " "*pad+"</form>\n"
        self.components = []
        self.value = ""
        self.setSubmit()
        self.setReset()
        tableatts = []
        if border != None:
            tableatts.append("border=\""+border+"\"")
        if cellpadding != None:
            tableatts.append("cellpadding=\""+cellpadding+"\"")
        if cellspacing != None:
            tableatts.append("cellspacing=\""+cellspacing+"\"")
        if bgcolor != None:
            tableatts.append("bgcolor=\""+bgcolor+"\"")
        if align != None:
            tableatts.append("align=\""+align+"\"")
        if width != None:
            tableatts.append("width=\""+width+"\"")
        if extra != None:
            tableatts.append(extra)
        if len(tableatts) > 0:
            self.tablehead = " "*pad+"<table "+" ".join(tableatts)+">\n"
        else:
            self.tablehead = " "*pad+"<table>\n"
        self.tabletail = " "*pad+"</table>\n"
        self.table = []
        self.value = ""

    def addTitle(self, title, cols = "2"):
        row = TableRow()
        row.addCell(title, colspan = cols, cssclass = "tableend")
        self.addRow(row)
        return
```

```
    def addLegend(self, text, cols = "2"):
        row = TableRow()
        row.addCell(text, colspan = cols, align = "center", cssclass = "small")
        self.addRow(row)
        return

    def addHiddenComponent(self, name, value):
        self.addHidden(name, value)
        self.formhead += self.components[-1:][0]
        return

    def addComponent(self, label, type, formarg):
        if type == "hidden":
            name, value = formarg
            self.addHiddenComponent(name, value)
            return
        row = TableRow()
        row.addCell(label, cssclass = "tablehighlight")
        if type == "text" or type == "file" or type == "password":
            self.addInput(formarg, type)
        elif type == "textarea":
            self.addTextarea(formarg)
        elif type == "select":
            name, args = formarg
            self.addSelect(name, args)
        elif type == "checkbox":
            name, args = formarg
            self.addCheckbox(name, args)
        elif type == "radio":
            name, args = formarg
            self.addRadio(name, args)
        row.addCell(self.components[-1:][0])
        self.addRow(row)
        return

    def addButtons(self, cols = "2"):
        row = TableRow()
        row.addCell(self.reset + self.submit, align = "right", colspan = cols, c
ssclass = "tableend")
        self.addRow(row)
        return

    def getvalue(self):
        content = ""
        for row in self.table:
            content += row.getvalue()
        self.value = self.formhead + self.tablehead + content + self.tabletail +
 self.formtail
        return self.value
```

```
class RepeatedForm(TableForm):
    def __init__(self, action, size = DEFAULTSIZE,
                 pad = 2,
                 method = "POST",
                 enctype = None,
                 border = "0",
                 cellpadding = "5",
                 cellspacing = "1",
                 bgcolor = "#cccccc",
                 align = "center",
                 width = None,
                 extra = None):
        self.size = size
        formatts = []
        formatts.append("action=\""+action+"\"")
        formatts.append("method=\""+method+"\"")
        if enctype != None:
            formatts.append("enctype=\""+enctype+"\"")
        self.formhead = " "*pad+"<form "+" ".join(formatts)+">\n"
        self.formtail = " "*pad+"</form>\n"
        self.components = []
        self.value = ""
        self.setSubmit()
        self.setReset()
        tableatts = []
        if border != None:
            tableatts.append("border=\""+border+"\"")
        if cellpadding != None:
            tableatts.append("cellpadding=\""+cellpadding+"\"")
        if cellspacing != None:
            tableatts.append("cellspacing=\""+cellspacing+"\"")
        if bgcolor != None:
            tableatts.append("bgcolor=\""+bgcolor+"\"")
        if align != None:
            tableatts.append("align=\""+align+"\"")
        if width != None:
            tableatts.append("width=\""+width+"\"")
        if extra != None:
            tableatts.append(extra)
        if len(tableatts) > 0:
            self.tablehead = " "*pad+"<table "+" ".join(tableatts)+">\n"
        else:
            self.tablehead = " "*pad+"<table>\n"
        self.tabletail = " "*pad+"</table>\n"
        self.table = []
        self.value = ""
        self.templates = []

    def addHeaders(self, headers):
        row = TableRow()
        for header in headers:
```

