The Hydra Common Data Model

- Esme Cowles, University of California San Diego, escowles@ucsd.edu
- Robert Sanderson, Stanford University, azaroth@stanford.edu
- Jon Stroop, Princeton University, jstroop@princeton.edu

One of the many successes of the Hydra community is the fundamental notion from which its name is derived—the concept of many interfaces ("heads") over top of a single repository (the "body"). The recent release of Fedora 4, with its internal RDF-centric model, has spurred efforts for a community-wide model of collections and works, such that the heads can be sure that the body will behave as they expect it to. That model has been designed and vetted by the Hydra community, and its architecture and initial implementations will be presented in this paper.

Introduction

The Hydra Common Data Model is a flexible, extensible domain model that is intended to underlie a wide array of repository and digital asset management applications. Previously, each Hydra implementer developed their own models tailored to their local use cases. Some Hydra heads had simple models that only supported a flat structure, and others had models that supported varying amounts of complex object structure. While there were Hydra heads developed as shared tools, adopting them required also adopting the models they implemented.

The primary objective of this model is to establish a framework that tool developers can use for working with repository data in a general way. Given this core goal of interoperability, the fundamentals of the model are focused on relationships between resources and access control, since these are critical to shared tool-building and are not domain specific.

To ensure adoption, this model must elegantly support the simplest use cases, such as a single user-contributed file with a few metadata fields, but also support complex use cases, including rich hierarchies of inter-related collections, works, and files. It must provide a compact interface that tool developers can easily implement, but also be extensible enough for adopters to customize to their local needs.

As the Hydra community[1] migrates *en masse* to Fedora 4[2], the community is taking advantage of Fedora 4's enhanced RDF support. The data model promotes linked data best practices, such as using URIs to identify all resources, using well-known vocabularies where possible, and adopting standard interaction patterns where they exist.

Model

The model makes two fundamental distinctions:

1. The difference between a Collection and an Object, and
2. The difference between a Resource (Collection or Object) and its constituent File(s).

A Collection is a grouping of Resources, which might include either Objects, other Collections, or both. Collections and Objects can be part of multiple Collections at the same time, allowing for modeling multiple hierarchies simultaneously such as an Object's participation in a temporary exhibit without disrupting its membership in a more stable curated Collection.

An Object is the abstract notion of the types of works and their component parts which are commonly found in cultural heritage institutions, be they digital or physical. Objects may contain other Objects, such as an atlas containing maps, an archived letter containing pages, or a sculpture containing distinguishable parts. This second hierarchy allows for the identification and description of arbitrary segments without the encumberance of domain-specific modeling concerns like the distinction between journal issues and volumes, they are all just Objects with metadata.

A File is a single, managed bitstream which must be associated with exactly one Collection or Object, and is typically a representation of that Resource. Each Object or Collection can have multiple Files associated with it. For example, an Object that represents a page in a book may have a large image, a thumbnail image, and a text transcription associated with it. Each File may have technical metadata asserted about it.

This results in the architecture depicted in Figure 1.

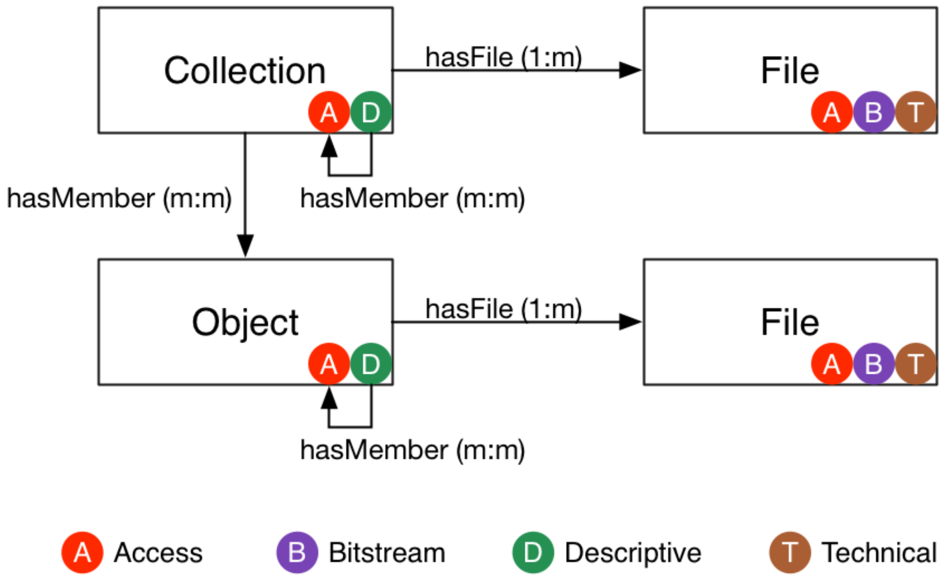

Figure 1. Hydra Common Data Model

Functional Capabilities

These distinctions allow us to assign functional capabilities to the different types. All types are *identifiable* by assigning them a URI, and *protectable* by asserting access controls – the A in the diagram above.

