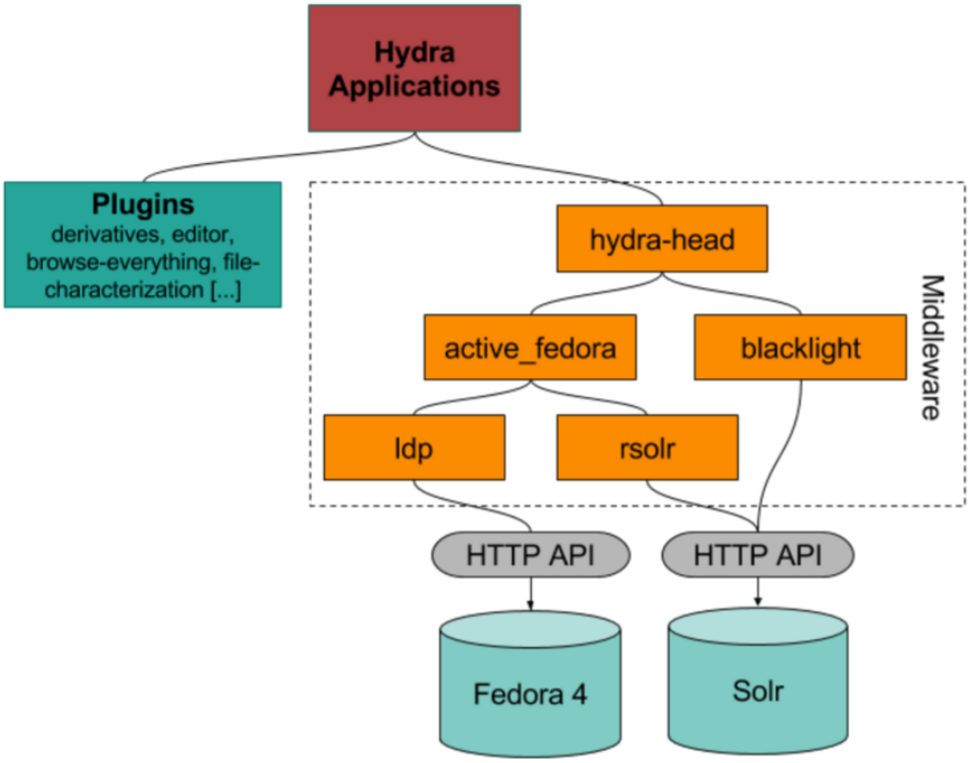
Technical Framework and its Parts

2016 Architecture and Components



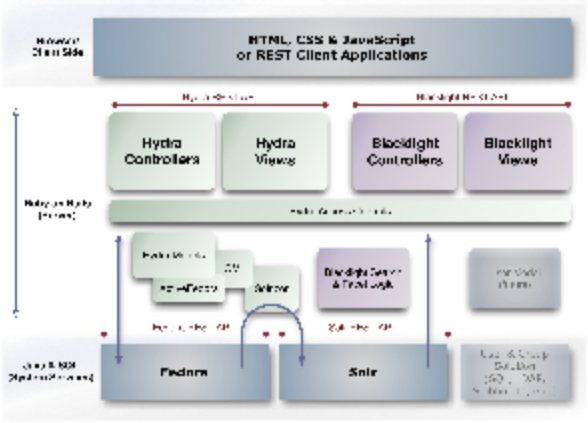
Architecture and Components

Here is a ***Series of Diagrams Explaining the Components of the Hydra Framework*** (from April 2012):

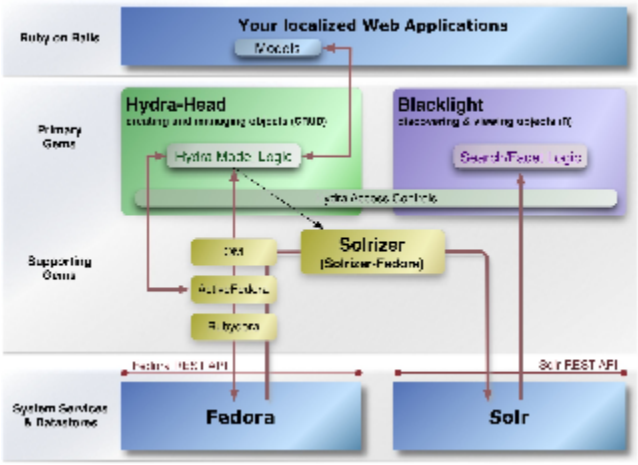
- PDF
- PowerPoint

The Hiearchy of Promises document is another view of the Hydra Stack.

