Protocol Issues

This issues list has been superseded by the issues list on GitHub.

Access token formats (see issue #30)

How to fill in the access token format property in the AM's hostmeta XRD?

Claims formats (solution under way; see issue #2)

How to fill in the claims format format property in the AM's hostmeta XRD?

XRD vs. JRD (see issue #19)

Should JRD be used instead of XRD? in addition to XRD?

Compliance with OAuth 2.0 (Closed)

We have begun moving the UMA core protocol spec onto an OAuth 2.0 basis, and try to stay on top of any changes on a weekly basis.

Modularity (Closed)

Some UMA features on top of OAuth 2.0 are separable; some seem to be attractive to a wider developer audience even outside the UMA value proposition. Which should be defined in separate specs for greater modularity? Will some pieces be in a position to be submitted to the IETF in the near term, ahead of submitting some other pieces? Some possibilities:

- Dynamic discovery
    - Back-channel token validation
- Claims-required mechanism
- Resource-oriented scope management

We will be opportunistic in splitting out any modular features that are requested for use outside a unified UMA context.

Token-getting flows (Closed)

Need to identify specific OAuth flows (and profile them, as necessary) in Step 2 to cover:

- Alice-to-service (on behalf of Alice)
- Alice-to-service (on its own behalf)
- Alice-to-Bob

Interactions between refresh tokens and the claims-required flow (Closed)

Do we have any need or desire to require refresh tokens to be issued in all cases, perhaps due to the positioning of the claims-required request, or is this a matter purely between a requester (client) and AM (authorization server)?

Paul recommends that this is a matter purely between these two parties. Recommendation APPROVED on 2010-04-08.

Error messages (Closed)

We've been asked for input on whether we need new OAuth error messages beyond the HTTP error layer. Do we need a new OAuth error around "claims-required", or an explicit extension point for an UMA error, or no explicit extension point at all for adding our own flow at this point?

Identity tokens and claims (Closed; see issue #2)

(This relates to the Claims 2.0 signing work, and possibly to the OAuth and OpenID Artifact Binding signing work.)

George Fletcher and Praveen Alavilli have mused on how to create "identity tokens" that would represent a particular user online in a generic way. Is this applicable to UMA scenarios in which the authorizing user or the requesting party must be identified? If so, is it useful for us to standardize it in some spec module? See:

- Protecting "discovery" information
- Open identity token
- Open identity token and personal discovery service

- Continuing the discussion

George has motivated the need for identity tokens in an email thread. The next step is for the group to discuss and validate the need.

How do we solve the bob@gmail.com problem? There are three ways to get a verification of this:

- The requester may have a current session that will suffice, and generates a claim.
- The requester is willing to be an RP to the gmail IdP, and redirects Bob there to log in, and gets the claim and passes it along. (??)
- The requester redirects you synchronously to the AM, which acts as an RP itself. In this case, there's no claim being passed.

Do we have to allow for asking for claims in one language and getting claims back in another? E.g., you might want to ask for an identity claim in JSON /Claims 2.0, but maybe the IdP only speaks SAML. Does the resulting assertion need to be converted into a different form?

Making some simplifying assumptions, is it likely that Alice intends to share **specific** kinds of resources with **specific** other people whose "usernames" at the related requester app sites she already knows? In this case, can can configure the ACL policies with the "right" entries, such that the requester can either use an existing requesting-user session or sign a claim in lightweight fashion on behalf of the requesting user.

Token validation models (Closed)

Several different ideas have been floated:

Real-time remote validation: This is the option currently in the spec. A token validation URL is treated as a protected resource at the AM, which the host accesses in real time by supplying the requester-presented token and describing the type of access that was attempted (resource and method). This has the advantages of allowing the host to have an extremely lightweight structure, and allowing the AM to keep detailed and accurate records of access attempts for display in analytics etc. It has the disadvantage of Internet-scale performance issues.

Hybrid validation: At their leisure after being introduced in step 1, host asks AM (in an as-yet undefined interaction) to provision it with the means to do accurate local validation (probably the AM's certificate) whenever a requester does present a token. This has the advantage of creating little or no delay in the first validation and no delay in subsequent validations. It has the disadvantage of not, all by itself, allowing the AM to know the exact nature of requester attempts.

If hybrid validation is enabled, its disadvantage could be mitigated by allowing or requiring the host to use the back channel forged in step 1 to provide **acce ss logs** to the AM on some schedule.

Whether validation is real-time or hybrid, the host could use the back channel to provision the AM on some schedule with **protected resource descriptions**, such as all the URLs at this host controlled by this authorizing user at this AM, along with "display names" chosen for them at the host. This has the advantages of improving the authorizing user's experience in setting policies and reviewing analytics directly at the AM, and potentially making it easier to offer the feature of letting the user protect different resources at one host with different AMs.

Which of these are useful to standardize?

Maciej has laid out requirements and considerations in an email thread (and the group discussed the issue a tiny bit on 2010-04-01). The next step is to discuss how our use cases inform our need for a technical solution here.