```
            row.addCell(header, align = "left", cssclass = "tablehighlight")
        self.addRow(row)
        return

    def addTemplateComponent(self, name, type, args):
        self.templates.append([name, type, args])
        return

    def computeForm(self):
        for i in range(0, self.size):
            row = TableRow()
            for t in self.templates:
                name, type, args = t
                name += "." + repr(i)
                if type == "text" or type == "file" or type == "password":
                    self.addInput(name, type)
                elif type == "textarea":
                    self.addTextarea(name)
                elif type == "select":
                    self.addSelect(name, args)
                elif type == "checkbox":
                    self.addCheckbox(name, args)
                elif type == "radio":
                    self.addRadio(name, args)
                row.addCell(self.components[-1:][0])
            self.addRow(row)

    def getvalue(self):
        content = ""
        for row in self.table:
            content += row.getvalue()
        self.value = self.formhead + self.tablehead + content + self.tabletail +
 self.formtail
        return self.value

def makeAnchor(uri, name = None):
    if name == None:
        return "<a href=\""+uri+"\">"+uri+"</a>"
    else:
        return "<a href=\""+uri+"\">"+name+"</a>"
```

12.9 Test Generator

```
#!/usr/bin/python2

def main():
    template = """
# input-%s-%s
@prefix log:     <http://www.w3.org/2000/10/swap/log#> .
@prefix here:    <#> .
```

```
@prefix :        <http://www.w3.org/2002/01/pedal/pedal#> .
@prefix dc:      <http://www.purl.org/dc/elements/1.1/> .
@prefix p3p:     <http://www.w3.org/2002/01/p3prdfv1#> .
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#> .
@prefix foaf:    <http://xmlns.com/foaf/0.1/> .

here:%s a :Policy ;
    :authoredBy :%s .

[ a :PolicyStatement ;
    :forPolicy here:%s ;
    :hasPriority %s ;
    :forResource %s ;
    :withVisibility %s ;
    :ruleSubject { :someone %s %s . } ;
    :hasComponent [ a :Component ;
                    :withPredicate %s ;
                    :withRange %s ] ] .
"""
    ap = [":Must", ":Should", ":Must"]
    pp = [":Should", ":Must", ":Must"]
    resource = "dc:title"

    for i in range(1,13):
        f = open("input-%s-1"%`i`, 'w')
        f.write(template % (`i`, "1", "MyDocumentPolicy", "Author",
                            "MyDocumentPolicy", ap[i%3], resource,
                            ":visibleTo", "contact:familyName", "\"Allen\"",
                            "contact:familyName", "\"Allen\""))
        f.close()
        f = open("input-%s-2"%`i`, 'w')
        f.write(template % (`i`, "2", "DocumentPolicy", "PolicyAuthority",
                            "DocumentPolicy", pp[i%3], resource, pv(i),
                            ppr(i), pr(i), ppr(i), pr(i)))
        f.close()

def pv(n):
    if n >= 4: return ":hiddenTo"
    else: return ":visibleTo"

def ppr(n):
    if n >= 7: return "contact:firstName"
    else: return "contact:familyName"

def pr(n):
    if n >= 10: return "\"Bishop\""
    else: return "\"Allen\""

