High Level Storage

Overview

This section is for discussion and description of a proposed "high-level" storage interface. The notion of a new storage interface originated with use cases that were hard to satisfy using the current implementation in Fedora. Subsequent discussions have led to a realization that "storage" is a much larger issue with great potential both within Fedora and within a greater infrastructure. In this section you will find references to the earliest work with respect to Fedora and draft proposals that were aimed towards concrete implementations for the Fedora Repository. You can also find and participate in the wider discussion about "storage" that resulted from our initial attempts to design the Fedora-specific implementation and led to our beginning to understand this subject extends far beyond Fedora. In a nutshell, this forum aims to define an architecture for storage (a.k.a persistence) which is of general use and it suitable to support short and long-term access and persistence of digital assets regardless of the underlying physical mechanism. But also, there needs to be immediate implementation of new persistent components to meet current needs. This forum will also be used to design and facilitate implementation of usable components.

To keep this page short, it will mostly consist of organizing links to other pages. Please note this work will cross link to other participants notable Fedora Create, the Data Conservancy, Policy Driven Repository Interoperability (PoDRI) in conjunction with iRODS, and others who will be named shortly.

Use Cases

This section will lead to use case pages which inform the discussion both Fedora directed and use cases which are beyond Fedora's scope.

Glossary

A common nomenclature is needed to facilitate understanding. It's best to use common terms but consistency within this discussion is more important if the term has multiple definition in common use.

Documents

- Draft proposal (pdf) - Initial proposal for HighlevelStorage layer. Discusses motivation, separation of concerns, and possible use cases.
- Aaron's Presentation - Presented at Feb. 2010 ondon meeting. Summarizes proposal, and introduces a possible configuration of internal modules connected by chaining.
- Asger's Presentation - Presented at Feb. 2010 London meeting. Introduces Writable/Readable store interfaces used for plugging in indexing, caching, etc.
- DEV:OR '10 extended abstract - Extended abstract submitted to Open Repositories 2010.
- DEV:OR '10 presentation slides - Slides used for OR'10 Fedora user group presentation

Issues for Discussion

In a nutshell, this proposal aims to remove certain hard-coded storage assumptions in Fedora, and present a storage layer/interface that would allow a place for extensions to Fedora that implement multiplexing, non-blob storage, lock-free updates, cloud storage, etc.

- Interface to DOManager layer
  - Do we present a single HighLevelStorage interface to the DOManager for reads and writes, or do we present Readable and Writable?
- Datastream versioning.
  - There is a proposal to get rid of datastream versions in the model, and present versioning as a storage layer concern (**TODO: get proposal and link to it**)
  - If versioning becomes a concern of the storage layer, how does this affect the proposed interface?
    - Add get(PID pid, String version)? Is this sufficient?
- DigitalObject representation
  - Is the existing DigitalObject interface sufficient? Can it be improved? Should it be replaced?
  - If versioning moves from the object model (datastream versions) to the storage layer, then presumably DigitalObject would have to be updated so that it no longer exposes versions
  - Is there anything that would obviously present a problem to performance or scalability with corner-case objects (lots of datastreams or versions)?
- Return values
  - Operation or transaction ID?
  - A generic map that any module in the storage layer can populate?
- Chaining of modules
  - Should HighLevelStorage modules be assembled into a chain, a tree, both?
  - Tree configuration implies a list Writable modules, all seeing the same input.
  - Chain configuration implies modules that implement HighLevelStorage, but also may have a delegate HighLevelStorage class configured beneath
    - May need to pay attention to InputStreams passed up/down chain

Implementation plan

Since the storage layer in Fedora resides beneath the object management (DOManager) layer in Fedora, adopting HighLevelStorage implies creating an alternate DOManager instance that interacts with HighLevelStorage rather than ILowLevelStorage. Ideally, this alternate DOManager would be a simple drop-in replacement for the existing DefaultDOManager. Initial development of HighlevelStorage could then be be largely independent from the core Fedora code, deployment would be enabled by a simple configuration change. Unfortunately, this is not easily possible today due to unnecessary coupling between certain Fedora components, and an abundance of unrelated functionality in DOManager that can/should exist elsewhere. These issues would need to be addressed in order to create a truly pluggable DOManager.

While HighLevelStorage is not scheduled to be a feature of Fedora 3.4, it may be developed concurrently or slightly after the 3.4 release date. As many of the prerequisites for drop-in replacement of DOManager are general improvements to the Fedora code base that are not storage-specific, there is distinct appeal to incorporating these basic improvements into the core in time for Fedora 3.4. With these prerequisites in place, work on HighLevelStorage may proceed entirely as an add-on/replacement module, hopefully without further changes to the core. Combined with Fedora's enhanced modular architecture, it would potentially allow HighLevelStorage to be distributed as an add-on bundle to Fedora 3.4 for evaluation or testing before it becomes a core feature.

Relevant tracker items

Need to figure out how to link to the issues using the new Jira version!