Art Institute of Chicago Use Case - Structural Validation

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