Hydra Authorization Use Case

Hydra does not rely on Fedora 3's AuthN/Z capabilities, but rather stores it access policy on the "rightsMetadata" datastream of Fedora objects. The handling of access controls is then managed at the application level above Fedora.

The intent of this document is twofold:

1. To clearly record how Hydra is currently designed to manage user access to resources across:
    - Fedora REST API
    - Fedora Resource Index
    - Repository Search
2. To describe how Hydra could potentially leverage the nascent Fedora 4 AuthN/Z framework

Hydra AuthN/Z Design

A good and fairly up-to-date overview of basic Hydra access controls can be found at https://github.com/projecthydra/hydra-head/wiki/Access-Controls-with-Hydra.

In addition to object-level permissions recorded in the rightsMetadata datastream, an object can have a relationship (via RELS-EXT) to special "admin policy object" (APO) which stores "inheritable" permissions in a "defaultRights" datastream using the same XML schema as rightsMetadata. The inheritable permissions of an APO are effectively **added** to the object's rightsMetadata in determining authorization.

For the actual application-level (Ruby on Rails) authorization enforcement mechanism, Hydra uses the CanCan library (or the CanCanCan fork in hydra-head 7.0). Additional access controls – for example, for types of access not specifically provided in rightsMetadata – can be defined in the Ability class. A simple example can be seen in Hydra::Ability, which provides a special "download" permission on datastreams (by default mapping to read permission on the object).

Fedora REST API

Hydra uses the Rubydora library (via ActiveFedora) to handle Fedora REST API requests and responses. The credentials of the Hydra application authenticated user, however, are **not** used for authorization of Fedora REST API requests. A single Fedora user account provided in a configuration file is used for access to restricted methods (API-M).

Fedora Resource Index

Hydra makes no direct usage of the Fedora RI, since it relies on Solr for repository queries (via Blacklight and ActiveFedora). (ActiveFedora::Indexing#reindex_everything does use findObjects because it's intended to rebuild the Solr index from Fedora, but this is a maintenance operation and probably not pertinent to user-level AuthN/Z considerations.)

Repository Search

The principal means that Hydra enforces access controls on repository search is by adding permissions-related clauses to the Solr select queries.

Hydra and Fedora 4

The potential delegation of AuthN/Z responsibility from Hydra to Fedora 4 is captured in the following use cases.

Title (Goal)	Use the hierarchical node structure to manage access controls on deeply nested resources
Primary Actor	
Scope	
Level	
Story (A paragraph or two describing what happens)	I have a complex structure of objects with more than two levels of hierarchy – A is parent of B, which is parent of C – and I want the access controls on A to govern access to C without having to create an explicit relationship between A and C (or relate both A and C to some other governing object).

Title (Goal)	Manage datastream-level permissions in the same manner as object-level permissions
Primary Actor	
Scope	
Level	

Story (A paragraph or two describing what happens)	I want to apply custom access controls to specific datastreams (such as those storing preservation master images) of objects without custom programming.

Title (Goal)	Leases and Embargos
Primary Actor	
Scope	
Level	
Story (A paragraph or two describing what happens)	Support leases and embargos that can go from an arbitrary access state before the expiration/release to another arbitrary access state. Definitions: Lease: Access is restricted after an expiration date. Embargo: Access is restricted until a release date. Access state: A set of groups or users who may view or discover an object at a particular point in time.