UMA Scenarios and Use Cases

Abstract

This document is a product of the User-Managed Access Work Group. It records the scenarios and use cases governing the development of the User-Managed Access protocol and guiding associated implementations and deployments.

Status

This document is currently under active development. Its latest version can always be found here. See the Change History at the end of this document for its revision number.

Editors

- Hasan Ibne Akram (lead editor)
- Gerald Beuchelt
- Domenico Catalano
- Maciej Machulak
- Eve Maler
- Christian Scholz
- Thomas Hardjono

Intellectual Property Notice

The User-Managed Access Work Group operates under Option Liberty and the publication of this document is governed by the policies outlined in this option.

Table of Contents

- Abstract
- Status
- Editors
- Intellectual Property Notice

Table of Contents

Introduction and Instructions
- Common Building Blocks (Dimensions)

Scenario: Sharing a Calendar with Vendors (Accepted)
- Travel Calendar Sharing with Vendors
- Soliciting Timely Interactions from Vendors
- Use Case: Separate Resource Host, Relationship Manager, and Recipient (Accepted)

Scenario: Packaging Resources for E-Commerce Vendors (Accepted)
- Problem scenario
- Desired improvements
- Solution Scenario
- Use Case: Online Purchase with Setup of a Long-Running Account Relationship (Accepted)
- Issues
- Use Case: Engaging in a Purchase "One-Night Stand" (Accepted)

Scenario: Online Personal Loan request scenario (Accepted)

Scenario: Distributed Services (Pending)
- The story
- The Service Catalogue idea
- Distributed Authorization

Scenario: Controlling Two-Way Sharing of Location Information (Pending)
- Actors
- Short Description
- Dimensions
- Scope Detail
- Assumptions and Preconditions
- Use Case 1: Alice Sets Up AM Protection Over Location Information at HotLocale
- Use Case 2: Alice Shares HotLocale Location Access with RovingAround and RoadWarrior
- Use Case 3: Alice Monitors and Controls Location Information Access from Her AM
- Issues

Scenario: Requester Delegate (Accepted)
- Use Case: BizTools Impersonates BizService (Accepted)
- Use Case: BizTools Provides Its Own Network Endpoint (Rejected)

Scenario: Managing Information in Which Employers and Employees Both Have a Stake (Pending)
- Use Case: Employer as AM and Host (Pending)
- Use Case: Employer as Host (Pending)
- Use Case: Employer as Requester (Pending)
Introduction and Instructions

This document is a product of the User-Managed Access Work Group. It records the scenarios and use cases governing the development of the User-Managed Access protocol and guiding associated implementations and deployments, and outlines technical issues raised thereby.

Please copy and revise an existing scenario in adding new scenarios and subordinate use cases. Each scenario is created as a separate child wiki page with a name like xyz_scenario and then linked from here. Change the status keyword in each scenario and use case title as appropriate:

- **Pending**: Initial status when first submitted
- **Accepted**: Needs to be accounted for in UMA V1 and/or its associated compliant implementations
- **Deferred**: Relevant to the problem space; may be considered in future versions
- **Rejected**: Out of scope

Edit the descriptions of technical issues and scope questions to reflect (or point to) group decisions about how to handle them.

Common Building Blocks (Dimensions)

As a further refinement to help us in classifying and prioritizing the use cases, we would like to add a section to each use case describing the building-block features or dimensions that are present in a given use case. The current list of features or dimensions are as follows:

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>

Change History
<table>
<thead>
<tr>
<th>Nature of protected resource</th>
<th>A description of the sorts of protected resources at hosts in this scenario, and the scope values that might be applicable, ideally with real-life supporting evidence. Protected resources appear to fall on a continuum from API endpoint (such as status updates) to content-oriented (such as photos), and further, typical actions on them may usefully be classified in terms of how RESTful they are.</th>
<th>In the location services scenario, protection of a location service (a set of one or more API endpoints) might involve scope values such as “write location data”, “read precise location data”, and “read location data at a city level”.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sharing models</td>
<td>A description of what sorts of access or sharing the scenario facilitates. Person-to-self sharing occurs when an authorizing user shares access with a service that is acting directly on the same user's behalf (a la “three-legged” OAuth). Person-to-service sharing occurs when an authorizing user shares access with a legal person operating a service that is acting on its own behalf (a la “two-legged” OAuth, where the client is autonomous). Person-to-person sharing (also known as “Alice-to-Bob sharing”) occurs when an authorizing user shares access with a different natural person. (@@Fourth category of “person-to-rep sharing” where the autonomous company’s service is operated by a human company rep?)</td>
<td>The location services scenario is an example of person-to-self sharing. The calendar scenario is written to explore person-to-self and person-to-service (@@and person-to-rep?) sharing options, but could also apply to person-to-person sharing.</td>
</tr>
<tr>
<td>Nature of policies and claims</td>
<td>A description of the types of policies, and any resulting claims requested, that might be suitable for this scenario and its use cases.</td>
<td>The terms negotiation scenarios discuss likely claims needed in the course of assessing a requesting party's right to get access.</td>
</tr>
<tr>
<td>Cardinality</td>
<td>An accounting of whether the scenario or any of its use cases necessarily involve multiple instances of any of the UMA entities.</td>
<td>The financial loan scenario by its nature involves a requester having to access multiple protected resources, likely from different hosts. The scenario related to moving resources by its nature involves two different AMs: the user's previously chosen AM and their newly chosen one. This information is often usefully conveyed with an architectural diagram along with descriptive text.</td>
</tr>
<tr>
<td>Colocation</td>
<td>A description of likely cases where real-world applications might want or need to support multiple UMA endpoints.</td>
<td>The health data scenario tends to involve actors that serve as both hosts and requesters. The trusted claims scenario proposes that an application offering AM services might also need to be a requester in order to access the UMA-protected claims of some other (“primary”) requester.</td>
</tr>
<tr>
<td>Host-AM relationship</td>
<td>An accounting of how hosts and AM come to meet and trust each other in this scenario or its use cases. Dynamic discovery might be required or forbidden. AMs and hosts respectively might need to be well-known, or at least dynamically qualified before the connection is made.</td>
<td>The health data scenario might have a short list of approved AMs to which many hosts around the world may need to dynamically connect as soon as a new patient needs medical care.</td>
</tr>
<tr>
<td>Protected resource discovery</td>
<td>A description of the anticipated method(s) by which a requester learns about a resource of interest to them. The methods may have a dynamic element to them or may require reconfiguration. The methods may or may not involve direct human assistance.</td>
<td>The calendar scenario could mention that most calendar feeds are shared today through URL copying and pasting.</td>
</tr>
</tbody>
</table>

---

**Scenario: Sharing a Calendar with Vendors (Accepted)**

Submitted by: Eve Maler

Online calendars are an example of personal data that is readily shared with other people in a manner that evokes VRM paradigms. Because calendar data is fairly volatile, static calendar snapshots are rarely shared; rather, a calendar feed is provided and authorized recipients can pull fresh calendar data as required. The data is often considered sensitive and is expected to be kept secure, hence "private URLs" and (minimal) ACL features offered by Google Calendar and other hosts.

In this scenario, personal online calendars are shared with "vendors" (online services) rather than other individuals, and they are shared in such a way as to allow permissioning and auditing from a central location rather than wherever the calendar is hosted. For the purposes of this scenario we’ll focus on sharing a single online calendar (such as for "work", "soccer", or "travel") as a unitary Web resource, on an ongoing basis, with one or more individually-authorized recipients.

User interface mockups of a calendar-sharing interaction can be found in the initial blog post made about ProtectServe and, in somewhat more sophisticated form, slides from a speech made at an identity conference.

Following are some motivating circumstances in which calendar-sharing with vendors may make sense. (NOTE: All references to real vendors are hypothetical.)

**Travel Calendar Sharing with Vendors**

Alice, who is based in the Seattle area, has an online calendar that specifically contains business travel details such as flights, hotel stays, and car rentals, and since she travels quite frequently and often to international destinations, she wishes to share it with the following vendors:
• Her Visa credit-card company, Chase

Often when she tries to charge European hotel stays to her Chase Visa, the credit card company denies the charges or asks the hotel desk clerk to put her on the phone to make sure it's really her flitting around Europe and racking up big hotel bills. To let Chase know ahead of time what her travel plans are, Alice decides to share her travel calendar with them on a long-term basis so they can know ahead of time that it's likely truly Alice who's putting a Barcelona hotel stay on the card.

Note that this recipient of her data already has a lot of quite personal and sensitive information about Alice, so she's fairly comfortable giving them access to this data under certain conditions, such as refusing to accept third-party direct marketing.

It must be possible for Alice to cut off the flow of travel calendar data to Chase (even though she continues to use that card for personal purchases) when Alice is told that she has to begin using a corporate AmEx card for all business travel purchases.

• The Seattle Times newspaper delivery service

She'd like to avoid having to go to their website to put her newspaper delivery on hold every time she travels. By sharing a travel calendar with the delivery service that accurately reflects when no one will be at home, she saves one more to-do item as she prepares for each trip.

This is data she would have had to share with the service "manually" anyway (by filling out a web form or using the phone), so she already had to trust the service not to rob her house while she's away. It's likely her full travel calendar contains more data than the service strictly needs, however.

• The U.S. Postal Service

Instead of having to go to the Post Office or its website to fill out a mail hold form, she wants to let them know automatically. This is very similar to the Seattle Times situation, but in our project we need to solve for being able to attach different data-sharing policies and possibly have a different data-sharing lifespan between the two.

• Her mobile carrier, T-Mobile

Alice would like to be offered the option to purchase pre-paid roaming minutes when she travels overseas. By sharing her travel calendar, she can let T-Mobile know that she'll be in Brazil next month and would welcome a special offer on mobile roaming. (Note that this use case has an element of volunteered personal information to it; by positively choosing to share her information, Alice gets new opportunities to transact with vendors.)

• Her travel data social-networking sites, Dopplr and TripIt

Alice wants to keep all her "source" travel information in one place, but some of her friends and colleagues use different Web 2.0 sites to share such information. Rather than re-input all her travel destinations into Dopplr and TripIt, she'd like to let them pick up her planned locations and trip dates from her travel calendar.

Today, Dopplr and other similar sites often use OAuth to share such information, but the actual data passed isn't standardized, and the protocol for creating that long-term connection between the sites is OAuth. (See the forthcoming scenario Granting Service Access to a Photo Set for more observations on this flavor of scenario.)

Soliciting Timely Interactions from Vendors

Alice happens to work from home. Her typical work day is very busy, and she rarely has time to sit on hold when calling the various vendors in her life. She has a calendar that exposes the times during the day when she is free to accept a phone call or consider an invitation to a meeting or other event. She would like to share this information with the following vendors:

• Her TV cable carrier, Comcast

Alice's TV cable system has stopped working, and she needs to have a Comcast repairman come over to the house to fix it. She's too busy to spend time jockeying with the customer support person on the phone about which three-hour period she might be free, so she decides to let Comcast get a limited view into her potential free times so they can send her an event invitation for a repair slot.

• Her general-practitioner doctor's office

Alice needs to talk to the medical assistant in her doctor's office, but it's impossible to get hold of her. The MA calls while Alice is on a telecon but the MA can't leave a substantive message because of HIPAA laws/fears, and then when Alice calls back, of course the MA is in the middle of making a series of other calls and can't be reached. It's a "telephone tag" nightmare. Alice would like to share her free/busy times for the next few days so that the MA can at least pick a likely time to call her successfully.

Use Case: Separate Resource Host, Relationship Manager, and Recipient (Accepted)

Submitted by: Eve Maler

The most generic possible configuration of protocol endpoints solving this scenario is to have one service hosting the calendar in question, a different service getting permissioned read access to it, and yet a different service functioning as the authorization manager, all of them "in the cloud" from the perspective of the user and all operating on the open Internet rather than on a corporate intranet (since our user is an individual acting on her own behalf). This configuration is illustrated below.
Scenario: Packaging Resources for E-Commerce Vendors (Accepted)

Submitted by: Eve Maler

This scenario focuses on the typical set of information that we hand over to online vendors repeatedly, and the desire to avoid sharing the data "by value", instead focusing on how to share it "by reference" (pointers).

Problem scenario

Let's look at how an online buying scenario might look today.

1. Bricks and mortar

   Maya recently became extra-concerned about identity theft and fraud because a friend had his bank account stolen from, and she has decided to buy a shredder so she can dispose of old bills, credit-card offer junk mail, and outdated backup CDROMs. She visits her local Staples store, prepared to buy a shredder that day (with cash! hey, it's anonymous), but can't find a shredder in stock that handles CD shredding.

2. Comparison shopping

   She goes to Google and a couple of specialized comparison-shopping sites and plugs in search terms like paper shredder cdrom, but can't easily figure out which ones have the features she wants, but the prices at Staples.com, the site for the local store she was just in, look good and she decides to just go ahead and shop there.