Collections and Objects are *describable* (the D in Figure 1) as they may have descriptive metadata, such as creator, dates, subjects and so forth. This model makes no attempt to define which properties are required, but extensions and application profiles may be written in the future to do so. They are also both *collectible*, meaning they can be included within a Collection. Similarly, Objects are *composable* within other Objects.

Files are *attachable* as they are associated with Collections and Objects. Their metadata descriptions are technical rather than descriptive, and the model does define a baseline set including file size, checksums, and media type.

Finally, Collections and Objects are *orderable* within their parent Collections and Objects. This requires an additional construction within the model.

Order

Expressing order in a graph, such as RDF, is a problem that has been tackled from many different angles. Of the many different possibilities, the ORE Proxy[3] pattern was chosen for ordering resources within a Collection or Object. The primary driver for this decision was the ability to overlay the ordering information on top of the basic model, as it is not a requirement for all types of resources.

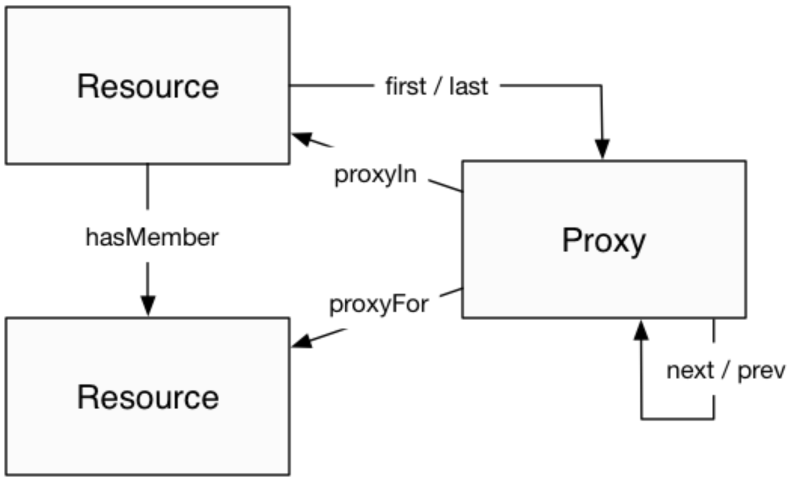

Figure 2. Proxy construction for Ordering

Every resource to be ordered has a Proxy resource associated with it and the resource in which it is to be ordered. For example, to order a set of Objects within a Collection, every Object will have a Proxy that references both the Object and Collection. Ordering relationships are then expressed about the Proxy nodes, to ensure that order from one Collection does not become conflated with order in a second Collection containing some overlapping subset of Objects. The ordering relationships chosen come from the IANA registry for link relation types[4], and consist of first and last from the containing resource to the member resource's Proxy, plus next and prev relationships between Proxies. For resources that appear multiple times in the same order, multiple Proxy nodes may be created. Resources can be omitted from the ordered set by not creating Proxy nodes for them.

Implementation

Fedora 4 implements the Linked Data Platform API[5], now a full technical recommendation from the W3C. To ensure that clients can easily make use of the model, some patterns have been defined for interactions with the repository. The patterns are drawn from basic REST[6] principles and follow the best practices of Linked Data.

The primary interactions revolve around the notion of Containers that enable the creation of resources and the automatic inclusion of those new resources within referenced Collections or Objects. For each Collection or Object, there are Containers for its members, proxies for those members, and attached Files. Creating and deleting resources in those Containers ensures the consistency of the repository, rather than manually updating RDF triples across multiple resources.

Fedora 4 also implements the *describedby* link relationship for binary resources. When a bitstream is dereferenced or created, the repository returns a link header on the response to the RDF metadata about that resource, in this case the technical metadata. This allows the client to simply create the bitstream by POSTing its representation to a Container, and then PUTting the technical metadata to the URI created for that purpose.

By making use of Linked Data, standards and best practices, the Hydra community has defined a common model that will underlie all of its shared systems. Interoperability across the board will allow disparate tools to co-exist, while not preventing customization and specialization for individual domain usage.

References

[1] Hydra: http://projecthydra.org/
[2] Fedora 4: https://wiki.duraspace.org/display/FF/Fedora+Repository+Home
[3] ORE: http://www.openarchives.org/ore/
[4] Link Relation Types: http://goo.gl/07sSm3
[5] LDP: http://www.w3.org/TR/ldp/
[6] REST: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Acknowledgements

This is one output of the collaborative work by many people in the Hydra Community, including notable contributions from Joe Atzberger, Esme Cowles, Justin Coyne, Karen Estlund, Michael Giarlo, Jeremy Friesen, Tom Johnson, Lynette Rayle, Robert Sanderson, Thomas Scherz, Jon Stroop, and Adam Wead.