

Feb 23-24 2010, London Committer Meeting

Jump to:

- [Purpose and Summary](#)
- [Tuesday Notes](#)
 - [Enhanced Content Models](#)
 - [Service Definition/Deployment Improvements](#)
 - [Datastream Methods](#)
 - [OSGi and Spring](#)
- [Wednesday Notes](#)
 - [High Level Storage](#)
 - [Other Storage Topics](#)
 - [Semantic Web and Linked Data](#)
 - [WebDAV \(or not..\)](#)
- [Attendees](#)

Purpose and Summary

The purpose of this meeting was to formulate, discuss, and prioritize the major Fedora development items for 2010, and to start identifying who can help make them happen. It was held in the Franklin-Wilkins Building on the Waterloo Campus of Kings College, London, Feb 23-24.

Based on a poll of [attendees](#), the [agenda](#) focused on the following major topics:

1. **Content Modeling Architecture**
2. **Module Architecture**
3. **Storage**
4. **Interfaces**





Tuesday Notes

Enhanced Content Models



Asger presented an overview of the [Enhanced Content Model](#) work, and we discussed which parts made sense to fold into the core Fedora distribution. The discussion focused primarily on the extension mechanism and schema + relationship validation.

- [Presentation by Asger](#)



The group agreed that the following would be good to fold into the core distribution:

-  An extension mechanism for DS_COMPOSITE_MODEL, inspired by ECM's
-  Use of this extension mechanism to:
 - Include XSD references
 - Include OWL Lite references
-  The ability to validate XSD and OWL on-demand (after ingest)
-  The ability to clone objects via Fedora's APIs

The following need further discussion:

-  Ability to validate XSD and OWL as prereq. to "Active" state transition
-  Kai indicated interest in defining DSL(s) for validation





To drive this work forward, we identified:

-  Lead: Asger
-  Contrib: Kai

Service Definition/Deployment Improvements

Ben led a discussion of some of the [outstanding issues with SDefs and SDepts](#), suggesting several improvements in addition to what has been documented in JIRA.

The group agreed that the following would be good to have in future versions of Fedora:

-  The ability to support multiple bindings in service deployment WSDL. **Action:** Create a JIRA issue for this
-  Support for POST/PUT/DELETE object methods (front-end)
-  Support for POST/PUT/DELETE method bindings (back-end)
-  Support for SOAP method bindings (back-end)

We did not have time to discuss everything noted in the document, but the general consensus was that a rehaul of the way the service deployment code works in Fedora (versus small, incremental code changes) will probably be necessary to get us where we want to go.

To drive this work forward, we identified:

- 🧑 Lead: Ben
- 🧑 Contrib: Asger
- Followup --> [Re-Implementing Service Deployment](#)

Datastream Methods

We had planned on discussing Asger's proposal for [adding datastream methods to Fedora](#), but decided to discuss this later in the interest of time.

OSGi and Spring

Eddie kicked off a discussion on what we've learned with OSGi so far, and Bill and Andrew shared some of their OSGi experience with DuraCloud.

The idea with the first chunk of our Fedora-OSGi work was to a) prove that Fedora can be packaged as a complete OSGi bundle, and b) begin OSGi-fying some of the key pieces that Fedora uses under the hood. To that end, Eddie experimented and had some success with [building Fedora as an OSGi bundle](#), and started doing the same for Mulgara. Chris successfully changed Akubra's plugins to be OSGi bundles and has gained experience in making existing artifacts OSGi-friendly.

Bill and Andrew are actively using OSGi for the "service" portion of DuraCloud, but noted that there is a significant learning curve and a new set of dependency issues to be concerned with, similar to the "growing pains" we experienced in the transition to Maven.

Those of us with some OSGi experience agreed that OSGi still represents a promising direction for Fedora, but there are still a lot of hurdles to getting there. Like maven, these hurdles will become shorter with time, as Java-OSGi adoption increases and the community around it grows.

During this discussion, we noted again that moving toward OSGi is not incompatible with using Spring for dependency injection. The latter seems to be a more tractable goal for the time being, with more immediate payoff. We agreed to:

- ✅ Document best practices for being "OSGi Friendly"
- ✅ In the short-term, move to Spring for dependency injection (Fedora modules = spring beans)
- ✅ Keep the long-term goal of having a Fedora OSGi bundle that can be used by other apps

To drive this work forward, we identified:

- Lead: 🧑 Chris
- Contrib: 🧑 Dan, 🧑 Andrew, 🧑 Asger
- Followup --> [Module Architecture Development](#)

Wednesday Notes

High Level Storage

Aaron presented his [proposal for a high level storage interface for Fedora](#), describing the motivation and use cases that it enables.

- [Aaron's presentation](#)

This proposal was generally well-received.

During the discussion, Aaron identified the need to have Fedora's internal datastream locations (stored within the FOXML, for pointing to managed content) be specific. That is, rather than "pid+dsid+dsVersionId", they should be an internal location that means something to the underlying storage. **Action:** Validate this with Aaron and add as a JIRA issue.

We also touched on what it would mean to make this interface compatible with asynchronous read/write use cases. For reads, if Fedora cannot find the content immediately, a 503 http response and a retry-after header should be sent. It is unclear exactly what should be happening at this level to support such use cases, but one possibility that was mentioned as to have the operations return a Map of name-value pairs as a possible extensibility point.

