Art Institute of Chicago Use Cases

Access Policies via Models

Title (Goal)	Support Fedora 3-style object classes (content models) - Access Policies via Models
Primary Actor	Repository architect & implementer
Scope	Data architecture and access
Level	High
Story (A paragraph or two describing what happens)	As a repository manager, 1. I can associate access policies to "content models"

Examples

1. Given the myns:image object, I want to create some access policies that allow only users in imaging group to view and update that object.
2. If a user adds the mymix:published_web mixin, access to that object, some of its properties, and the web_thumbnail, web_small and web_large datastreams is granted to web users.
3. If a user removes the mymix:published_web mixin, access to the object, all its properties and children is revoked for all web users.

Content Model API

Title (Goal)	Support Fedora 3-style object classes (content models) - Content Model API
Primary Actor	Repository architect & implementer
Scope	Data architecture and access
Level	High
Story (A paragraph or two describing what happens)	As a repository manager, 1. I will be able to associate "content models" to objects via an API 2. I will be able to disassociate objects from "content models" via an API a. In so doing, services associated with those "content models" are no longer available on those objects 3. I will be able to associate objects with a single content model only 4. I will be able to define relationships between objects

Object Services

Title (Goal)	Support Fedora 3-style object classes (content models) - Object Services

Primary Actor	Repository architect & implementer
Scope	Data architecture and access
Level	High
Story (A paragraph or two describing what happens)	1. As a repository manager, a. I will be able to associate services with "content models" i. Example services include the generation of image derivatives, and the serving of images 2. As a repository user, a. I will be able to access resources via services that are associated with defined "content models"

Examples

a. Given the myns:image node above, I want to run a method (sequencer?) that creates/updates a thumbnail datastream under it every time the master datastream is updated.
b. The myns:image type constraints assures that master is always present so if a new myns:image is created, a master child is expected to run the method on.
c. Some users should be able to attach a pre-defined mymix:published_web mixin to the same object; when that happens, a sequencer is there waiting to create aweb_thumbnail, a web_small and web_large datastreams as children of the object.
d. Conversely, if the user removes mymix:published_web, the web_thumbnail,web_small and web_largedatastreams are destroyed.

Challenges

1. How do we handle the creation process so that the object does not go through validation before all mandatory children are ingested?

Issues / limitations

1. A mixin can define a mandatory type, but the mixin itself is not mandatory. If some editor removes the mixin, that would break the thumbnail method. In order to prevent that, access policies can be defined for who can remove or add a certain mixin; but for basic functionality not meant to be ever removed, a primary type would solve the problem more easily and elegantly.

- Stefano Cossu: tests should be made with transactions for Challenge #1 when https://www.pivotaltracker.com/s/projects/684825/stories/64058980 is fixed. If I am able to create a node with a primary type or mix-in that defines a mandatory child , and that child as well within a transaction, and the validation is performed ONLY after the transaction is committed, this would solve the issue.

Structural Validation

Title (Goal)	Support Fedora 3-style object classes (content models) - Structural Validation
Primary Actor	Repository architect & implementer
Scope	Data architecture and access
Level	High
Story (A paragraph or two describing what happens)	As a repository manager, a. I can define "content models" that ensure the presence of defined datastreams i. A defined datastream has a defined name and a defined mime-type b. I can define which type(s), name(s) and number of children or properties a Fedora node can have c. Child nodes and properties introduced by a mix-in "content model" are removed when that mix-in is un-assigned, if no other content models depend on them.

Examples

1. I have a myns:image asset type that is auto-assigned to assets ingested by Imaging department.
2. myns:image has mandatory properties and/or children such as a master datastream, of type nt:file or a subtype thereof.
3. myns:image assets can only have children of nt:file type. Ideally, that nt:file should be within a range of defined MIME types (not a critical feature for now)
4. I need a validation mechanism that throws an error if an user adds or updates a child or property that doesn't conform to that definition.

Issues / limitations

1. The default primary type, nt:folder, allows all Fedora nodes to have children of any type, with any name, in any number. There is no way to restrict that with Fedora's current tools.
2. The auto-assigned mixin type, fedora:resource, allows nodes to have properties of any type, with any name, in any number. Ditto as above.
3. If a mix-in is removed that defines some properties and/or child nodes, currently these properties/child nodes are not removed. It is not easy to find which properties/child nodes were introduced by a content model, in order to "cleanly" remove it.
    a. Bad solution: mirror the content model schema in the client systems that are adding/removing content models so they know which properties/children can be removed along with the content model.
    b. Better solution: expose content model schema via REST API methods (e.g. provide more details in /rest/fcr:nodetypes)
    c. Another solution: provide a REST API method that automatically removes all properties/children before removing the content model (in one transaction, so no mandatory constraints are violated).

Use case: AIC type hierarchy

att_D-AIC_JCR_classes.pdf

Structural Validation - Properties

Title (Goal)	Support Fedora 3-style object classes (content models) - Structural Validation - Properties
Primary Actor	Repository architect & implementer
Scope	Data architecture and access
Level	High
Story (A paragraph or two describing what happens)	As a repository manager, 1. I can create "partial match" restrictions on content model properties

Example

1. Given the myns:image object, I want to make it able to have any number of metadata children. Children should be of type myns:meta or a subtype thereof, and their name should start with meta_ (e.g. meta_exif, meta_geo, etc.)

Limitations

1. There is no way, even in CND, to achieve this.