3. Clicks and mortar

   Once Maya is at Staples.com, she finds a suitable shredder and adds it to her shopping cart (which is, so far, "anonymous" with respect to everything but some sort of device identification, possibly based on a cookie, associated with her browser session).

4. Checkout

   When she goes to check out, Maya is asked for consent and personal data for various purposes. First, she must choose a username and password, on the theory that this will make her future purchases at the site easier. She also has to provide her home address and phone number (though this isn't so onerous because her browser auto-fills the data) so Staples can transfer the shredder to its outsourced shipping company for delivery, and her credit card number, its security code, its expiration date, and her real name (the name the card was issued to) so Staples can be paid for the purchase. She might be given the opportunity to provide some third-party store loyalty program information, to get "extra points" from transactions here. Finally, she is asked to click "I Agree" certifying that she agrees to Stapler's site terms of use and has seen its privacy policy.

Desired improvements

Following are some key questions we can ask, identified by whether they capture an identity management (IdM) issue, a vendor relationship management (VRM) issue, or a social networking issue. (Note that some of these questions highlight scenarios and use cases that the calendar scenario has already captured. Some of these might want to get turned into unique use cases for this scenario.)

- Can we imagine better ways for Maya to set up a data-sharing relationship with Staples.com? (IdM, VRM)
  - She's planning to move in a couple of months, and that means the address information Staples has saved will go stale.
  - Same for her credit card: it will expire next year. When these items change, she has to go fix them at dozens of sites.
  - She's not crazy about having to supply things like credit card information to every vendor on the web.
• She thought the site terms and privacy policy were just "okay", but accepted them because she effectively has no choice – and OfficeArmory.com is probably the same anyway.

• Is it possible for Maya to have a "one-night stand" with Staples.com rather than a long-running relationship? (IDM, VRM)
  • ...if she doesn't really want Staples to track her purchases, browsing habits, or anything else over time.
  • ...if she wants to share only the minimum personal information Staples really needs to do its job this once, and then only temporarily.
• Can we imagine better ways for Maya to engage in the shopping-around process, possibly involving her sharing more data about herself? (VRM – particularly volunteered personal information?)
  • What if she could "issue a personal RFP" indicating the price and features she's interested in, and entice vendor site "bids", such that not only Staples.com and OfficeArmory.com but also Ann, who has a used shredder she'd like to sell?
  • What if she could let Staples know her customer-support phone line preferences, such as wait time and ad-playing tolerance?
• What would it look like for Maya to get a unified understanding of all of her data-sharing relationships? (VRM)
  • She sure would like to get a handle on her own "personal data analytics" – "who knows what" about her.
  • If Staples behaves badly (gives out her data against her rules or allows a data breach to occur), she wants to have better options for recourse.
  • ...and she wants to be able to cut off their future access to information about her.

Solution Scenario

Maya shares the information about herself that Staples.com needs at the beginning of her e-commerce relationship with them, but instead of having to share it "by value", she shares it as some form of pointer to a package of resource pointers that Staples can dereference and refresh as they need to over time. She can change the underlying information whenever she wants to without worrying about paying special attention to Staples (or any of the other hundred e-commerce sites with which she has registered).

Actors:

• Maya (User)
• Authorization manager (AM)
• Personal datastore (Host) in which authoritative versions of resources to be shared reside (that colocation of AM and Host is not a requirement, but for this scenario the individual resources are assumed to live on a single Host)
• Staples.com (Requester)

Distinctive aspects:

• User can package and reuse pointers to resources commonly needed for e-commerce into a rolled-up resource that is available for access by multiple requesters (assume for this scenario that all the individual resources and the rolled-up resource are available from the same host in the "personal datastore" model).
• The data involved is "self-asserted" to a first approximation. (The credit card data we often share today is "asserted" solely by us, but then the vendor validates it out of band.)
• Requester can handle receiving and dereferencing both a pointer to a package resource and pointers to individual resources.
• AM can manage the offering and meeting of terms for resource-sharing for the whole package and can take advantage of efficiencies where the terms for individual resources are identical (possibly similar to the Distributed Services scenario).
• Requester will often represent a requesting party who is the same human being as the authorizing user, and can take advantage of efficiencies in any real-time requester/AM/user connection for obtaining user consent in that moment.
• Requester, host, and AM are likely to be willing to deal with each other solely on the basis of the user's say-so (unlike in the personal loan scenario).

Use Case: Online Purchase with Setup of a Long-Running Account Relationship (Accepted)

Submitted by: Eve Maler

Preconditions: Maya has already stored, and packaged together, pointers to the set of relevant resources frequently needed for online purchases:

• Her desired credit card number, expiration month and year, security code, name on card, and billing address
• Her shipping address and phone number

So...

1. She goes to Staples.com and puts her desired shredder in her shopping cart.
2. When she goes to check out, she's presented with a requirement to register for an account. The form has fields for the data listed above, but also has a new field for "Personal data feed".
3. In a separate browser window, Maya visits her Relationship Manager and generates a unique URL representing the disclosure package she wants to offer Staples (CC number etc.) and the policies she expects it to adhere to in accepting her information (they can't sell her data etc.).
4. She goes back to Staples and pastes the URL she just generated, and presses a button that says "Share personal data".
5. The Staples web application follows the link, discovers it has to agree to a set of policies before getting through to the info it needs, decides to agree, and gets through.
6. Staples retrieves data items with content-types that indicate they contain CC numbers, addresses, etc., and then displays the values retrieved in the regular registration form fields, possibly with some graphical indication that Maya can override any one of them.
7. (Later) Maya can use her RM to view the activity related to Staples' retrieval of her information, check what she's told them (and others), and also check the conditions under which she released information.

Issues

• How can we ensure that the sensitive data is secured in motion (while being conveyed to Staples)? (generic across all scenarios)
• What is the right UX paradigm for letting Maya override information? Ideally, any info that's changed should be updated in the RM, not on the Staples site. But if she wants to override a value just once, with values to be updated in future pulls of the feed, the right place to change it is on the Staples site itself. Does this latter situation sound likely?
Use Case: Engaging in a Purchase "One-Night Stand" (Accepted)

Submitted by: Eve Maler

This is the same as the first use case outlined above, except that Maya provides her resource package not as part of a request to register a new account, but as part of a one-time purchase. (Some websites today allow for purchases without registering, and are prepared not to give you a browser cookie or retain your information beyond necessary for the purchase and its aftermath.)

The policies Maya chooses in this case are likelier to be more stringent about not retaining personally identifiable information (PII) for any significant length of time, and may ask the vendor to generate "positive" assurance messages about policy adherence (not just silent adherence).

(The protected-inbox scenario might play an especially important role if Maya is engaging in a one-night stand purchase, since it enables the vendor to report product recalls and such to Maya without her having to expose other more compromisable communications endpoints such as persistent email addresses or phone numbers.)

Scenario: Online Personal Loan request scenario (Accepted)

Submitted by: Domenico Catalano

(Also see the scenario slide deck presented by Domenico at UMA telecon 2009-12-17.)

The Economy downturn and the Financial institution crisis have introduced new needs to increase the control on loan or mortgage request process verifying user credit information and user status, in order to reduce financial risks.

Online Personal loan request is a specific use case in which a user/customer apply a request for a personal loan to a financial service. To approve or reject the loan request, the financial service must verify many user personal information from different Service Provider/host. For instance, the amount of monthly user salary (i.e. 3 last monthly salary) from user's Employer, user bank account information (account number, net) and need to access to the user credit information (credit history, score, ect.) from the Financial Risk central service.

Today, human interactions are based on online access to Financial Risk Central Service, by Financial service operator, or operator telephone call to user's bank to verify account or to the user's Employer to verify employee status, ect.

Distinctive aspects:

- A Consumer (requesting service) that needs a collection of information from multiple sources
- A high-value, privacy-sensitive transaction
- Ensuring that information about the user is third-party verified by using the third parties directly as SPs/Host

Actors:

- User as Authorizing User
- Financial Service as Requesting Service
- User's Employer as Host (salary information)
- User Bank as Host (user account information)
- Financial Risk central service as Host (user credit information)
- Authorization Manager
Scenario: Distributed Services (Pending)

*Submitted by:* Christian Scholz

The story

(*this text is just to describe one possible environment in which distributed or mass authorization of services is useful)*

If you look at the social networking scene today one thing is obvious already: There is lot of data online on various services and much of this data is redundant because it is available in various copies which are usually not synced. The main area for this problem is probably profile and friendship/contact information. On each social network or service you register you usually have to enter your profile information again and try to find your contacts.

With the advent of more and more of such social services the amount of redundant data will grow even more and this will lead to a acceptance problem.

The Service Catalogue idea

It is unlikely that users will centralize all their data in one place. It's more likely that data will be distributed even more. So one problem might already be to manage all the places where data is stored about you or where services can provide functionality on your behalf. One solution to this might be a concept called "Service Catalogue" which came up in discussions in the open web/DiSo/DataPortability communities. The basic idea is to have a list of all these places stored under your control which can be queried by services.

Another point is that for reducing the amount of copies of your data it is necessary to link to your data instead of copying it (or even worse asking the user to type it in again). The Service Catalogue can serve basically as such a link list where each service/type of data is marked up with a location (URI) and type (probably another URI). Obvious things to link to are your profile and contact list but other things make also sense, like photos, videos, blog posts, recommendations, your attention profile, travel information and much more.
Having this catalogue you can easily tell a new service which other services you already use by simply pointing it to the Service Catalogue:

Note though that this Service Discovery is out of scope for the work at UMA and only serves as an example of how to obtain a list of services to authorize later on. Another method apparently is the user typing in various URLs which is not that user friendly though.

The result is in any case a list of services you want to authorize.

Distributed Authorization

The problem is how you authorize that new service to get access to all the other 3rd party services. OAuth is one possible solution but at least if the default mechanism for retrieving a token is used this means that the user has to be redirected to each of these 3rd party services in order to give consent for the new service to use that data.

Moreover OAuth does not contain a mechanism to define what permissions should be used on the service endpoints. This can be done individually by each service but having a central place for such policy decisions and being able to store policies and share them among services might be beneficial as well.

An example for such profiles would be that you can filter which fields of your profile a PortableContacts endpoint actually gives out to a certain consumer.

For the sake of usability what we want is a single page where you can define the relationships between that new service and all the other services you have access to.

This could look like this:
Additionally a user should be able to quickly revoke tokens again in a central location as well as getting an overview of which services have access to which other services under which policies.

**Dimensions**

- **Scope:** This use case involves the Scope dimension, as the new service added to (or through) the Service Catalogue may need to be given the Scoping information pertaining to all the existing services already listed in the Catalog. Furthermore, the syntax and semantics of the scopes (scoping rules) will need to be interoperable between all the the services listed in the Catalogue.
- **Resource Discovery:** This use case relates to resource discovery in that the Service Catalog acts as a service directory from which external entities can learn about services relevant to the user. (NB. privacy issues).

