Albert started to recap where we are & to introduce the new proposal.

A couple of years ago the SeamlessAccess (SA) working group made some recommendations, the first one was to disambiguate display name. However, this pushes decision to user. We are now looking at filtering (not disambiguation) at infrastructure level. There’s a proposal to develop new metadata "trustinfo" to facilitate filtering. Albert’s use case is that of a fedop using SeamlessAccess as the federation default, so the fedop must ensure that only IdPs which are supposed to work in that federation will work. However the introduction of a new metadata schema may be problematic for metadata producers, aggregators and consumers (especially those which validate the XML). We are now looking to introduce an entity attribute (EA) which points to a trustinfo profile. As a second priority, we will consider the more general trustinfo use case.

Marco’s proposal: Use mdrpi:PublicationInfo for filtering? PublicationInfo is a MUST in the eduGAIN SAML profile. Could be added to to any trustinfo profile.

We had a short digression on where does SeamlessAccess get metadata:

- OpenAthens
- eduGAIN
- A couple of federations directly (because they don’t export all IdPs to eduGAIN)
- Some other sources including bilateral

As an aside: Alex has asked OpenAthens to add RegistrationAuthority to their published metadata.

There was wide agreement that it would be easier for fedops and metadata aggregators and consumers to deal with a single entity attribute which refers to a trustinfo profile. We noted that the profile must be implemented somewhere:

1. SeamlessAccess registers the profile and the EA refers to it. The profile still needs to be defined, perhaps as trustinfo metadata but could be javascript or other code. SeamlessAccess doesn’t want to have to maintain a lot of rules so we must ensure that the filtering rules are fairly static and that many SPs use the same one. If SeamlessAccess controls, they could say only for fedops and not invididual SPs, and this could be used to limit the number of profile; recall that we have the more general case to consider.
2. fedop puts out blob of actual trustinfo ruleset. fedop needs to host and make it available. It’s a relatively static signed XML file so straightforward to host (and not as difficult as MDQ high availability). SA would need to validate the XML; it couldn’t use its own internal representation in Javascript.

We noted that most of the work about where the profile is located is really a SeamlessAccess issue, and that this WG might be only for standardising the EC and facilitating a discussion.

Action on Albert and Alex: draft examples of the two options to give concrete examples to show to the group & to the Seamless

Post meeting update (Albert):

I agreed to providing a concrete example of how that pointer idea might work… Here it is: This is an example of how UCLA’s wiki would invoke the InCommon IdP filter in SeamlessAccess:

1. UCLA’s metadata administrator checks the option to use InCommon Discovery in our metadata management tool for UCLA wiki (SP)
2. the Tool injects the following entity attribute (in bold) into the SP’s metadata:
NOTE: attribute name and value are placeholder examples.

SeamlessAccess, upon receiving a request from SP to perform discovery, reads the SP metadata, uses that attribute value (http://seamlessaccess.org/trust-profile/incommon-wayfinder) to look up the associated Trustinfo (or equivalent) blob. If the blob were Trustinfo, it'd look like:

```xml
<ti:TrustInfo xmlns:ti="urn:oasis:names:tc:SAML:metadata:trustinfo">
    <ti:FallbackHandler>https://www.example.org/about</ti:FallbackHandler>
    <ti:TrustedEntities match="md_source"/>https://incommon.org</ti:TrustedEntities>
</ti:TrustProfile>
</ti:TrustInfo>
```

NOTE: the md_source value is a a placeholder example for illustration purpose.

“InCommon Wayfinder” is the name we are adopting for the SeamlessAccess-backed InCommon Federation Discovery Service. Its filtering rule happens to be quite simple and can be expressed using the md_source MatchType. One could just as easily define highly complex rules here as well.