Design Objectives

General Requirements and Design Principles which Determine Success

The following are identified as broad, general requirements that supplement the collected use cases and permissions matrix developed during our analysis phase. These identify gaps in existing behavior, as they both clarify and expand on existing functionality in Hyrax.

1. Permissions are extensible to locally customized features, in addition to existing Hyrax functionality
2. Permissions should apply comprehensively and retroactively (beyond current Hyrax behavior)
   - 1. There are some use cases for this retroactive assignment to be optional
   - 3. Expand workflow capabilities beyond the current 1:1 constraint with Admin Sets so that workflows can be applied to additional actions (delete, etc)
      - 1. Ability to perform Deletion actions must be configurable (vs. current Hyrax ability)
   - 4. Expand workflow capabilities beyond just works, so higher level actions can also be configurable (flag collections for deletion)
      - 1. Ability to perform Deletion actions must be configurable (vs. current Hyrax ability)
3. Provide pre-configured application Roles
   - 1. abilities can be locally customized
   - 2. ability to create local, custom Roles
4. Provide ability to manage permissions via GUI
   - 1. Assignment of permissions is non-disruptive... can be applied without application redeployment
5. Enable assignment and management of permissions to members of (works within) Collections vs. current Admin Set permissions management structure
6. Support for integration with institutional directory-based groups (e.g. Directory Services/AD)
7. Support for simple IP range based access
8. Separate intellectual "ownership" concept from asset management permissions
   - 1. Intellectual ownership, stewardship, and provenance information is handled via metadata, not application logic
      - i. creator of original content
      - ii. primary contact (department or user responsible for questions and information)
      - iii. "My" information, for reporting & analytics context
      - iv. original depositor
      - v. person on whose behalf the work was deposited
   - 2. Asset management means ability to create, modify, view, delete entities (i.e. manager role)
   - 3. Hyrax currently includes a "transfer ownership" option. This WG has been able to determine[1] that this ownership transfer is easily addressed by implementing the separation of intellectual ownership from asset management permissions proposed here.
9. Ability to broadly differentiate staff activities from end-user/public activities (e.g. "administrative view" of repository entities and metadata)
10. Design must allow for an ability to easily audit who can perform what actions in the application and troubleshoot permission failures

Other Design Goals

The following goals should inform the design, but there is no implied commitment that all will be implemented.

1. Ideally, this would be implemented in a way allowing use beyond only Hyrax adopters.
2. Designed with a well-formed API plugin interface.
3. Expunge the concept of asking is_admin? in favor of asking what the given user can do
4. An API-based group implementation to easily incorporate an institution's identity management system which is separate from but interacts with the application permissioning system.
5. The documented use cases should guide the design.
   - a. Many of the batch use cases are not currently available in Hyrax, but may be implemented where feasible.
   - b. Use cases may not have broad enough community needs to justify community development.

Remaining Questions

1. What is the desired relation to CanCan? Samvera has a long-standing tradition of leveraging CanCan and Blacklight for permissions.

Notes

1 - From Mike Giarlo on slack on 9/6/2018 - This was originally developed in Sufia for ScholarSphere by the folks at Penn State. My recollection is that the ownership transfer feature was intended both to change the "depositor" of a work and, coupled with the custom rights datastream (in the parlance contemporaneous w/ this feature's initial development) ensuring the depositor always has edit access, to grant/ensure edit access. You can consider it as an ex post facto execution of a proxy deposit. (The nomenclature is clumsy here since—at that time, at least—there was no stronger notion of ownership in our stack than setting a user as depositor and granting them edit access.) (FWIW, the latter bit about custom rights preventing removing edit access from depositors has since been removed from Hyrax. I believe this was done in order to allow for mediated workflows' ability to remove edit access temporarily from depositors.)