---

**Scenario: Controlling Two-Way Sharing of Location Information (Pending)**

Submitted by: Eve Maler

**Actors**

- Authorizing user Alice, who chooses to set and share her current location through various applications that she accesses with a commercial off-the-shelf browser
- An application that serves as an **Authorization Manager** (AM) on Alice's behalf, orchestrating which location applications can write to and read from other applications
- Web-based location applications called **HotLocale** (H1) and **HipHappeningPlaces** (H2) that are Hosts of location information on Alice's behalf
- Web-based location applications called **RovingAround** (R1) and **RoadWarrior** (R2) that are Requesters of location information on Alice's behalf

**Short Description**

Today's location services such as FireEagle, BrightKite, and Dopplr let Alice set her location within any one of several applications, and then use OAuth-enabled connections to propagate that information through other such services. Since Alice can end up "chaining" services this way quite easily, with a thicket of pairwise connections, it's valuable for her to know and control where this information is flowing to. In this scenario, a user of HotLocale and RovingRound wants to arrange to connect them up so that her location can be propagated among them, and she wants to get a global view through her AM about who's allowed to do what, so she can change and stop permissions in a coordinated way.
**Dimensions**

<table>
<thead>
<tr>
<th>Nature of protected resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API endpoint. For example, see the FireEagle developer documentation. The resource being protected is a single endpoint URL, with different options and parameters possible for using it. The possible scopes in today's location services tend to include (a) whether the requesting app (client) can write the user's location, and (b) to what degree the client can read the user's location. For example, see the FireEagle scope permissions documentation; the location-reading options include: no read access, as precise as possible, postal code level, neighborhood level, town level, regional level, state level, and country level.</td>
<td></td>
</tr>
</tbody>
</table>

| Sharing models | Current OAuth-protected usage of location services such as FireEagle and BrightKite assumes a **person-to-self** sharing model, where Alice has login accounts at the various host and requester apps, and wants to instruct them to share her location among themselves on her own behalf; an UMA-enabled version of this would allow her to control such sharing centrally at her AM. |

| Nature of policies and claims | For person-to-self sharing, a typical policy might require the requesting party to be able to authenticate as Alice directly at the AM (that is, dictating a particular Step 2 web-server flow that uses normal OAuth user authentication), or might require a claim that it is an authenticated Alice asking for forge the connection. Further, a policy might set (unilateral) boundaries on which scopes can be accessed. |

| Cardinality | This scenario is 1:1. There is no need to aggregate multiple resources from multiple hosts, for example. |

| Colocation | The same requester app might both read and write location data (by using different methods with the same API endpoint), which allows it to write data to a host while still remaining a requester (think "requester of API access"). With today's OAuth-enabled location services, the services act relatively "peer"-ish, in that they have each done the necessary integration to support any one of them serving as an initial host with the others being requesters. So it could be said that the same app might be a host for some users and a requester for others, and/or might be set up as both a host and a requester for the same user's location; however, the latter may be very confusing given current OAuth realities. |

| Host-AM relationship | Today's OAuth-enabled apps require static introduction and configuration, but the light weight and low security of these Web 2.0 services suggest that dynamic introduction is a possibility as long as the location API is well-known/standardized. |

| Protected resource discovery | Today's OAuth-enabled apps advertise this location in the process of static introduction and configuration, but with a well-known/standardized location API, it seems possible that a location service host could advertise its endpoint through dynamic means such as hostmeta/XRD. |

**Scope Detail**

Below is a screenshot showing that FireEagle and Dopplr have the capability today to have two-way location information flow. Our user wants to be able to see this "combinatorily", for all connected location services and indeed for all such services on the web that she chooses to use for hosting any data or content.
Assumptions and Preconditions

This scenario assumes that Alice has an account at each of AM, H1, H2, R1, and R2. These accounts might or might not be driven off of federated login, for example, Alice might log in to HotLocale through Google and RoadWarrior through Facebook.

This scenario assumes that AM, H1, H2, R1, and R2 are all UMA-enabled but have otherwise not met before. (Simplified circumstances that assume prior introduction are described in their turn below.)

This scenario assumes that H1 and H2 use a hypothetical standard location API that R1 and R2 are configured to understand, and that H1 and H2 expose APIs through URLs that differ per user in some fashion (e.g., through a URL query parameter or through a part of the URL path).

Use Case 1: Alice Sets Up AM Protection Over Location Information at HotLocale

This flow can be embedded in other flows, or can be standalone.

1. H1 asks Alice to choose an AM to protect her location information in H1
2. Alice tells H1 her preferred AM
3. H1 discovers how to get started connecting to the AM
4. Alice is redirected to AM to log in as Alice-on-AM, consent to protection, and choose a policy that applies to H1’s endpoint for her
   • The policy indicates that the requesting party must accept her chosen scope of access, which is "write, read-city"
   • The policy indicates that the requesting party must be able to log in at the AM synchronously as Alice-on-AM and consent
   • The policy does not otherwise discriminate against particular Requester apps (note that we don’t currently have a way to do this; do we need to?)
   • She labels the policy “location services policy” (outside the scope of the UMA protocol)
5. Alice is redirected back to H1 to continue what she was doing before

The identical sequence can be done with HipHappeningPlaces (H2).

Use Case 2: Alice Shares HotLocale Location Access with RovingAround and RoadWarrior

1. Alice visits H1 and logs in as Alice-on-H1
2. Alice decides to share access to her location information in H1 with R1 (if use case #1 has not been done, it must be done at this time)
3. Alice asks H1 to give her a "share" URL to use
4. Alice visits R1 and logs in as Alice-on-R1
5. Alice gives R1 the "share" URL from H1 and asks it to set up access to her location information there
6. R1 attempts to access the URL and discovers H1 uses a location API that R1 understands
7. H1 redirects R1 to the AM as unauthorized
8. AM tells R1 it needs to provide a claim acknowledging the offered scope (need to standardize this claim request/response)
R1 sends the claim to AM
AM tells R1 it needs to redirect its user (Alice) to AM to authenticate
R1 redirects Alice to AM to authenticate and consent
AM concludes that Alice authenticated correctly as Alice-on-AM
AM issues an access token to R1 for location information access of the offered/accepted scope and redirects Alice back

The identical sequence can be done with RoadWarrior (R2), except that once HotLocale is introduced to the AM, it never needs to be introduced again, so the optional embedded UC1 block isn’t ever done again unless Alice wants to switch AMs.

Use Case 3: Alice Monitors and Controls Location Information Access from Her AM

To monitor access, Alice interacts with value-added functionality provided at the “data-sharing relationship manager” application that serves as her AM endpoint; that is, this use case does not involve standardized UMA protocol behavior. These interactions might include:

- Looking at a unified log for occasions when both RovingAround and RoadWarrior accessed HotLocale on her behalf, and what they did on those occasions
- Looking at a unified log for occasions when RovingAround accessed both HotLocale and Alice's calendar application (not part of this scenario) on her behalf
- Looking at a unified log for all hosts to whose resources she applied her “location services policy”, including both HotLocale and HipHappeningPlaces
- Showing a graphical depiction of location host and requester access relationships and their types, if it is able to characterize hosts according to the APIs/resource types they expose

To control access, Alice may take various actions:

- If Alice discovers that her HotLocale account was hacked, she can go into her AM and revoke access to it from all requesters. If access tokens issued to requesters are sufficiently short-lived, the next time any of them seeks access by presenting a refresh token in order to get a new access token, they would be denied access. (would the OAuth 2.0 token revocation extension help here?)
- Alice may decide that she wants to limit access to include reading her location from HotLocale only, instead of reading and writing her location, because the previous situation was too confusing. She can change the “location services policy” to limit the scope list. If access tokens issued to requesters are sufficiently short-lived, the next time any of them seeks access by presenting a refresh token in order to get a new access token, they would be asked for a claim acknowledging the reduced scope and be issued a new access token.

Issues

- **Scope claim**: Need special claim request/response regarding some scope list being offered by the authorizing user, as imagined above?
- **RESTful API**: Do we really need the assumption that the protected resource URL has something special and authorizing-user-specific about it (with the invention of the “share URL” concept above)? Note that the share URL doesn’t have to be a secret URL; it should be protected by UMA regardless (we’d need to think about how the host registers resources if we do make the “share URL” assumption). Is there some way we can handle current common API practice, where the endpoint URL is static and doesn’t change per user?
- **Host API dynamicism**: Use case 2 could be called “host-initiated access”; this only works if the host uses a standard API that the requester can discover and work with. Do we need to build an alternative use case that assumes a proprietary API (akin to current OAuth usage), non-dynamic introduction of the parties, and requester-initiated access? How does scope work in that case – does the AM just end up recording the requester-offered scope?
- **Redirection of requesting user**: Do we need to bake a special claim request/response into UMA for handling the person-to-self flow (or at least usage of the web server user-authentication flow) in Step 2 so that the requester knows for sure to redirect the user to the AM? A requesting user who isn’t Alice may not need to “authenticate” there, but they could (e.g.) provide a special code that was given to them by the authorizing user offline.

Scenario: Requester Delegate (Accepted)

Submitted by: Michael Hanson

The Requester may be using a hosted service, which may need to make requests on its behalf.

The user has entered a relationship with BizService, and wants to authorize it to access her calendar. BizService is using a website hosted by BizTools, which is the entity that will initiate all network activity and actually hold the tokens generated during the transaction.

The user should be able to authorize BizService to access her data, without granting any privileges specifically to BizTools, and without granting privileges to any other company hosted by BizTools.
Issues:

- Does the user need to be aware of BizTools, or can she grant authorization to BizService in a way that allows BizService to relay access?
- Does this scenario require an explicit model of delegation enforced by the AM, so that BizService can't hand off an access token to anybody they want?

Use Case: BizTools Impersonates BizService (Accepted)

Today, app-hosting relationships commonly involve sharing of private keys (covered by service-level agreements), and concomitant "impersonation" of the company by the outsourced service. This use case would seem to be transparent to the user (for example, if the user is given a real-time opportunity to consent to access when BizTools/BizService attempts it, the request will appear to come from BizService) and to the UMA protocol.

Use Case: BizTools Provides Its Own Network Endpoint (Rejected)

If BizTools approaches the resource not by impersonating BizService but in its own right (on BizService's behalf), true delegation would somehow have to come into the protocol picture. The goal would be to avoid creating an "omnipotent token" that allows the proximate Requester (BizTools) to use the token for access on the behalf of other parties. As discussed on 2009-10-08, we are inclined to reject this use case.

Scenario: Managing Information in Which Employers and Employees Both Have a Stake (Pending)

Submitted by: Eve Maler

Both an employer and their employees might want to impose their own constraints on the sharing of the same employee-related resource. Examples of pieces of information your employer holds that you might want to share with others:

- Employment status (e.g., active or inactive; often needed when you apply for a loan)
- U.S. Internal Revenue Service W-4 (tax withholding) form details (handy for sharing with accountants and investment planners)

Some additional ones listed in the Liberty ID-SIS Employee Profile Service specification:

- Employee ID internal to enterprise
- Date of hire
- Job start date
- Employee type (e.g., part-time or full-time)
- Internal job title
- etc.

The following "user stories" capture the distinctive aspects of this scenario:

- As an employee, Alice wants to audit and control the further dissemination of information her employer must know about her as a condition of employment.
- As an employer, BigCo wants to adhere to laws and best practices regulating its sharing of information about its employee.
See the use cases below for the different configurations in which the actors might appear.

**Issue:** In large companies, typically the function of verifying someone's employment is outsourced to a specialized company. The employer is still seen as authoritative for employment status and other such data, though. For such information, where in the use cases below the employer is assumed to be the authoritative Host, perhaps the employer needs to provide a pointer to the employer's chosen verification service resource, such that the resource being shared is a pointer to a pointer (double indirection). Or perhaps the employer instructs Alice to introduce her AM to the real Host directly.

**Use Case: Employer as AM and Host (Pending)**

*Submitted by:* Eve Maler

Here, the employer runs an employee profile self-service application that could include both AM and Host functionality. The AM could let Alice configure her sharing policies, but could also let Alice know that it will be enforcing additional constraints out of band with respect to UMA.

This is probably a "legacy" solution because it forces the employee to seek out other relationship managers in the outside world where they're just an individual rather than an employee, and it seems the employer would be hosting the AM only for corporate inertia (admittedly, a force to be reckoned with).

**Use Case: Employer as Host (Pending)**

*Submitted by:* Eve Maler
For information for which the employer is authoritative ("Is this person employed here?"), it could offer a Host willing to attest to this on request (in accordance with the instructions issued by Alice's personal AM). If the employer doesn't want to release the data even though the employee wants to allow the sharing, it could use existing access control mechanisms that are out of band with respect to UMA.

**Issue:** Should the employer-Host surface a response code to the Requester that reflects this type of refusal? Should it provide audit-log data back to the AM?

**Use Case: Employer as Requester (Pending)**

**Submitted by:** Eve Maler
For information that Alice already self-asserts to the employer (“What is the employee’s home address of record?”), the employer should ideally consume this data in the same way some other “vendor” (online service) on the open Internet could. If the employee moves, a number of workflow actions have to unroll on the employer’s side as they would have anyway (in the U.S., moving to a different state might involve withholding a different amount of state income tax), but this is already handled in existing systems when the employee provisions the new information into employee profile apps by value. An on-board “personal datastore” Host is shown here with the user’s chosen AM, but the Host could just as easily be remote.

Scenario: Delegating Access Management to Custodians (Pending)

Submitted by: Maciej Machulak

Social applications become increasingly important for a large part of the society. Young and mature Internet users participate in social networks and exchange information about their personal or professional activities. They create connections with friends or other professionals. They share their personal information and digital content using various social applications.

Young people, in particular, have little knowledge about technical complexities of those applications. They have little understanding about the value of information that they submit and share among their peers and other users of those applications. Personal information such as age, sex, telephone numbers or hobbies is often not perceived as valuable. Similarly, other digital content such as photos, short video clips or documents is viewed as any other information which can be freely available for other users of social networks.

In reality, information submitted by users of social networks may be of great value to third parties. Personal information is often used for advertising purposes or can be abused by malicious users for other purposes. Digital content, on the other hand, has influence on how a particular individual is perceived by others, be it employers or peers. As such, restricting access to information and ensuring one’s privacy is a necessity. Younger users of social networking applications may not be aware of the above mentioned security and privacy issues related to the information that they submit. As such, they may expose too much information, which is not desirable. It is often the case that information is shared only with members with whom a person has a direct connection in a social network. However, to increase popularity by having many so-called friends, many users of those applications make connections with others even if they do not know them in person.