The ideas around high level storage are still forming, and are inspiring some ideas about caching and sending updates to modules in Fedora in pluggable ways. Asger presented a follow-on proposal:

- [Asger's presentation](#)

This proposal splits the high level storage interface into a read and write side. There was some discussion about how to "stack" these interfaces. One possibility that came up was having the wholistic Read+Write interface (Aaron's interface) be a singleton within the repository, through which all storage calls go, and some implementation of that might then send reads/writes to delegates beneath, which are read-only or write-only (Asger's interfaces).

One of the major questions that Asger's presentation provoked was whether versioning was still important to do at the datastream level, or whether it can be done at the object level. We had a follow-on discussion about this. The idea presented was: What if Fedora no longer held information about old versions in the DigitalObject class and in the stored FOXML? In other words, these would be designed to work only with the current version of the components of the object. If a datastream changed, a new version of the entire object would be made (and object-level version number would be incremented), and older versions of the datastreams would be retained only if storage was configured to do so. While discussing this, one concern we landed on was that there would no longer be a manifest pointing to all versions of everything stored within an object. We cut this portion of the discussion short in the interest of time. **Action:** Continue this discussion with others in the Fedora community (wiki page, mailing list, etc)

To drive the high level storage work forward, we identified:

- Lead: 😊 Aaron
- Contrib: 👍 Asger, 👍 Dan, 👍 Chris
- Possible Contrib: 🤔 Lee Namba (re:Caching), 🤔 Kai, Others at FIZ (re:Versioning)
- **Followup** --> [High-level storage layer](#)

Other Storage Topics

The original agenda had several "Hot Topics" defined for storage:

- Hierarchical Storage Support
- Multiplexing
- In-Place Ingest
- Large Datastreams
- Replication and Messaging

While there was not enough time to discuss these topics by themselves, we touched on some of them as part of the high-level storage topic (hierarchical storage support, multiplexing) and the WebDAV topic (in-place ingest).

Semantic Web and Linked Data

Steve presented his latest thoughts on improving SemWeb and Linked Data support in Fedora.

- [Steve's presentation](#)
- [Wiki page](#)

We did not have much time to discuss with the group, but the following ideas seemed well recieved and worth persuing immediately:

- ✅ Deprecate "Lite" APIs
- ✅ HTTP URIs for RI queries: new parameter: scope = local|global, where local scope is vs. "info:" uris, and global scope is vs. "http:" uris (translated on the way in and out)

Steve also touched on the following ideas, which need some more experimentation/specification:

- 🤔 Storing quads vs triples: Each object's triples are in a graph. Performance? Compatibility w/multiple triplestores?
- 🤔 Graph hierarchy: Does it make sense to have addressable graphs for each datastream as well? Or just per-object. Advantages /disadvantages.
- 🤔 Declarative specification of which datastreams/triples to index.
 - Could be driven on a per-cmodel basis.
 - For base triples, system object methods might specify which triples to "generate".
- 🤔 REST API for PUT/POST/DELETE of triples on a per-object basis. Interestingly, this could allow for RDF management without the requirement that an index is present. We discussed at which endpoint this logically fits: object/datastreams/datastream (straightforward to figure out where triples are stored), or object (can be figured out in some cases, but hard in other cases)

To drive this work forward, we identified:

- Lead: 😊 Steve
- Contrib: 👍 Asger, 👍 Ben
- Possible Contrib: 🤔 Paul Gearon


WebDAV (or not..)

Kai led a discussion on having a [WebDAV interface for Fedora](#). One of the major motivating use cases was to have an easy way to ingest content into the repository.

We talked about the fact that WebDAV might not actually fit the bill here because OS-level clients (what most people would presumably be using for the "easy drag and drop case") are generally not that great. OSX likes to do unnecessary locking for all writes. Windows' client has been poorly supported for some time as well.



During the discussion, an alternative idea came up: If we want an easy drag-and-drop interface for the repository, how about ftp:? OS clients' support for FTP is generally very good, and clients and servers already exist for FTP. So the real problem for us is simplified to: How do you turn a directory full of files into Fedora objects?

This elicited a positive response from the group, especially for the simple "Drag and Drop" ingest scenario.

-  Develop a "drop box" module for the Fedora server, which scans a directory periodically and when new items come in, wraps them and ingests them as Fedora objects. This would be a prototype initially, and wouldn't have to be developed as part of the core.

Another idea that came up was that of a "live box", where content doesn't actually get moved out of the directory it's placed into, but is pointed to as a kind of "in-place" ingest. We just scratched the surface of this latter idea.

To drive this work forward, we identified:

- Lead:  Kai
- Contrib:  Dan

Attendees

- Aaron Birkland (Cornell)
- Andrew Woods (DuraSpace)
- Asger Askov Blekinge (State & Univ Lib, Denmark)
- Ben Armintor (Columbia U)
- Bill Branan (DuraSpace)
- Brad McLean (DuraSpace)
- Chris Wilper (DuraSpace)
- Dan Davis (Cornell)
- Edwin Shin (MediaShelf)
- Gert Pedersen (Tech Univ of Denmark)
- Kai Strnad (FIZ Karlsruhe)
- Paul Pound (UPEI)
- Simon Lamb (Hull)
- Stephen Bayliss (Acuity Unlimited)
- Tim Donohue (DuraSpace)
- Thorny Staples (DuraSpace)