UMA telecon 2017-03-02

Date and Time

- Thursdays, special meeting times continuing: 8:30-10am PT
  - Screenshare AND dial-in: [http://join.me/findthomas](http://join.me/findthomas)
  - UMA calendar: [http://kantarainitiative.org/confluence/display/uma/Calendar](http://kantarainitiative.org/confluence/display/uma/Calendar)

Agenda

- Roll call
- Approve minutes of UMA telecon 2017-02-23
- Logistics:
  - Last 90-minute meeting is next Thursday; shoot for quorum so we can vote on WG drafts.
  - Interest in ad hoc early next week?
- UMA V2.0 work
  - 2016 roadmap / GitHub issues for V2.0 (all issues to be kept here for the duration) / dynamic swimlane
  - Core is up to 17 (updated before the ad-hoc-that-didn't-happen) and RReg is up to 05 (no change)
  - Work on open RReg issues; see also relevant emails, many presenting options to choose from and some with people weighing in – let’s take issues in this order:
    - Discuss #277 – see also Re issue #270 (for its commentary about issue #277...)
    - Discuss #271 and #272– see also Re issues #271 / #272 (not much additional in the message)
    - Discuss #276 – see Re issue #276
    - Discuss #269 – see Re issue #269
    - Discuss #270– see Re issue #270
    - Discuss #273– see Re issue #273 (we’ve talked about using “last call” to seek OAuth/OIDC usage of the RReg spec)
    - Discuss #31– leave on backlog? close without action? decide on some action?
- AOB

Minutes

Roll call

Quorum was reached.

Approve minutes

Approve minutes of UMA telecon 2017-02-23: APPROVED.

Logistics

- Last 90-minute meeting is next Thursday; shoot for quorum so we can vote on WG drafts.
- Interest in ad hoc early next week?

Eve can do anytime Monday morning PT. Mike and Justin can as well. Let’s make it 8:30-10am PT.

AI: Eve: Put the ad hoc on the calendar.

UMA V2.0 work

#277: Conceptually, should we make it possible to register multiple resources at once so as to align the capabilities of registering resources and requesting permissions? Otherwise, an RS might have to register three resources in three calls, and then it would get to request permissions “in bulk” once. Per the discussion about resource registration timing in the UIG, in some use cases, the two steps could be very close together. Gluu actually has already implemented bulk registration as a kind of extension, based on the logic of calling them “resource sets” and there being a many:many relationship between resources and policies through scopes.

George points out that the answer really depends on the use case (a la the UIG). For use cases for #1, where the RS is using wildcards the AS doesn't understand, the RS only registers one resource. For use cases for #2, for a Share button there would be only one resource, but for "relationship-driven graph DB sharing" you could have many many resources. For use cases for #3, it might be a finite low-ish number as we anticipate for FHIR. (E.g., each of the use cases in the GDoc imagining RS permission requests for various APIs gives us a way of imagining finite numbers.)

Given that there could be a lot of RReg API work in cases of "what do we do about rollback if some of the bulk registrations don't work but the rest do?", acknowledge the fact that we've left this unoptimized for now, and give ourselves room to add a batch registration endpoint later.

This is an explicit question we could ask during "last call".

AI: Eve: Start a "last call" question list, including the RReg optimization question and the RReg applicability to OAuth and OIDC question. Add this to the UIG.

AI: Eve: Extend the discussion in the UIG of resource registration timing with the above discussion.
#271 and #272: First of all, we need some security considerations around sanitizing all such text (including the existing human-readable text fields), to ensure there's no active scripting in there. We don't currently have that; we only have it for the icon URIs. What does sanitizing mean? It seems ill-defined in the case of natural-language text being displayed.

AI: Eve: Insert language about "sanitizing", pointing to a non-normative example of such action, such as this (from George); Jin points to this as the vuln.

Next challenge: Resource description documents have an RO context, and the RS is basically responsible for doing any L10N before the human-readable strings get to the AS; not our problem for I18N. But the human-readable strings involved in a scope description do have this problem. Do we want to suggest the OIDC-style "hash trick"? We discussed this in the email around #269. Justin likes the normative language in RFC 7591 better. George suggests making this being a "developer-readable" mechanism only.

We agree that it's the right pattern to have both the name and the description fields for both. We prefer the dynamic client registration definition of the I18N/L10N mechanism, and will point to its mechanism by reference for both the resource and scope name and description fields. 7591 already covers the circumstance of "any human-facing fields", so we don't have to add that.

#276: What's already in the spec shows that this can be closed.

#269: Note that OIDC's discovery doc already has scopes_supported, which, if filled with URIs, enables static discovery of an UMA scope description document if you want it to. So a closed ecosystem of the sort James mentioned in the #269 thread could be handled this way. (And in the decision just above, we already have an I18N mechanism.)

So option 4 from the email thread is the current best option on the table: "Change even more to say that scope description metadata is possible to standardize, publicize, and reference through a URI, but there is no expectation of run-time resolution by the AS; rather any customization of policy condition setting interfaces with respect to scopes is, um, out of scope." So far, most implementations are only handling the simple strings, while one implementation is drawing the relevant text at run time. Since there's a way to do it statically but machine-readably, let's drop the MUST. Clear consensus on this.

#273: There is a shallow way to treat this issue, and a deep way, and maybe a medium way.

1. A shallow way is s/user/resource owner/. This gets rid of the jarring impact of seeing "user", considering that Mike has implemented RReg for an OAuth-based API protection use case (as in the first bullet of the RReg Intro), using the client credentials grant.
2. Another shallow way (in addition to #1, presumably) might be to require OAuth for protecting the RReg endpoint, which requires providing a resource owner context.
3. The deep way might be something like absorbing the entire RReg spec into the UMA Core spec, if we think it's truly incompatible with OAuth and OpenID Connect use cases generally.
4. A medium way might be refactoring the RReg spec to draw the dividing line in a different place, such as having RReg merely provide the bare bones of the API, and having Core provide extension properties to do all the "policy-setting-relevant" work, such as the name, icon_uri, and description fields. This would lengthen Core kind of significantly, but keep an interesting RReg module.

Eve and Mike aren't crazy about #4 because it's invasive and really lengthens Core, and it isn't proven yet that RReg as it stands isn't relevant to all three of the technologies or others yet to come.

AI: Eve: Send out dedicated emails about #273 and #31.

**Attendees**

As of 19 Jan 2017 (post-meeting), quorum is 5 of 8. (Domenico, Sal, Andi, Maciej, Eve, Mike, Cigdem, Sarah)

1. Sal
2. Andi
3. Eve
4. Mike
5. Cigdem
6. Sarah

Non-voting participants:

- Justin
- Dean Grannes - new participant
- George
- Jin

Regrets:

- JohnW
- Domenico