To prevent information leakage, parents often require having insight into what information is submitted and shared. They can then restrict publishing of sensitive information. In order to be able to control information, parents need to be given usernames and passwords. This, however, is often perceived to be too intrusive from the perspective of younger users.
In this scenario we discuss how User-Managed Access can be used to support parents with restricting information publishing by their children. We present how younger users of social networking applications can benefit from our proposal. With our scenario we show how the User-Managed Access approach allows a user to delegate access control related tasks to other entities that may have a better understanding of security requirements for resources owned by those users.

Desired Improvements

- Would it be possible for Alice not to be concerned with security and privacy issues and only take care of publishing here data on her favorite social networking application?
- Can Alice achieve that without revealing her username and password to the entity which should have access to her privacy and security settings?
  - The Social Networking Application may allow users to add accounts which could be used to control security and privacy of data published by those users (e.g. Alice may create an additional account with a different username and password which she can hand over to her father – this account could be used by her father to change security and privacy settings for Alice). This approach, however, is not efficient. If Alice uses more than one Social Networking Application then she might need to create multiple accounts and she will end up having security and privacy settings hosted in multiple places.
- Could Alice allow the Social Networking Application to delegate access control related tasks to a third party component? Can such component be under control of a different entity than Alice? Could such entity define arbitrary terms that must be met in order to access Alice’s Web resources hosted by her Social Networking Application?

Improved Scenario

Alice, a 14 year old girl, wants to have an account on a popular social networking application - FaceSpace. She wants to create a network of her friends with whom she wants to share photos and discuss her hobbies. She wants to keep in touch with them and does not want to be left behind with new technologies that have been used by her peers for some time now.

When Alice sets up the account at FaceSpace she needs to provide a variety of information. This includes providing information about her age. When Alice states that she’s 14 then FaceSpace detects that she is very young and informs her that she will need parental control over all the information that she submits and wishes to share with other users of the application. What this means is that an adult (legal guardian) will be able to restrict access control for information that Alice submits. However, Alice will still be able to share information if her access control rules do not contradict with those specified by her legal guardian. To set up the parental control, Alice asks her father Bob for some help.

Bob is happy that his daughter will be able to communicate with her friends but he is concerned with what information will be released and how this information might be used by legitimate or malicious users. He knows that FaceSpace has been certified to support parental control and allows third party UMA-based access control systems to be used for that purpose.

Bob is already using a specialized Authorization Manager – CopMonkey – for his own purposes. He uses this AM to define access control policies for his personal data (home address, telephone number) so that he can point other Web services to pull such data as required (e.g. during registration process at different Web sites). Moreover, he protects his online calendar service using CopMonkey and he established necessary relationships between his calendar feeds and his Visa payment service. He wants to be sure that Visa knows where Bob is and can correlate such data with data regarding his credit and debit card usage.

Bob decides that CopMonkey will be perfect for parental control over Alice’s information. Therefore, he introduces this AM to FaceSpace (establishes a trust relationship between these two services). As such, he is able to restrict how different information published by his daughter is accessed by her friends or other users of the application. Bob sets up basic policies regarding sharing of such information:

1. (1) Alice’s pictures and video clips can be shared with her friends only.
2. (2) If the request to either a picture or a video clip comes from a user who does not have a connection with Alice but is a member of the same group as Alice is then such user needs to agree not to further share this picture. Moreover, he needs to provide a certified claim that he is a member of a particular group.
3. (3) Only friends of Alice can see her personal information such as telephone number or email address.

When the account is set up and parental control has been configured then Alice is able to use FaceSpace just as any other user. She writes comments about her day, posts links to interesting movies. Additionally, she uploads some of her photos and short video clips. She knows that her father is very concerned with Alice’s safety and the privacy of her information and that her sharing options are therefore limited. She knows that she can share the information by herself but that she cannot override higher level sharing constraints imposed by her father.

She’s not sure how that works but leaves this issue to her father – after all he is more proficient in defining correct privacy and security settings than she is.

Alice uploads a photo from her birthday party and wishes to share with some of the users of FaceSpace. When the upload is finished, she clicks on the “Share” link next to the photo and defines the users with whom the picture should be shared. Her sharing options include all of her friends and members of one of the groups that Alice participates in - "Youth Sport". She confirms her sharing choice by clicking on the "Apply Sharing Options" link.

Jane, one of Alice’s best friends, heard that a new photo from a birthday party has been uploaded and wishes to see it immediately. She logs in to FaceSpace and clicks on the Alice’s photo album link. She can see a link to the newly uploaded photo. When she clicks the link, the application detects that a request has been made to an UMA protected resource. Therefore, such access request is subject to access control by the configured Authorization Manager apart from the internal mechanisms used at FaceSpace. Jane knows nothing about UMA and does not see the underlying protocol that is being executed by FaceSpace. She is not aware of the fact that FaceSpace acting on her behalf obtains an access token from Bob’s CopMonkey (meeting requirements of policy no. 1). She observes a very small delay before accessing the photo. This delay, however, is visible only when she wishes to access the photo for the first time.

Patrick does not have a connection with Alice but is a member of the "Youth sport" group. He saw that a new picture has been uploaded by one of the group members. Therefore, he clicks on a link to see the photo. Similarly to Jane’s case, FaceSpace detects that a request has been made to an UMA protected resource. Therefore, such access request is subject to access control by the configured Authorization Manager apart from the internal mechanisms used at FaceSpace. Patrick gets redirected to a page presenting a short description of what just happened (i.e. that he is subject to additional access control mechanisms as defined in policy no. 2). He sees that he must agree not to distribute the photo and that is a member of the "Youth sport" group. He agrees to that happily and is redirected back to the photo hosted by FaceSpace. Just as Jane, he is not aware of any complexities of the UMA protocol and only knows that he had to agree to some simple terms. Thanks to the UMA protocol, CopMonkey collects the following claims: (a) one self-asserted by Patrick that he will not share a photo, (b) a claim certified by FaceSpace that Patrick is a member of the "Youth sport" group. Both claims are required to authorize FaceSpace access Alice’s photo on behalf of Patrick.
Patrick likes the photo very much and he decides to see who Alice is. Therefore, he clicks on her "Profile" link. This access request is also subject to parental control configured for FaceSpace. In this case, Patrick is denied access which conforms to Bob’s policy no. 3.

Over time, Alice learns that allowing her father to have impact on security of the resources that she shares with her friends is not a bad thing. She feels safe and knows that everything she submits to FaceSpace is secure. Over time, Alice also learns more about security and sees what information is prevented from being shared with her friends. In the future she hopes to make better security decisions by herself. At some point she’ll be fully responsible for controlling access to her resources.

Her father Bob is also happy as he knows that his daughter can communicate with her friends in a safe and secure way. He checks his Authorization Manager on a daily basis and composes access control policies and defines terms as new photos and other resources are uploaded by his daughter. Moreover, he audits all access requests and sees how Alice’s friends access her photos and video clips. He hasn’t noticed any abuses and is confident in whatever her daughter does. After all, he’s fully responsible for her privacy and security and he puts much effort into ensuring that his daughter stays safe and still enjoys the benefits of social networking on the Web.

Discussion

Actors in the described scenario are as following:

- Primary Resource User – Alice
- Authorizing User – Bob
- Protected Resource #1 – Photo
- Protected Resource #2 - Personal Information
- Host - FaceSpace
- Requester – FaceSpace
- Requesting Party #1 – Jane
- Requesting Party #2 - Patrick
- Authorization Manager – CopMonkey

Distinctive aspects:

- The typical UMA Authorizing User is split into two roles: Primary Resource User and Authorizing User (Custodian)
- UMA serves the purpose of Mandatory Access Control (MAC) with regards to Discretionary Access Control (DAC). MAC policies are defined by Bob. They are unconditionally enforced and cannot be overridden by sharing options chosen by Alice.
- The same Web application acts as a Host and a Requester, externalizing part of its access control functionality to AM.
- Claims provided by the Requester must be certified by an authoritative body (e.g. Jane needs to provide a certified claim that she is a friend of Alice – such claim is signed by FaceSpace)
The following scenario shows how a user can delegate (a part of) access control functionality to a different user. In this case, an owner of a resource decides that a different entity (a custodian) will be also responsible for security of their resources. A user is concerned with producing and submitting content on the Web and may define sharing options for such content (e.g. Alice can upload a picture and define who sees that picture). However, it is the custodian that is responsible for ensuring that such content is well protected by defining mandatory access control policies. Such MAC rules cannot be overridden by rules defined by the primary resource user (e.g. Bob may define a rule that Alice's photos can be shared with her friends only which may restrict Alice's sharing options). It is then up to the custodian what access control rules are applied effectively to resources. An Authorization Manager in such setting can be viewed as an access control module externalized from a Web application that is simply under control of a different entity. Such Authorization Manager may serve the purpose of Mandatory Access Control.

A custodian can be fully responsible for defining access control policies and may be fully separated from an owner of resources. In the described scenario, Bob could be the only entity that defines access control rules for Alice's resources and Alice would only be concerned with producing and submitting these resources to FaceSpace. In such case no direct interactions are needed between the primary resource user and an authorizing user. A primary resource user may not have any knowledge about the security that is applied to a resource. As such, he or she can focus on the main tasks related to producing a resource (e.g. writing a document, submitting a photo) and can leave applying security to those who have greater knowledge and experience in defining security rules.

In the scenario, a particular Host could only accept to establish its trust relationship with a whitelisted AM. Such AM would be able to certify that a particular Authorizing User is an adult and a legal guardian of the Primary Resource User. Bob needs to get involved in the registration process to establish a trust-relationship between FaceSpace and CopMonkey. Another option for Bob, for example, would be to get an email with the instructions concerning introducing FaceSpace to AM and to perform this introduction later. Before this happens (i.e. resources submitted by Alice are protected by an adult), Alice can share resources by herself but FileSpace will not accept any liability for potential injuries that Alice may suffer from.

Scenario: Moving Resources Between Hosts (Pending)
In a rapidly developing Web environment there is a wide spectrum of various Web applications that combined together provide users with a powerful set of features. New interesting Web applications are being offered on a daily basis by corporations, small start-ups or single developers.

As new applications are being delivered, it is often the case that Web users wish to test functionality of those applications. Often they decide that a new application meets their needs and should substitute their old application. Apart from a better functionality, an application may additionally compete on the price which plays an important role when choosing which Web application to use.

In a situation where a user decides to switch between two or more Web applications then resources need to be transferred between those applications. Most commonly, the user will need to download all resources from an old application and upload them to a new one. In some cases, tools are provided to support users with this time consuming process. However, security settings and access control policies need to be manually set up on a new Web application and cannot be reused from an old one.

When a user wishes to transfer resources from one application to another, be it manually or automatically, then the problem of access control can be easily resolved if those applications support User-Managed Access. Once resources are transferred a user may simply plug in their existing and already configured Authorization Manager to a new application. Then a user may apply already composed access control policies for the same set of resources which is hosted by a new Web application.

In the next section we present how a User-Managed Access can be used to support users when resources are moved between Web applications (Hosts). We show how our approach allows a user to compose access control policies for their resources once and apply them independently of Web applications that host those resources.

*(NOTE: All references to real Web apps are hypothetical.)*

**Use Case: Moving Resources Between Web Applications (Pending)**

Submitted by: Maciej Machulak

Caroline is using the Web mostly to share short video clips and pictures of herself with her friends. She is using YouTube and Picasa Web Albums for that purpose. She is very happy with those services and is particularly pleased with their reliability and the fact that those services are free to use for non-commercial users.

Caroline is very security conscious. She wants all of her video clips and pictures to be well protected and only accessible by legitimate users (i.e. her friends). She is fine with spending some time every week to make sure that access control policies are in place to protect her resources. She is using an Authorization Manager for that purpose. All of her access control policies are composed centrally and are applied to her resources hosted by YouTube and Picasa.

Apart from pictures and video clips, Caroline stores her documents online as well. She uses Office Live Workspace as it is integrated with her word processor. She can access her documents over the Web from almost any place in the world. She configured Office Live Workspace to delegate access control to her Authorization Manager - the same which is used for Picasa and YouTube.

After some time, Caroline decides to use a single Web application to host her pictures and video clips. However, she does not want to change the Office Live Workspace application as she finds it very good. Caroline finds a PhotoBucket service. This Web application allows storing pictures and video clips and sharing them with other users of the Web.

Caroline reads reviews on the Web and decides to try the PhotoBucket service. She creates an account on the service. She then configures her Authorization Manager to allow PhotoBucket access her resources hosted on YouTube and Picasa. The PhotoBucket service then acts as a consumer of those resources and retrieves them. Caroline does not have to be actively involved in that process and she does not have to download any of her pictures and videos from YouTube and Picasa and upload them to PhotoBucket by herself.