Here are two individual diagrams showing the Hydra framework:



Prepared for OR11, June 2011



Prepared for LibDevConX and Hydra-partners, March 2012 (draft)

The primary components of the Hydra technical framework are:

- ***Fedora*** , providing a robust, durable repository layer for persisting and managing digital objects. Fedora's disseminator features allow us to place an abstraction layer between it and our Hydra heads, shielding an institution's applications from any future changes to the repository structure.
- ***Solr*** indexes, providing fast access to information about the institution's resources. Solr can be used as a lingua franca: content from any source that can generate a Solr index (perhaps an OPAC, or repository metadata records with different schema) can potentially be brought into a Hydra discovery environment.
- ***Blacklight*** , a Ruby on Rails gem that provides faceted searching, browsing and tailored views of objects
- ***HydraHead*** , a Ruby on Rails gem that works with ***ActiveFedora*** to provide create, update and delete actions against objects in the repository, as well as to support various content management actions (e.g., upload file, edit metadata, change permissions)
- A suite of web-based services, supporting granular actions against content to support their management, access and preservation (e.g., checksumming, indexing, transform MARC to MODS, djatoka-based JPEG2000 image streaming)

Finally, these components rely on several background services:

- *authorization*, provided by FESL (Fedora Enhanced Security Layer - a new Fedora framework service part funded by the Hydra partners and others in the community)
- *authentication*, provided by local institutional systems
- *workflow*, which can either be provided as a bundled part of the Hydra framework, or provided by a local institutional systems

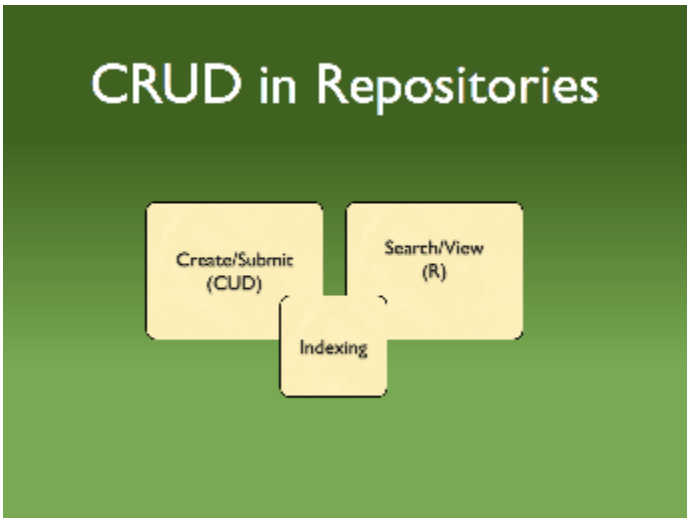
Within the framework, the Hydra Project itself directly supports development and maintenance of the following:

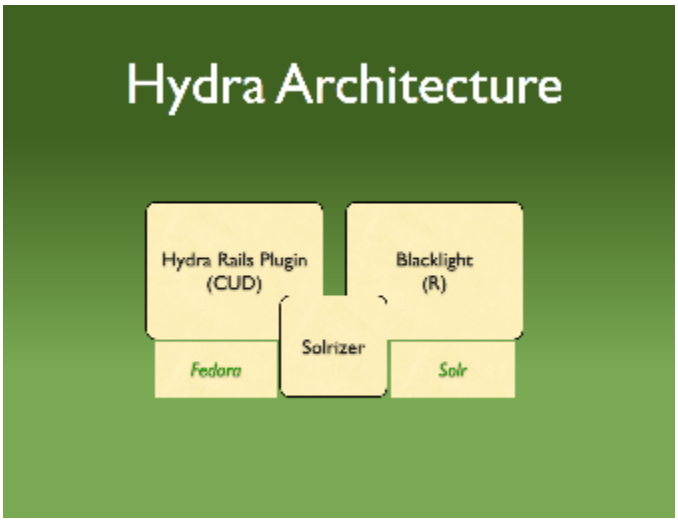
- Active Fedora
- HydraHead
- Hydra-jetty
- Jettywrapper
- OM (Opinionated metadata)
- Rubydora

- Solrizer
- Solrizer-Fedora

The purpose of each of these components is explained in our Hierarchy of Promises.