main()
```

References

[1] Tim Berners-Lee. Transcript of talk. <http://www.w3.org/1999/04/13-tbl.html> April 14, 1999

[2] United States of America Federal Trade Commission. *Privacy Initiatives* <http://www.ftc.org/privacy/>.

[3] W3C. Platform for Internet Content Selection. <http://www.w3.org/PICS/>

[4] W3C. *PICSRules 1.1.* <http://www.w3.org/TR/REC-PICSRules> December 29, 1997.

[5] W3C. Platform for Privacy Preferences (P3P) Project. <http://www.w3.org/P3P/>

[6] W3C. *Removing Data Transfer from P3P.* <http://www.w3.org/P3P/data-transfer.html> September 21, 1999

[7] W3C. *A P3P Preference Exchange Language 1.0 (APPEL 1.0).* <http://www.w3.org/TR/2001/WD-P3P-preferences-20010226.html> February 26, 2001.

[8] Electronic Privacy Information Center, Junkbusters. *Pretty Poor Privacy: An Assessment of P3P and Internet Privacy.* <http://www.epic.org/reports/prettypoorprivacy.html> June, 2000.

[9] W3C. *An RDF Schema for P3P* <http://www.w3.org/TR/2002/NOTE-p3p-rdfschema-20020125/>.

[10] Michelle Delio, Danit Lidor. *BN.com: Insecure About Security?*. Wired.com News, August 1, 2002. <http://www.wired.com/news/privacy/0,1848,54251,00.html>.

[11] Ian Clarke. *A Distributed Decentralised Information Storage Retrieval System.* <http://www.freenetproject.org/freenet.pdf>. 1999.

[12] Marc Waldman, Aviel D. Rubin, Lorrie Faith Cranor *A Robust Tamper-Evident, Censorship-Resistant Web Publishing System.* <http://publius.cdt.org/publius.pdf>. August, 2000.

[13] Roger Dingledine. *The Free Haven Project: Design and Deployment of an Anonymous Secure Data Haven.* <http://www.freehaven.net/doc/freehaven.ps>.

[14] W3C. *Extensible Markup Language* <http://www.w3.org/XML>.

[15] T. Berners-Lee, R. Fielding, L. Masinter. *IETF RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax* <http://www.ietf.org/rfc/rfc2396.txt>.

[16] W3C. *Namespaces in XML* <http://www.w3.org/TR/REC-xml-names/>.

[17] W3C. Resource Description Framework. <http://www.w3.org/RDF/>

[18] W3C. *Resource Description Framework (RDF) Model and Syntax Specification.* <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/> February 22, 1999

[19] W3C. *RDF Model Theory (Working Draft)* <http://www.w3.org/TR/rdf-mt/>.

[20] Tim Berners-Lee. *Notation 3 (N3)* <http://www.w3.org/DesignIssues/Notation3/>. <http://www.w3.org/TR/2002/WD-rdf-schema-20020430/> April 30, 2002.

[21] Tim Berners-Lee. *Primer - Getting into the semantic web and RDF using N3* <http://www.w3.org/2000/10/swap/Primer.html>.

[22] W3C. *RDF Vocabulary Description Language 1.0: RDF Schema*.

[23] *The Dublic Core Metadata Initiative Overview*. <http://dublincore.org/about/overview/>.

[24] Eric Prud'hommeaux. *W3C ACL System* <http://www.w3.org/2001/04/20-ACLs.html>.

[25] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels* <http://www.ietf.org/rfc/rfc2119.txt>, BCP 14, RFC 2119, March 1997.

[26] W3C. *Peddler: PEDAL Generator* <http://www.w3.org/2002/01/pedal/peddler/>.

[27] IETF. *RFC 2527: Internet X.509 Public Key Infrastructure: Certificate Policy and Certificaton Practices Framework* <http://www.ietf.org/rfc/rfc2527.txt>.

[28] Ron Rivest, Butler Lampson. *Cryptography and Information Security: SDSI (Simple Distributed Security Infrastructure)* <http://theory.lcs.mit.edu/ cis/sdsi.html>.

[29] Carl Ellison, et al. *Simple Public Key Certificate* <http://world.std.com/ cme/spki.txt>.

[30] IETF, W3C. *XML-Signature Syntax and Processing* <http://www.w3.org/TR/xmldsig-core/>.

[31] Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana. *Web Services Description Language (WSDL) 1.1*. <http://www.w3.org/TR/wsdl.html> March 15, 2001

[32] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*. <http://www.ietf.org/rfc/rfc2616.txt>. June, 1999.