The PhotoBucket service supports a User-Managed Access so Caroline configures it to use her already tested Authorization Manager. By default, all newly uploaded resources to PhotoBucket are made private. However, Caroline logs in to this service and clicks on the Administration option. She is then presented with a menu where she can perform various administration tasks. Among all of the options she can choose to apply security to all of her resources. Once she clicks on that option, the PhotoBucket contacts her Authorization Manager so that policies can be applied to resources hosted by this application.

Typically, when Caroline defines access control rules for her resources (i.e. pictures, video clips and documents), she logs in to the application that hosts those resources and clicks on an Access Control link next to a resource. She is then redirected to an Authorization Manager of her choice. Under the hood, the application contacts the Authorization Manager with information about the resource and operations supported by this resource. When Caroline is redirected to her Authorization Manager then she sees that it waits to compose an access control policy for this resource. She uses the interface of her AM to specify access control rules that will be later applied to this resource.

In the following case, however, Caroline does not click on the Access Control link. Instead, she clicks on a link to apply security for all of her resources. Similarly, she is redirected to her Authorization Manager and sees that a group of resources waits for policies to be specified. Her AM detects that access control policies were previously applied for those resources and informs Caroline about that. What she needs to do at this moment is to confirm that those policies can be reapplied. Once she does that, her resources remain protected in the same way when those resources were hosted by YouTube and Picasa Web Albums.

**Architecture**

The architecture for a User-Managed Access for the provided scenario is depicted below.
A user delegates access control for a set of resources from a Web application to an Authorization Manager. If a user decides to move this set of resources from one application to another then the same set of policies can be reapplied to those resources at the new application. Policies stored and evaluated by an Authorization Manager are therefore application agnostic. A user is able to define access control rules to resources independently of the application that hosts those resources.

View of the actors presented in this scenario with regards to the generic architecture of a User-Managed Access is depicted below. Presented diagram shows an Authorization Manager (1), a User (2), a set of Hosts (3), Requesters (4) and a Resource (5).

An access control policy protects resources independently of Web applications (Hosts). As such, if a resource is moved from one application to another application, the same access control policy can be easily reapplied.

Discussion

The following scenario shows how a user is able to reapply already composed access control policies to resources if those resources are transferred from one Web application to another Web application. Typically, in such situation it would be necessary to define access control policies from scratch or to transform policies from one application to the format used by another application. However, in case of a User-Managed Access where policies are stored in a central location, it is possible to simply apply those policies to the same set of resources that is hosted by a different Web application.

Reapplying an access control policy to a set of resources does not differ much from applying a single policy to a resource hosted by a Web application. Typically, when a user creates a resource on the Web and wants to protect it then a Web application contacts an Authorization Manager so that either a new access control policy can be composed or an already defined policy can be applied. In a situation where a resource is transferred from one application to another, the new application perceives such resource as newly created. Therefore, this new Web application contacts an Authorization Manager as usual. It is an Authorization Manager that detects that a policy for this resource has been previously defined and applied. Therefore, this Authorization Manager proposes to reapply the same policy to protect this particular resource.

An important issue that needs to be resolved in the discussed use case is the possibility of having different operations that are supported by different applications for the same type of resources. An example of that is when one application allows downloading, writing, deleting and shrinking a picture while another application allows downloading, writing, deleting and transforming pictures. When an application contacts an Authorization Manager it sends information about a resource along with operations supported on this resource. An Authorization Manager may detect that a policy has been already specified for such resource and may propose such policy to be reapplied. However, a set of operations described by a new Web application may differ from the set of operations that are defined in a security policy. This can be either a subset of original operations, a superset of those operations or a different set of operations.
If a new Web application supports only a subset of operations that were originally supported by the previous Web application, then rules for those operations that exist in an access control policy are simply removed. In case a new application supports a superset of operations then all rules from an access control policy are retained. New rules for newly supported operations can be easily added to the policy. In case the set of operations differs from the operations as defined in an access control policy, a human intervention may be required to map names of old operations to the names of new operations.

Dimensions

- **Scope**: This use case touches the notion of Scope is so far as the moved resource is to be assigned the same Scope values.
- **Cardinality**: This use case involves multiple Hosts and multiple AVFs, and as such it may have a high degree of cardinality.
- **Nature of access to protected resource**: This use case may require the nature of access to be determined. (In the diagram, an API method is indicated.)
- **Person-to-** **Self**: This use case may be implemented in a manner that involves a Person-to-** **Self transaction, in the sense that the User that authorizes the move is also the same User that authorizes the new Web Application to accept the existing AM currently used by the User.

---

**Scenario: Sharing Trustworthy Personal Data with Future Employers (Accepted)**

Submitted by: Maciej Machulak

Throughout the entire period of studies, students fill in a portion of a larger database with their own personal data. Such data is often sensitive and of very high value. This includes information about attended modules and courses, obtained marks and comments, awarded certificates, and acquired skills. This data may change very often (e.g. exams marks are added on a daily basis in the exam period; coursework marks can be released to students many times during a single semester) while other data may change infrequently (e.g. language certificates are usually awarded once every few years).

In the modern, highly competitive, professional world it is necessary for students to be able to present themselves to their potential future employers from the very best perspective possible. Students, acting as job applicants, may want to list all the possessed skills, list all the modules and present all the good marks which may positively influence the way they are perceived during their job application process. Some of the information that future employers may require is of very high value and may result in two similar candidates being different in the eyes of an assessor.

The following scenario presents motivating circumstances in which sharing information hosted within HE institutions with future employers makes sense. In this scenario we focus on sharing a limited set of resources available as Web resources with unique URIs. Each URI represents a single Web resource created or composed by a student to expose some of its Information, e.g.:

- List of modules: www.ncl.ac.uk/eportfolio/john.smith/modules
- CV: www.ncl.ac.uk/eportfolio/john.smith/CV
- Certificates: www.ncl.ac.uk/eportfolio/john.smith/certificates
- Marks: ness.cs.ncl.ac.uk/john.smith/marks
- Marks with comments: ness.cs.ncl.ac.uk/john.smith/marks/CSC2501

**Problem Scenario**

John is a full-time graduate student at Newcastle University and is currently doing his final year project in the School of Computing Science. He is preparing to write his final dissertation. However, he now focuses on doing well on the forthcoming exams he has to take. He has been attending four compulsory modules and he needs to pass them all with distinction to get a good overall mark. He also attends two additional modules in the Business School. He does not have to take exams from those modules but needs to submit reports at the end of semester.

John uses online systems provided by the University to manage his studies. He uses the NESS system (University’s in-house Virtual Learning Environment) which has information about all modules that John is currently enrolled to. It allows John to submit his work in a digital form while staff can comment on such work, mark it online and comment on it. Additionally, NESS holds all the information about marks obtained by John and comments as received by module leaders, demonstrators or examiners.

The other system that John uses is the ePortfolio system. He can record progress of his research project and also document how his research is helping him to develop new skills that can be of great value to potential employers. It is through the recognition of these high-level competencies that he will be able to present the value of his qualification to future employers. In ePortfolio, John has an online version of his CV that is updated regularly based on courses that he takes and skills that he gains through his education at the University. ePortfolio allows professors to complement on John’s research activities and to make comments on John’s work.

John is a very active student. Apart from his taught programme, John also leads a small research group at his university. He managed to interest his friends in this project and they've been working on graphical passwords for some time now. They use a shared file space (accessible over the Web as We bfolders) to share documents, presentations and data which they obtain from user surveys. They also use a shared code repository (SVN) for their innovative prototype of an authentication system. They have a private Wiki to discuss various research issues and a public one where they describe their project. The project team also has a blog where John and his friends discuss their progress.

John participates in various sport activities as well. He has joined the University’s sports centre during his first year of studies. He is a member of a 5-a-side soccer team and plays in a regional league. His team plays really well and is often able to perform well during various tournaments. His team has been second during the last regional championships but has been awarded a prestigious award – Fair Play Award – from the organizers. John has an account on the online system (CPRS - Centre for Physical Recreation & Sports) that manages the league and has insight into all the statistics of his team and himself. The system lists all his achievements and awards.

John knows he needs to prepare in order to start his professional career after graduation. He has performed well over those three years of his BSc studies and he is sure he will be able to find a perfect job. However, John he is a little worried about competition as many of his friends were good as well. He knows he needs to articulate all his strengths and list all of his accomplishments perfectly in order to be distinguishable during the first process of recruitment - the employee application pre-screening.
Therefore, John decides to spend an entire week on preparing all the documents that may support him during job application process. John starts with his CV by updating the list of modules he has already attended and by listing skills he has gained during the project. He also creates a list of all the additional courses he has attended and identifies skills he has gained. He wants both lists to be trustworthy by potential employers so he sends them to his school’s reception so that they can be certified. Because John attended modules from various schools it takes time before he can get both lists back. Moreover, he will be having some important exams from which he expects good marks and he is disappointed that it’s too early to include them in those lists.

John also updates the list of publications he and his research group has already published. He traverses through the Wiki and his team’s shared file space and compiles a package of some of the more interesting materials. This includes PowerPoint presentations and reports he and his friends have produced over the last couple of months. Moreover, John packages the software he has developed with his colleagues as he wants to send it for review by potential employers. He attaches the license that code should be only used for review as it is considered confidential and under ongoing development. He’s proud of the already developed functionality in this version – 0.9.8. However, he knows that the next release will be the first which is stable (1.0.0) and will have some major improvements which would make the software even more interesting. The code should be refactored as well which could probably influence how it is perceived by potential employers.

To further support his application, John asks for reference letters from two of his professors. He sends them emails and hopes to get letters as soon as possible. He knows one of the professors is away for the examination period but really counts on his letter. He has previously got very good comments on his coursework, exam and module in general. He wishes he could have shared this comment instead of having to wait for a letter.

Over nearly three years at the University, John has also successfully passed two language exams and has obtained language certificates. He knows that his marks from both exams are very good so he decides to include both certificates when applying for his dream job position. He logs into his ePortfolio system and goes to “Certificates” section. He sees the list of all his certificates and decides to download them. Unfortunately, both certificates get downloaded without required signatures so he prints them, asks his school’s reception for signatures and scans them back. It’s a pity he just cannot give access to interested parties to those certificates hosted by the ePortfolio system.

He knows that knowledge and university skills are not enough in order to get a good job position. He decides to show that he’s a healthy young man and that he finds time for sport. He logs in to his online account at the regional 5-a-side soccer league system. Then he copies all his achievements from the website and produces a document out of that. It’s a pity that he cannot give access to some of the information hosted by this system as this would be both more efficient and probably more trustworthy for the interested party. He notices that the current list does not include the recently obtained Fair Play award and his team being second during the tournament. Therefore, he adds that manually to the prepared document. He knows that information within this system is updated daily, including achievements and statistics, but he can’t wait till the last day before applying for a job to get the most recent information.

At the end of the week, John is ready to start searching for the job and has the following documents in his “toolbox”:

- CV
- Two signed lists: Core Modules + Additional Courses
- Two reference letters from his professors
- Two language certificates
- Package of software version 0.9.8
- Document describing sport achievements
- Research project materials (presentations, reports)

With all the required documents, John searches through many popular “Job Searching” Web sites and looks for his dream job position. He decides to pursue only two of such websites – the MyCareerBuilder.com and SearchJobs.ac.uk.

MyCareerBuilder.com allows students and professionals to create their personal accounts where they can create their online CVs, list their skills and search through a vast amount of job positions. Once a suitable job position is found, application is also done through the MyCareerBuilder.com website. An applicant fills in a uniform application form and attaches other sources that may support their application (e.g. lists of modules and courses, packaged software projects, scanned versions of certificates, scanned reference letters).

John creates an account on the MyCareerBuilder.com website and provides all the required information. He then uploads all the documents he has prepared. He searches through many job offers and ticks those which are of his interest. Using his “Account Settings” page he agrees to allow “head hunters” to search through his profile and send him invitations for job interviews. He specifies what kind of offers are of his interest (i.e. IT sector only, based in UK) and applies rules which offers should be automatically rejected (i.e. offered salary below the specified threshold). He knows it may take even few months before he gets any information regarding a positive pre-screening process but he’s sure he’ll get a good offer some day.

SearchJobs.ac.uk, on the other hand, does not offer creating a custom account and only provides advertisements along with links to organisations and companies that wish to hire new employees. If an applicant wishes to apply for a job position then he must click on the provided link and is usually redirected to a Website where he may fill in all the details as required by a future employer (e.g. personal information) and upload necessary documents (e.g. CV, reference letters, etc.).

John searches through various job offers and follows those which sound very interesting. He finds dozens of them so he spends lots of time filling in various forms and uploading documents on many different websites. After few of advertisements he already looses track of the offers he decided to respond to. He knows, however, that the more offers he replies to the better the chances are of getting employed.