Signed messages (see issue #30)

Does UMA require messages from the requester to the host to be signed in any circumstances, or does it need to ensure that the option is preserved in OAuth, or does it not care?

Paul has made a case that UMA needs no signing of messages in an email thread, and the group further discussed the issue on 2010-04-01. We await more written scenarios from TomH to inform our decision-making here.

(See also the newest OAuth signing proposals.)

Discovery (some parts solved; also see issue #20)

Needs for various kinds of discovery have come to light:

1. The host needs to be introduced to, and to learn about how to trust, a user's chosen AM
2. The requester needs to learn what resources it can get at the host
3. The requester needs to learn what AM endpoint to go to to get an access token for a desired resource

Step 1 of the UMA protocol is all about discovery type #1 in the general case. The question of token validation as discussed above is about how the host can come to trust the AM, whether at introduction time, access time, or somewhere in between.

The hData scenario involves a model of "trusted discovery" that encompasses type #1 and also type #2. The introduction of the host to the AM involves the extra step of the host registering its own metadata with the discovery service associated with the AM, possibly including a manifest of resources and available scopes of access to them, which then becomes relevant to discovery type #2. Is there interest in standardizing this outside of the UMA core protocol?

Regarding discovery type #2, if the user doesn't directly provision the requester with a resource URL and some notion of possible method/scope of access ("Data Dominatrix"), and doesn't advertise the URL broadly ("Hey, Sailor"), the other possibility is that the requester can discover a list of resources and appropriate scopes at a discovery service or resource catalog (shall we call this "Look Me Up"?). This is the approach mooted in the hData scenario through the requester accessing the discovery service mentioned above. (Note that the notion of scoping mechanisms pervades requester->AM, requester->host, and host->AM actions at various points, as discussed below.)

Regarding discovery type #3, the core protocol spec currently says that "If the requester knows, by whatever means, the access token URL for the AM that is protecting the desired resource without first approaching the host, it MAY proceed directly to that URL" (meaning that the requester doesn't have to discover the AM endpoint only by trying to hit the resource first). Given the new WRAP paradigm we are using, the requester could therefore get an access token to use at a host without ever having approached the host before. This amounts to pre-authorization, which we had rejected before (as "bulk" pre-authorization) in the case where a requester wants access to resources at multiple hosts. Do we see any pitfalls here?

Are we missing any other discovery types?

Resource-oriented scope management (Closed)

Resources are represented differently in different systems. E.g., Flickr has its own system of naming and grouping resources. How can UMA handle all these different systems? What if you want to set the same policy over two sets of photos, one at Flickr and one at Picasa? It's suggested that hosts control exactly the granularity of access to sets of resources – it could inform the AM of every single resource it has, or could tell the AM it has "sets" (groups?) of resources. Policies at the AM can only be applied at the level of granularity that the host has informed it of. However, what if a single resource appears in multiple sets? The host needs to give the requester a description of the scope it needs to ask for. This could be "encrypted", with some label that maps to a struct that the host gave the AM earlier.

Optimizing for "baskets" (see issue #31)

The following issue was discussed on 2010-04-22:

Eve advocates that we keep things very simple for how to configure resources and policies into baskets: She imagines that users will likely want all resources in a basket to inherit whatever the basket's policy is. She has been conceiving of baskets as a special category of resource that is hosted by a host that has a special tight relationship with the AM.

She guesses that the whole topic of baskets doesn't really have protocol implications per se, but does have some fairly deep application implementations. Or does it have protocol implications? George asks if a requester has already proved that it has a right to get to the basket, can it proceed to each of the linked resources (on whatever hosts) and get immediate access with the token it was already given? Paul believes that the procedure for having the requester prove itself at each host should remain in place, but that it's possible some of these interactions can, on a per-host basis, be optimized through scope-parameter tricks when the AM issues the first token involving each host.

George wonders if this is missing an opportunity for truly intuitive and useful optimization on the requester's part when accessing all the things in a basket. Should it be possible for the requester to approach each host/protected resource in a basket with some kind of intermediate token saying that it's asking for access as part of a "basket request", so that it doesn't have to re-qualify over and over at the AM, by (e.g.) presenting the same set of claims? Then again, if inheritance works such that the "strongest" policy (according to whose definition??) is inherited by everything in the basket, that could be valuable.

Policy support (see issue #2 for reserved claims support decisions)

What kinds of policies would an AM support? What sorts of policy engines would they use? The guess is that there would be a core set of claims that would be supported, driving a minimum set of policies. Other kinds of policies, such as time-based ones, would likely be popular too. If real-time token validation is performed, then time-based policies (and auditing) could be really precise; otherwise there would be some variability based on access token validity windows.

Scalability (see budget request #UMA WG B

With a centralized component, what are the problems with scale? How can we test the scalability of real-time token validation? It's likely that every user can only have n login sessions a day, but they may have 10n or 100n data-sharing connections happen on their behalf a day.