CRUD in Repositories





This first diagram shows the most common pattern of how you Create, Retrieve, Update and Delete (CRUD) objects in a Repository-driven application. The second diagram shows how the Hydra Framework applies that pattern by using the Hydra Plugin on top of Fedora for managing content (*Create, Update* and *Delete* operations) and Blacklight on top of Solr for Search and Discovery (*Read* operations) with Solrizer handling the crossover -- ensuring that anything stored in Fedora is indexed appropriately in Solr.

Taken altogether, these technical components support the following five primitive functions, essential for any digital asset management solution:

- *Deposit* - uploading simple or multi-part objects, singly or in bulk
- *Manage* - editing and updating an object's content, metadata and permissions
- *Search* - full-text and fielded search supporting user discovery as well as administration
- *Browse* - sequential viewing of objects by collection, attribute or ad hoc filtering
- *Deliver* - viewing, downloading and otherwise disseminating objects through Hydra applications, web services and third party applications

Hydra Terms

Hydra Framework refers to this entire structure - Hydra Plugin + Blacklight + Solrizer sitting on top of Fedora & Solr, as well as implementations & documentation of the recommended Hydra Content Models

Hydra Head Plugin -- one part of the Framework. It is a Rails Plugin (Note: Blacklight is also a Rails Plugin) that combines ActiveFedora, Opinionated Metadata (OM) and a number of other libraries to provide the baseline functionality and tools that you need in order to make a Hydra Head that Creates, Updates and Deletes Fedora Objects.

Hydra H eads -- full stack, user-facing solutions that use the Hydra Framework to provide a specialized interface for specific users, content types and workflows. **Hydrangea** is the reference example of a Hydra Head.

Hydra C ontent Types -- reusable code that defines the structure & metadata for a given type of content along with information about how to display objects of that type as search results, object detail, and edit forms.

The Hydra Head Plugin

The Hydra Head Plugin code, as well as further info about how to use it, is hosted on github at http://github.com/projecthydra/hydra-head

The unique code within the Hydra Head Plugin includes

- tools for constructing web forms to edit your objects and their metadata
- a javascript library for adding the interactive behaviors that you see in Hydra Heads
- support for file uploads and managing/displaying "child" objects
- the REST API that allows clients (primarily the browser) to Create, Read, Update and Delete (CRUD) objects managed by the Hydra Head
- access controls enforcement based on the Hydra Rights Metadata Schema

The Plugin also relies on two independent libraries to do some of the heavy lifting:

- ActiveFedora provides the ability to model Fedora content and perform operations on Fedora Objects
- OM provides the ability to map between your application's terminology and the underlying XML metadata structures

Note: Though the Hydra Plugin is one distinct part of the whole framework, it can only function when used with the rest of the framework. This is because the Plugin is designed to *extend* Blacklight by adding Create and Edit behaviors to the existing Blacklight experience. Likewise, the Plugin needs Solrizer in order to ensure that everything is indexed into Solr. The Plugin also assumes that you have both Fedora and Solr running underneath.

The Point of Having a Framework

Notice that a Robot or a Service, neither of which generates metadata editing forms or search interfaces, might only need ActiveFedora & OM in order to do its job. It doesn't need the full Hydra Plugin, nor Blacklight. However, it does need some assurance that it is doing things in the same way as the Plugin. This is where the idea of the Framework becomes especially important -- so we can build robots, services, and any other sort of utilities that will merely conform to the assumptions and expectations of the Framework. Ideally, the Framework not only defines how an application or tool should work, but actually makes it easier to do things, so that the *correct* way to implement something is also the *easiest* way.