Desired Improvements

Future employers may need ways to be able to trust the information that they are presented with by applicants. A list of all the attended courses and modules now needs to be signed by an authoritative body from a HE institution. When a student attends various courses and is enrolled to many different courses either within distinct HE institutions or governed by different administrative bodies, then separate lists are required to be certified. Moreover, in

Following is the list of improvements that could be done in order to support John with the job application task:

- John provides a link to his already created CV hosted by the ePortfolio system at his university
- John may only need to create a single CV and host it using a single system and does not have to worry about multiple (and possibly inconsistent) copies sent to support his job applications. The system may allow for multiple CV versions to be created in order to support applying for various job positions where different sets of skills are important (e.g. one CV to support his job application in System Administration sector and another CV to support his application in Software Development sector).
- The ePortfolio system may have the option to certify information inserted in the CV (e.g. the list of attended modules, completed courses, degree etc.) is valid and acknowledged by the University. Future employers may consider such information more trustworthy.
In many cases the job application process takes time (even months). It is possible that documents as submitted by John to his potential employers become easily outdated. In the provided scenario the list of modules and courses will change after the examination period with potentially important exams passed with excellent marks and comments. Software that has been submitted by John will have a major release soon so that John could allow potential employers to have access to their SVN account over the Web and be able to see introduced changes immediately. The document listing John’s sport achievements changes rapidly with every tournament his team participates in. Allowing John to give access to some information from his online sports account would simplify the process of notifying interested parties about such changes.

John needs to involve his professors in the job application process and ask them for reference letters. Such letters need to be signed. It is of the case that John cannot upload such letters but must rather provide email addresses of his professors so that potential employers can ask for references by themselves. This is because employers must be sure that opinion has been given by an authoritative person and has not been hand-made by the applicant. It might be better for University staff to host comments regarding students using existing e-learning systems and to allow students to publish some comments (and opinions) about themselves to interested parties (i.e. future employers). As such, the entire process of obtaining (and giving) reference letters would be significantly simplified.

The ePortfolio system that John uses may allow him to compose a list of already completed courses along with marks that John got during his exam period. It may be up to John which modules and courses should be exposed and which marks should be revealed. If he received good comments (e.g. for his practicals) then John may want to expose such comments as well. He does not have to prepare a list of his courses by himself or using an electronic system and have it signed but he can do that online. Again, such list can be certified by the University to be valid and may be considered more trustworthy by future employers than other lists created and submitted by candidates in a traditional form.

Could John give access to some sections of his research team internal Wiki and shared file space in order to share the important materials from the research project? In such case, he would make sure that interested parties always have access to the most recent copy. He would additionally keep track of who access what information regarding his project.

Solution Scenario

When John applies for a job he needs to make sure that all the information hosted by the systems that he uses is updated. This includes checking the NESS and ePortfolio system of Newcastle University and the sports system that he uses. Moreover, John would check where all the important information resides regarding his research project (be it the internal Wiki, blog or the shared file space). He would not have to spend too much time on that as all systems are meant to provide the most recent data about John (e.g. the list of modules or the list of obtained, etc.).

John would then list all the necessary links containing information that he wishes to use during his job application process. This would include links to information that he would previously hand over to potential employers in form of documents:

- **CV:**
  - www.ncl.ac.uk/eportfolio/john.smith/CV
  - Two signed lists: Core Modules + Additional Courses
    - www.ncl.ac.uk/eportfolio/john.smith/modules
    - www.ncl.ac.uk/eportfolio/john.smith/courses
  - Two reference letters from his professors
    - www.ncl.ac.uk/eportfolio/john.smith/references
  - Two language certificates
    - www.ncl.ac.uk/eportfolio/john.smith/certificates
  - Packaged software – latest version
    - svn.ncl.ac.uk/passresearch/trunk
  - Document describing sport achievements
  - www.ncl.ac.uk/eportfolio/john.smith/public_profile
  - Research project materials (presentations, reports)
    - webfolders.ncl.ac.uk/passresearch/public_materials

With the above mentioned list of links, John goes to two different “Job Search” websites: MyCareerBuilder.com and SearchJobs.ac.uk.

MyCareerBuilder.com allows students and professionals to create their personal accounts where they can create their online CVs, list their skills and search through a vast amount of job positions. Once a suitable job position is found, application is also done through the MyCareerBuilder.com website. An applicant fills in a uniform application form and provides links to other sources that may support the application process (e.g. links to lists of modules, links to software, links to certificates, links to reference letters).

John creates an account on the MyCareerBuilder.com website and provides all the required information. He then provides links to information he think might support his application (previously mentioned list of links). He then needs to authorize the MyCareerBuilder.com website to be able to access those resources. As such, the MyCareerBuilder.com website establishes a trust-relationship with Web services used by John (i.e. ePortfolio, NESS, SVN, CPRS system and Webfolders). John uses his university’s Authorization Manager for that purpose. He creates a relationship between the MyCareerBuilder.com and those systems. He defines terms that MyCareerBuilder.com must accept in order to access this information – i.e. John wants all the supplied resources to be used solely for the job application process. He also defines that MyCareerBuilder.com shall only present this information to interested parties who meet John’s requirements – i.e. only companies from the IT sector only and located within the UK which offer jobs with a salary above a certain threshold may access information about John.

John also goes to SearchJobs.ac.uk which does not offer creating a custom account and only provides advertisements along with links to organisations and companies that wish to hire new employees. He searches through all the relevant job positions and follows links to those which sound very promising. Even though he finds dozens of them, all systems provided by his future employers allow providing URLs only instead of actual resources. Therefore, filling in application forms for each employer can be done quickly with minimum effort. John only provides links to feeds of his data – CV, reference letters, etc. He establishes a relationship between each of employer’s HR system and his Web services that host John’s resources. He does that using his preferred Authorization Manager which he can use to manage those relationships easily.

John now does not have to worry if the application process takes days, weeks or months and if the potential employer will not be able to see John’s latest accomplishments. John gave access to feeds of his data which are the most recent one and are updated as necessary. As such, information that can support John’s application process resides in a single place and can be easily managed. John can audit how his information is being accessed by interesting parties.
Scenario: Advertising a Resource to Arbitrary Potential Requesters (Pending)

(This is colloquially known as the "Hey, Sailor" scenario.)

While a resource may be sensitive and in need of protection against unauthorized access, the URL for the resource need not always be sensitive. An interesting way to provision a resource's URL to requesters is to advertise it directly – for example, on a blog or in a tweet. This may make sense in cases where the authorizing user has not yet met all the parties who may be interested in requesting access, or if the population of potential requesters is very large. When a person publishes URLs in this way, the act of publication functions as a "discovery service" for those resources.

Following are three specific circumstances in which advertising a resource widely may make sense. In all cases, terms negotiation becomes a key "line of defense" for authorizing access; see the scenarios specific to terms negotiation later in this document for more detail.

Sharing Exclusively with Family and Friends

Authorizing user Roxy wants to share her blog posts and song recordings only with family and friends. She publishes her posts and songs in feeds that are accessible only to those who can prove they are her family and friends, as she knows them through interactions on Twitter and Facebook and other social sites. (See the Requester Identification terms negotiation scenario to see details on how this access control scenario might work.) She also requires requesters to agree not to share this material further.

The feed URLs are available in the clear on her website, so that she can tell her loved ones simply to visit allaboutroxy.com to find the relevant links. But when they try to click on the links, they are routed to an interaction that requires them to prove their identity to her and to agree to her sharing terms.

In this case, each of her loved ones is a requesting party, working through some requester intermediary (such as Google Reader or Songbird) that helps them access her protected resources, which hosted either on her blog or elsewhere.

Sharing Resources on a Commercial Basis

Authorizing user Selma is an accomplished photographer, and she offers her photographs at full resolution for sale to any requester who can produce a receipt for US$10. The photographs are available at low resolution on the site, with the URLs of the full-size photographs advertised. When a potential photo purchaser tries to "hit" a full-size photo, they are told of the requirement for successfully getting the photo and given the opportunity to comply.

Sharing Personal Requests for Proposals (PRFPs)

When authorizing user Dirk's coffee maker dies, he constructs an online personal RFP (a description of his parameters and constraints for an acceptable replacement coffee maker, including height, capacity, basket style, price, and so on), hosted at a site that specializes in PRFPs that he has already registered with his AM.

One of the host site's services is to get his PRFP seen by various online retailers, and he might also tweet the PRFP. The online retailers will need to register with his AM as requesters in order to both see the PRFP and to supply a coffee maker proposal in a special inbox also hosted by the same site. Each proposal could contain an "offer URL" that allows Dirk to take advantage of each deal directly on their site, or could give other instructions on how he can get the deal.

Preventing Denial-of-Service Attacks on a Resource

Authorizing user Reed has a popular documentary blog where he exposes corruption and budgetary malfeasance in his home state, which he is happy for anyone to visit. However, the subject of his writing has made him collect some high-powered enemies. Some of them want to try to shut him down by flooding his blog and his comment threads with Denial-of-Service attacks. So Reed configures his AM to require that all requesters "prove their sincerity", in effect, by performing proof of work.

Scenario: Controlling Access to Health Data (Pending)

Submitted by: Gerald Beuchelt

Project hData defines the Representational State Transfer (RESTful) exchange of health-related hData Records and Section Documents. While systems with a single health organization may exchange health data without strong security controls in some cases, any exchange of health data across public data networks or between different actors will require strong information assurance. This scenario outlines the basic requirements and a high-level conceptual architecture for patient access-controlled hData network exchanges with a specific focus on cross-organizational interactions.

For other deployments of hData, a different set of information assurance and security requirements might apply: for example, if two separate hData enabled record systems are used within an organization – one as the authoritative medical record store for patient data, and another as the financial accounting system – there will be fewer requirements on security constraints, since the two systems are likely within the same trust domain.

Additional details about Project hData are available in its document repository.

Distinctive aspects:

- High-value, personal, sensitive data that requires appropriate security, access controls, and privacy controls
- A Discovery and Authorization Service component dictated by the hData architecture that requires its own access-control protection ("trusted discovery"), in addition to providing protection for actual health data
- Services that are frequently in the position of being both hosts and requesters
hData Format and Data Exchange

The hData format consists of a collection of individual documents (Section Documents), organized in Sections. Sections may contain Section Documents (i.e., individual data points) or other sections.

All Sections are referenced in a manifest called the Root Document. By default, Sections contain Section Documents of a specific type (e.g., medications, x-ray images, etc), but when explicitly tagged in the meta-data portion of the specific Section Document, Sections may contain Section Documents that are different from the default Section type.

hData records may be accessed through a RESTful Application Programming Interface (API), with the abstract Section structure providing a canonical mapping to a Uniform Resource Locator (URL) pattern.

Use Case: Protecting Health Data and Metadata (Pending)

Actors:

- Patient as Authorizing User
- hData DAS component in dual roles as an Authorization Manager and as a Host of health resource metadata
- Various parties such as primary care physicians, emergency rooms, testing laboratories, and personal health datastores in potentially dual roles as hosts of health data, requesters of health data from other hosts, and requesters of metadata services from the metadata Host

From a single patient's perspective, an hData deployment that crosses trust domains acts like a "circle of access" for the online services that handle the patient's health data in any fashion. (See this hData presentation for a step-by-step accounting of how a patient's visits to a primary care physician, and subsequently an emergency room doctor, would occasion a need for data-sharing by those two parties on the patient's behalf, and therefore a need for them to obtain authorization for the sharing and discovery of the resources in question.)

In the following diagram, the following services handle the patient's health data:

- The primary care physician, as both a Host of data generated from patient visits and tests performed in the PCP's office, and a Requester of data from emergency room visits
- The emergency room, as both a Host of data generated from patient visits and tests performed at the ER, and a Requester of data from the PCP
- A personal health datastore service chosen by the patient, as a Requester of data from the patient's medical visits (it could as easily have been a Host itself, if, say, the person chose to store the ongoing results of self-administered weight or blood pressure tests there and wanted to make that data available to a third-party health assessment website)
The "DAS" in the center is the Discovery and Authorization Service, an hData conceptual function. The XRD standard has been discussed as the likely metadata format to be used in the discovery component of the DAS.

This diagram describes how hData can use UMA to give the patient control over both health data and the mechanisms of introducing a new provider or other health data source/destination into the picture.

Assumptions:

- Health care provider systems are able to produce signed attestations of their identity and/or medical credentials when asked by the DAS to do so.
- The patient is able to craft authorization policies that constrain access according to identity and medical credentials.
- The discovery service component of the DAS is able to deliver up custom metadata (or equivalent resource lookup services) to each seeker of health data, based on the patient's policies.

Preconditions:

- The patient establishes an account at the DAS.
• The patient, when visiting a health care provider such as the PCP, directs the PCP's electronic health record (EHR) system to the patient's DAS in some fashion.

