Authorization Standard API Use Cases

These are some use cases for the Groups API.

(All examples are non-normative)

Shibboleth IdP.

An IdP queries the GWS for all groups that a user is a member of. It uses the name and other attributes to generate isMemberOf and eduPersonEntitlement attributes. Two examples:

Simple memberOf assertions.

Idp does a GET search for groups where the user is an effective member, requesting **name** attribute.

Computes memberOf assertion values:

```
urn:mace:washington.edu:groups:uw_staff
urn:mace:washington.edu:groups:uw_member
urn:mace:washington.edu:groups:u_weblogin_admin. . .
```

Computed eduPersonEntitlement assertions.

Idp does a GET search for groups where the user is an effective member, requesting specific course attributes: **year quarter, SLN**.

Computes eduPersonEntitlement assertion values:

```
urn:mace:washington.edu:courses:year:quarter:sln
. . .
```

Web application member list.

A web application uses groups for its membership list. Users can self-enroll. Authn is provided by a local SSO login, Shib login at an InCommon federation site, or by a Google login. In the first case a local netid is the user's identity; in the second it is an ePPN; in the third an email address. The three cases might have uris like "id:*users_local_id*", "eppn:*users_remote_eppn*", "email:*users_email_address*".

1. On authn, check if the remote user is already a member
2. If not,
    a. if local user, PUT member to group
    b. else:
        i. PUT remote subject to group system
        ii. PUT remote subject to group as member

Web application with user managed groups.

As part of its services, a web application allows its users to create and manage groups of other users and groups. For this it provides a simple, custom group UI. An example might be a classroom tool that allows instructors to create groups containing registered students plus drop-ins and other add-ons.

The application administers a stem in the group service. It creates a sub-stem for each user. Any groups created for the user are on that user stem.User stem name might be: *app_stem_name* + *separator* + *user_id*.

1. When the user enters the site, check for existing groups:
    a. GET stem by name for the user stem.
    b. GET search for existing groups with "stem=*user's_stem_name*".
2. When the user adds a member to one of her groups:
    a. PUT the member to the group
3. When a user creates a group:
    a. if no user stem yet:
        i. PUT the stem.
    b. PUT the group