terms_scenarios

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.

Strength of identification	**Pre-configured policy**	**Real-time consent**
Self-asserted label	"Anyone can gain access if they introduce themselves"	"Someone purporting to be 'Random' wants access"
Identity from known issuer	"Let (this identity, this list of identities) from (this issuer, one of these issuers) gain access"	"Requester 'Solid' (verified by 'Known') wants access"

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 soon 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.

Unable to render {include} The included page could not be found.

Unable to render {include} The included page could not be found.

Unable to render {include} The included page could not be found.