Steps as shown in the diagram:

1. The PCP's EHR system registers with the DAS discovery service's Host component by interacting with it as a Requester, attempting to add the PCP's own metadata to it to join the patient's circle of access.
2. The DAS authorization service's AM component imposes user-managed policy over the adding of PCP metadata, based on the PCP's credentials, ultimately allowing the PCP to add its metadata to the DAS. (In future, the PCP can update its metadata in the same fashion as necessary.)
3. The PCP then interacts with the DAS again, this time as a Requester retrieving the metadata resources that will help it discover where relevant health data is hosted for this patient. It can then approach the appropriate Hosts of patient health data in the usual UMA-protected fashion.
4. ... At some later time, the personal health datastore, having already performed its equivalent of steps 1 through 3, approaches the PCP system to read the latest versions of the patient's health data stored there. Once the personal health datastore successfully retrieves health data from all the systems it is authorized to access, the patient can get an aggregated view of this data in one place.

Terms Negotiation Scenarios

Terms negotiation forms a special sub-category of UMA scenarios. Requirement R0b says that UMA must "Allow an individual to select policies and enforceable contract terms that govern access, as well as data storage, further usage, and further sharing on the part of requesting services." As discussed in the issues section How Terms Can Be Met, while policies can be applied unilaterally, terms represent some set of requirements a requester must meet, where success means that an implicit or explicit agreement has been forged. Thus, the expressing and handling of terms have an impact on the UMA protocol in a way policies do not.

This section describes scenarios and specific use cases for terms negotiation.

Scenario: Imposing No Terms (Pending)

Submitted by: Eve Maler

It is reasonable, under an UMA resource protection scheme, for some resources not to need terms-based protection at all. For example, the Authorizing User might allow any Requester to gain access to a particular resource at any time, or to gain access that is throttled exclusively in a (unilateral) policy-controlled manner such as "This resource can be accessed only during certain hours".

This could be thought of as an "audit-only" mode, for which relationship manager applications might provide analytics akin to those provided today for open-access resources such as blogs.

The advantage of using UMA over existing analytics applications that operate on web-server logs is that the audit-log information from multiple Hosts can be more easily combined and analyzed as a whole, and that the policies for such access can be managed centrally. Obviously there is no "terms negotiation benefit" as such from this scenario.

Scenario: Require Requester Identification (Pending)

Submitted by: Eve Maler

(Note that the WG is currently discussing the proper way to understand and label all the parties on the requesting side; this scenario may be revised in the future to accord with later decisions about these concepts. The most recent proposed terminology is used here for now.)

If the requesting entity can identify itself (and any requesting user standing behind it) to the authorizing user's satisfaction, the user can set policies and make decisions that result in appropriate requesters gaining access.

Types of Requester Identity

There are several special cases of requester identity that are outlined in sections below. These may deserve optimization or special treatment in the UMA solution.

Requesting user(s) = authorizing user

The requesting user is the same human being ("natural person") as the authorizing user. For example, the calendar-sharing scenario and personal loan scenario involve a user who arranges to share her information with a variety of other applications that she herself logs in to use, and where the sharing is ultimately for her own benefit. Similarly, the Kantara InfoSharing WG's car-buying scenario suggests that car-buyer Sally can authorize her car's manufacturer to access personal data required for her membership in its frequent-road-trip club. Though she may have different usernames at the requesting application, the host, and the authorization manager, the connection being forged is, in a sense, "with herself", just as is true of OAuth connections today.

Issues:

• Are there special optimizations – and/or concerns (for example, about privacy) – in the situation where an Authorizing User is granting access "to themselves" in the guise of other applications and digital identities? Policies that specially mention all the digital identities under which the same person travels can be considered privacy-sensitive information since they expose a kind of "federation" of those identities.

No requesting user
The requesting entity is a company or other organization – a "legal person" – that is acting on its own behalf in seeking access. For example, the protected inbox scenario may involve some requesting entities (vendors) that want to send marketing messages to the authorizing user. In this case, there is no requesting user.

Other requesting user(s)

The requesting user(s) are natural persons who are distinct from the authorizing user. For example, the Kantara InfoSharing WG's car-buying scenario suggests that car buyer Sally might want to let her husband and a friend (individual people with online identities) see her collected research on new-car options.

Issues:

- Should this case be prioritized lower? The telecon of 2009-08-27 discussed deferring this set of use cases.
- It is not UMA's responsibility to solve the problems that a People Service or a Portable Contacts API. However, could it neatly integrate with such solutions in order to allow relationship manager applications implementing an AM endpoint to provide more sophisticated ACL management?

Interaction of requester identity and authorizing user actions

If an AM can find out the unique identity of a Requester/requesting party, it can make use of it in two main ways:

- Compare the identity to some policy with which it has been pre-configured in order to make a decision
- Convey the identity to the Authorizing User in a request for real-time consent to access (based on prior user instructions to do this) – for example, in an email or SMS message

This table summarizes specific motivations for use cases exploiting both of these choices, where the Requester's identity is either self-asserted or has been attested to by a third party.

<table>
<thead>
<tr>
<th>Strength of identification</th>
<th>Pre-configured policy</th>
<th>Real-time consent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-asserted label</td>
<td>&quot;Anyone can gain access if they introduce themselves&quot;</td>
<td>&quot;Someone purporting to be 'Random' wants access&quot;</td>
</tr>
<tr>
<td>Identity from known issuer</td>
<td>&quot;Let (this identity, this list of identities) from (this issuer, one of these issuers) gain access&quot;</td>
<td>&quot;Requester 'Solid' (verified by 'Known') wants access&quot;</td>
</tr>
</tbody>
</table>

Pre-Configuration of Policy with Self-Asserted Label

"Anyone can gain access if they introduce themselves."

Examples of resources that might be protected this way; these are policies that could be set up in an AM at any time:

- "Let anyone offering an identifier access my RSS feed 'Blog'"
- "Let anyone offering an identifier access my calendar 'Work Free/Busy'"

This use case is likely not to involve any sort of sophisticated matching of pre-configured policy to a particular identifier that any Requester can just make up. Rather, it is likely to involve a policy that freely gives access to relatively non-sensitive resources as long as the audit log entries can use some sort of Requester-chosen label. This is marginally more interesting than merely recording IP addresses, assuming the Requester chooses to use a label that is meaningful on some level.

Pre-Configuration of Policy with Identity from Known Issuer

"Let (this identity, this list of identities) from (this issuer, one of these issuers) gain access."

Examples of resources that might be protected this way; these are policies that could be set up in an AM at any time:

- "Let Google's 'Carlos', 'Dahlia', and 'Evan' and Twitter's 'Frank' access my photo album, 'Soccer Practice'" (calendars, photos, and other resources are already shared today in a selective fashion by means of specifying email addresses or known usernames of intended recipients)
- Where the Authorizing User is Alice Adams: "Let Amazon's 'AliceAdams02134' access my e-commerce personal datastore" (this is a special case)

In the case of Sally the car buyer, who is granting research access to her husband and friend, she could specify that Requesters acting on behalf of her husband's Google identifier or her friend's Twitter handle always get access to certain protected resources. If she can be sure that Google or Twitter has vouched for the requesting user on whose behalf the Requester is making its access request, this is a greater level of assurance that warrants her setting policies around specific identities even before people wielding those identities attempt access to the resource in question.

Likewise, it could be powerful to set up a policy ahead of time that says that Amazon.com, acting on behalf of a specific identity that is known to represent oneself, can get access to one's shipping address or vendor-neutral wishlist. Since the Authorizing User already knows all the identities he himself wields in various applications, setting up policies to grant a set of social networking applications (acting on these identities' behalf) access to one's social graph or geolocation information at other applications already in the "known circle" (as in the distributed services scenario).

The Authorizing User could also, assuming identities of friends and family at sites such as Google and Twitter are known, create "ACLs" (access control lists) that enumerate the allowed parties per resource or host. (Note that design principle DP9 protects Authorizing User privacy at the expense of parties standing behind the Requester; some authorization policy depends on knowing the identity of those who approach the resource looking for access.)

Real-Time Consent with Self-Asserted Label
"Someone purporting to be 'Random' wants access."

Examples of resources that might be protected this way; these are real-time messages conveyed to the Authorizing User for a "yes" or "no" answer:

- "Someone purporting to be 'BelleCare Dental' is requesting access to your calendar, 'All Free/Busy'."
- "Someone purporting to be 'Eve Maler' is requesting access to your photo album, 'IIW 2009B'."

There are two circumstances for arriving at this combination:

- The Authorizing User has recently provisioned a particular party with the URL for a resource that is UMA-protected (or a way to discover the URL), and is thus expecting that party's Requester app to come along shortly and attempt access.

  This is somewhat similar to how IM handles and email addresses are shared and heuristically authenticated today: Alice and Bob exchange, say, Skype handles in a face-to-face conversation, and sometime thereafter "Someone purporting to be 'Bob'" approaches Alice in Skype asking to be approved. (The difference is that Skype really does authenticate some user against a "Bob" Skype handle, whereas here the label is entirely self-asserted, perhaps having been typed into a web form field by the requesting user (or the requesting entity's representative) when resource access was first attempted.

- The Authorizing User has freely published the URL for a resource that is UMA-protected, and the Requester approaches without prior notice (known as the Hey, Sailor pattern).

  This use case involves a user who is satisfied with self-assertion of Requester identity for this resource, so presumably the resource is not terribly sensitive or high-value.

Real-Time Consent with Identity from Known Issuer

"Requester 'Solid' (verified by Issuer 'Known') wants access."

Examples of resources that might be protected this way; these are real-time messages conveyed to the Authorizing User for a "yes" or "no" answer:

- "Google's 'CPABobBaskin345' is requesting access to your spreadsheet, 'CF2010'."
- OpenID "=JeffH" is requesting access to your photo album, 'IIW2009B'.

This use case provides stronger protection than the self-asserted version for gathering real-time consent in the Hey, Sailor pattern.

Issues

Following are discussions of technical issues raised by one or more scenarios and use cases. Acceptance of a scenario or use case will imply agreeing to develop a satisfactory solution to applicable issues.

(See also the Protocol Issues docket, which records specific technical issues waiting to be resolved in the UMA 1.0 Core Protocol spec or some other related spec.)

Issue: Policies Specific to the Web Resource Type

(Note that a partial resolution to this issue is captured in requirement R4.)

There is a potential need to restrict, anonymize, blur, or otherwise transform a shared resource, possibly based on the unique characteristics of its content type.

With respect to calendar resources, the premier calendar format standard already accounts for a blurring of data details by providing a "free/busy" option in addition to a full-data option. It feels like it should be out of scope to solve for filtering the calendar data cleverly (beyond the format's natural capabilities) to hide Alice's destination, hotel, etc. (though generic solutions such as making events taggable, and then filtering on the tags in a relationship manager interface, come to mind). An "identity oracle" approach (filtering the data into a completely different type) might be necessary if what Alice is trying to convey is simply "don't deliver my newspaper on these days" vs. "here's all of my travel information".

In the Controlling Two-Way Sharing of Location Information scenario, note that FireEagle allows a user to choose to share locations only at the city level, and this level happens to be chosen for the connection that authorizes Dopplr to read the FireEagle location (a different level can be chosen for each application that reads location from FireEagle). As it happens, Dopplr does not offer the same policy capability. Without having to teach UMA generically about all the possible policy options specific to all the kinds of information in the world, is it possible for each Host to teach each AM about the policy options it offers, in some way that lets the the relationship manager application surrounding the AM present user interface options to see and select these policies? Seeing may have less protocol impact than selecting, and seems to be a minimum value-add if the goal is to allow OAuth users to get a global view.

Some data-usage policies and terms may possibly have an interaction with some resource types, such as requiring recipients to discard volatile data after a period dictated by the data's type.

It has been observed that if fine-grained calendar filtering were a solved problem, different calendar sites could be shared with different friends as a way of managing minimal disclosure through indirect.
Issue: Authorization Manager Endpoint Discovery

The mockups linked in the calendar scenario imagine that the user's authorization manager endpoint (what we imagine Alice will perceive as the name of her relationship management service) will be handled as if it were an OpenID, with introductions to popular relationship manager services offered in an array by potential UMA Hosts much in the way that the RPX solution presents options. (The user always has the ability to self-host an authorization manager endpoint, similarly to self-hosting an OpenID provider – and they might even be colocated.)

Issue: Handling the Resource URL and Provisioning It to the Consumer Site

The mockups linked in the calendar scenario imagine the simplest possible situation: The Consumer site literally asks for exactly the kind of information it needs, and the user copies and pastes a URL into a field.

This is how calendar feeds, photo streams, RSS feeds, and other such resources are shared today; it works but we need to consider its scalability to arbitrary types of information. There are several challenges here: The Consumer's ability to handle the information, its way of expressing the desire/need for the correct information, and the user's (or user agent's) ability to provide it in a convenient and correct fashion.

In addition, the relationship manager interface is shown having some knowledge of that resource as a unique object. We need to consider how to let the AM and SP communicate about this information appropriately.

