generic_issues

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 indirection.

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:

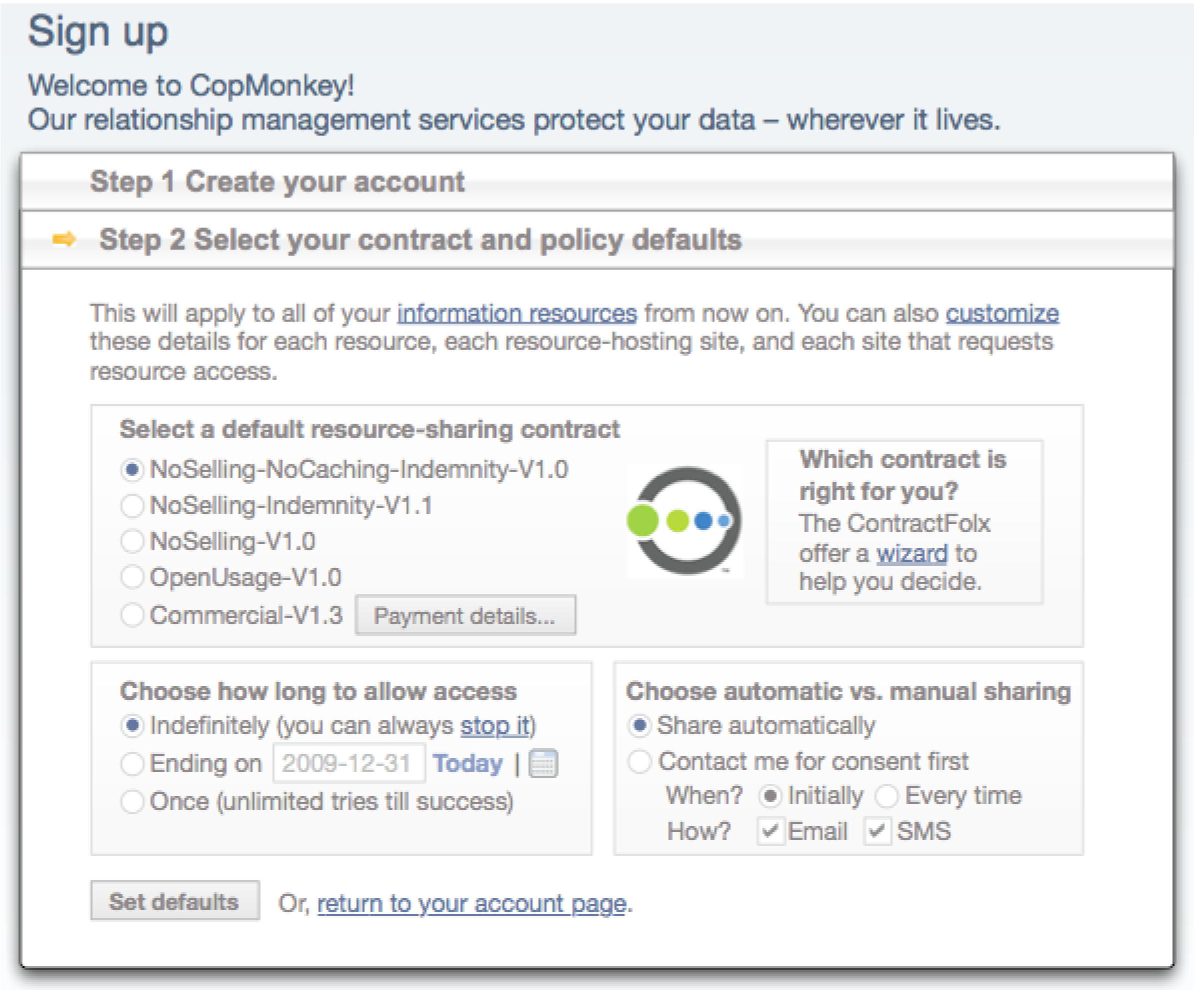
- How long to allow access: once, some number of times, for some period, indefinitely until the user says to stop, etc.

- 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 resistent, 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.