In the case of the photo set scenario, note that in OAuth usage today, the resource-based interaction is often accomplished silently from the user's perspective: the desired combinatorial effect simply "happens" as if the feature that was "outsourced" to a third-party app were native. Perhaps this is possible in the UMA approach.

Issue: How Terms can be Met

An AM has two major tools at its disposal in allowing access to a user's resources: policies declared by the authorizing user, and terms which the Requester must meet in order to gain access. To a first approximation, policies can be unilaterally applied, whereas terms require two parties to come to agreement.

Because policies are anticipated to be applied by an AM "silently" (out of band) with respect to the UMA protocol, this is an opportunity for AM business value and we should not dictate any answers here. But following are some policies that could be useful:

- How long to allow access: once, some number of times, for some period, indefinitely until the user says to stop, etc.
- Whether to let the user exercise a "right of refusal" by some interactive means (such as SMS) when a Requester approaches a particular resource: every time for that Requester, only the first time for that Requester, every time for every Requester, etc.

By contrast, terms might take some of the following forms:

- Make the Requester promise not to sell or otherwise commercially use the data thus acquired (in Creative Commons-like fashion)
- Require the Requester to pay the user ten dollars

The following hypothetical wireframe (with hypothetical Creative Commons-like sets of standard terms) imagines what a user interface could look like for an AM's default policy and term settings for all resources it manages:
The UMA group is hoping to borrow from the work of others in using any standard sets of terms that might exist, for example as might be developed by the Kantara Information Sharing (UD-VPI) WG. However, even if this area is well fleshed out, major design questions remain.

Human interaction by a party “behind” the Requester

Some parties behind a Requester’s actions may be big companies like credit card issuers, large e-commerce sites, or government agencies – but some may be small organizations, such as a dentist’s office. Small organizations may need a human-accessible interface and the option of an “I Agree” button so that the person manually fielding an offer of data can complete the transaction.

Requester resistance to user-driven terms

It may be necessary for us to consider “partial measures” in the V1 UMA effort to improve adoption. For example, it may be more difficult to demand evidence of positive action (such as payment) from a Requester vs. demanding a simple statement of passive acceptance of terms (such as “I agree not to sell the data”). This would be a natural first step if Requesters are at all amenable to the notion of user-driven terms.

If we discover that Requesters are resistant, we may need to consider options for allowing the user to passively inform the Requester of policies such as “I ask you not to sell this data”, rather than requiring action on the part of the Requester to accept such terms. Or given that Requesters are today in the habit of making their own terms of service and privacy policies known to users in passive fashion, we may need to account for a case where the user’s terms amount to an opening gambit of “What can you offer me?” in a contract negotiation.

Depth of contract negotiation

There is some minimum functionality needed around a sequence roughly like the following:

1. AM presents terms based on user configuration of same, followed by...
2. Requester demonstrates that it meets the terms presented

However, there are many layers of sophistication we could get into, depending on where our scenarios take us. For example, is it important for the user to be able to specify “you must satisfy these terms ‘or better’”? If so, what does “better” mean? Do we have to solve for “I will sell you \( n \) pieces of data for terms \( X \), but \( n+m \) pieces for terms \( Y \)?

Legal enforceability and terms persistence
We have discussed whether machine readability of terms is strictly needed, since having a URL that persistently refers to a human/lawyer-readable version seems to suffice in a lot of cases today for string-matched satisfaction (no complex negotiation), including very complex enterprise cases. Nat Sakimura’s blog post on contract exchange suggests various ways to characterize, share, negotiate, and record data-sharing contracts. How we answer these questions also has an impact on our goals around simplicity, particularly our emerging goal around not adding undue cryptography burdens.

Paul Bryan has stated a preference expressing a set of terms as a Web resource whose representation can be retrieved with an HTTP GET and modified (with an affirmation that the terms are being met) with an HTTP POST.

**Issue: Protected Resource Query**

How and whether to request the “protected status” of a resource: To answer this question, a host would pretty much have to go through the same dance as for a request for actual access. It might be protected against requester A but not requester B, or protected with a real-time user consent loop, etc. We’ll wait to see what real scenarios arise that need to be solved, and perhaps it will turn out that they can tolerate imprecision/latency. Before a host gives out the 401, it already knows intrinsically whether the resource is protected by some AM, and it has the ability to tell the requester this (whether this is a good idea or not we don’t know yet). It just doesn’t yet know whether the requester is going to be authorized to access it.

Since we’re now out of the authentication business, keep in mind that whatever the host does in response to the initial approach of the requester is up to it. It can assign a pseudonymous form of identifier (possibly literally using a cookie-based method), or really anything. If the host needs to protect the privacy of the requester according to its own policies or applicable laws or whatever, it’s up to the host to choose an ID wisely. We don’t really care about protecting the requester’s privacy, however; the whole point of letting the authorizing user control access is to let them do so on whatever criteria, and as part of that process the requester is simply going to have to authenticate, even if weakly.

**Change History**

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Version (v. 61)</td>
<td>Jul 11, 2017 09:06</td>
<td>Eve Maler</td>
</tr>
<tr>
<td>v. 60</td>
<td>Oct 05, 2010 22:08</td>
<td>Eve Maler: Migration of unmigrated content due to installation of a new plugin</td>
</tr>
<tr>
<td>v. 59</td>
<td>Oct 05, 2010 22:08</td>
<td>Eve Maler: Migration of unmigrated content due to installation of a new plugin</td>
</tr>
<tr>
<td>v. 58</td>
<td>Oct 05, 2010 22:08</td>
<td>Eve Maler: Migration of unmigrated content due to installation of a new plugin</td>
</tr>
<tr>
<td>v. 57</td>
<td>Oct 05, 2010 22:08</td>
<td>Eve Maler: Migrated to Confluence 4.0</td>
</tr>
<tr>
<td>v. 56</td>
<td>Oct 05, 2010 22:08</td>
<td>Eve Maler: Filled in some links to specific scenarios in examples of dimensions.</td>
</tr>
<tr>
<td>v. 55</td>
<td>Sep 30, 2010 12:44</td>
<td>Thomas Hardjono</td>
</tr>
<tr>
<td>v. 54</td>
<td>Sep 30, 2010 12:41</td>
<td>Thomas Hardjono</td>
</tr>
<tr>
<td>v. 53</td>
<td>Sep 30, 2010 12:34</td>
<td>Thomas Hardjono</td>
</tr>
<tr>
<td>v. 52</td>
<td>Sep 30, 2010 12:28</td>
<td>Thomas Hardjono</td>
</tr>
<tr>
<td>v. 51</td>
<td>Sep 30, 2010 12:25</td>
<td>Thomas Hardjono</td>
</tr>
<tr>
<td>v. 50</td>
<td>Sep 30, 2010 12:14</td>
<td>Thomas Hardjono</td>
</tr>
<tr>
<td>v. 49</td>
<td>Sep 22, 2010 10:56</td>
<td>Thomas Hardjono</td>
</tr>
<tr>
<td>v. 48</td>
<td>Sep 20, 2010 16:43</td>
<td>Thomas Hardjono</td>
</tr>
<tr>
<td>v. 47</td>
<td>Sep 20, 2010 16:42</td>
<td>Thomas Hardjono</td>
</tr>
<tr>
<td>v. 46</td>
<td>Sep 20, 2010 16:40</td>
<td>Thomas Hardjono</td>
</tr>
<tr>
<td>v. 45</td>
<td>Sep 20, 2010 16:19</td>
<td>Thomas Hardjono</td>
</tr>
<tr>
<td>v. 44</td>
<td>Sep 20, 2010 16:11</td>
<td>Thomas Hardjono</td>
</tr>
<tr>
<td>Version</td>
<td>Date</td>
<td>Time</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>----------</td>
</tr>
<tr>
<td>v. 43</td>
<td>Feb 16, 2010</td>
<td>10:35</td>
</tr>
<tr>
<td>v. 42</td>
<td>Feb 16, 2010</td>
<td>10:29</td>
</tr>
<tr>
<td>v. 41</td>
<td>Jan 28, 2010</td>
<td>19:35</td>
</tr>
<tr>
<td>v. 40</td>
<td>Jan 25, 2010</td>
<td>00:20</td>
</tr>
<tr>
<td>v. 39</td>
<td>Jan 24, 2010</td>
<td>10:03</td>
</tr>
<tr>
<td>v. 38</td>
<td>Jan 13, 2010</td>
<td>12:44</td>
</tr>
<tr>
<td>v. 37</td>
<td>Jan 09, 2010</td>
<td>18:16</td>
</tr>
<tr>
<td>v. 36</td>
<td>Jan 09, 2010</td>
<td>18:07</td>
</tr>
<tr>
<td>v. 35</td>
<td>Jan 09, 2010</td>
<td>18:05</td>
</tr>
<tr>
<td>v. 34</td>
<td>Dec 16, 2009</td>
<td>17:48</td>
</tr>
<tr>
<td>v. 33</td>
<td>Dec 16, 2009</td>
<td>17:43</td>
</tr>
<tr>
<td>v. 32</td>
<td>Dec 16, 2009</td>
<td>17:41</td>
</tr>
<tr>
<td>v. 31</td>
<td>Dec 04, 2009</td>
<td>17:32</td>
</tr>
<tr>
<td>v. 30</td>
<td>Dec 04, 2009</td>
<td>15:38</td>
</tr>
<tr>
<td>v. 29</td>
<td>Dec 04, 2009</td>
<td>15:33</td>
</tr>
<tr>
<td>v. 28</td>
<td>Dec 03, 2009</td>
<td>11:18</td>
</tr>
<tr>
<td>v. 27</td>
<td>Nov 21, 2009</td>
<td>10:17</td>
</tr>
<tr>
<td>v. 26</td>
<td>Oct 08, 2009</td>
<td>19:11</td>
</tr>
<tr>
<td>v. 25</td>
<td>Oct 04, 2009</td>
<td>20:16</td>
</tr>
<tr>
<td>v. 24</td>
<td>Sep 18, 2009</td>
<td>06:34</td>
</tr>
<tr>
<td>v. 23</td>
<td>Sep 18, 2009</td>
<td>05:47</td>
</tr>
<tr>
<td>v. 22</td>
<td>Sep 18, 2009</td>
<td>05:14</td>
</tr>
<tr>
<td>v. 21</td>
<td>Sep 08, 2009</td>
<td>14:00</td>
</tr>
<tr>
<td>v. 20</td>
<td>Sep 07, 2009</td>
<td>09:50</td>
</tr>
<tr>
<td>v. 19</td>
<td>Sep 06, 2009</td>
<td>22:43</td>
</tr>
<tr>
<td>v. 18</td>
<td>Sep 06, 2009</td>
<td>22:38</td>
</tr>
<tr>
<td>v. 17</td>
<td>Sep 06, 2009</td>
<td>22:28</td>
</tr>
<tr>
<td>v. 16</td>
<td>Sep 01, 2009</td>
<td>18:06</td>
</tr>
<tr>
<td>v. 15</td>
<td>Sep 01, 2009</td>
<td>14:23</td>
</tr>
<tr>
<td>v. 14</td>
<td>Sep 01, 2009</td>
<td>14:19</td>
</tr>
<tr>
<td>v. 13</td>
<td>Aug 13, 2009</td>
<td>05:42</td>
</tr>
<tr>
<td>v. 12</td>
<td>Aug 11, 2009 21:30</td>
<td>Eve Maler: This and all previous revs are &quot;editors' drafts&quot; and have not been approved by the group</td>
</tr>
<tr>
<td>v. 11</td>
<td>Jul 25, 2009 16:29</td>
<td>Eve Maler</td>
</tr>
<tr>
<td>v. 10</td>
<td>Jul 25, 2009 09:51</td>
<td>Eve Maler</td>
</tr>
<tr>
<td>v. 9</td>
<td>Jul 23, 2009 18:08</td>
<td>Eve Maler</td>
</tr>
<tr>
<td>v. 8</td>
<td>Jul 23, 2009 16:43</td>
<td>Eve Maler</td>
</tr>
<tr>
<td>v. 7</td>
<td>Jul 23, 2009 16:34</td>
<td>Eve Maler</td>
</tr>
<tr>
<td>v. 6</td>
<td>Jul 23, 2009 16:30</td>
<td>Eve Maler</td>
</tr>
<tr>
<td>v. 5</td>
<td>Jul 23, 2009 15:14</td>
<td>Eve Maler</td>
</tr>
<tr>
<td>v. 4</td>
<td>Jul 23, 2009 13:55</td>
<td>Eve Maler</td>
</tr>
<tr>
<td>v. 3</td>
<td>Jul 23, 2009 13:33</td>
<td>Eve Maler</td>
</tr>
<tr>
<td>v. 2</td>
<td>Jul 23, 2009 13:31</td>
<td>Eve Maler</td>
</tr>
<tr>
<td>v. 1</td>
<td>Jul 23, 2009 12:53</td>
<td>Eve Maler</td>
</tr>
</tbody>
</table>