Fedora is not Fedora is not Fedora

Formalizing the Fedora API

Motivation

- Why spend time specifying the API? What are the benefits?
- What do we mean when we say “Fedora”?
- What does the answer to the second question tell us about how to answer the first?

Why specify the API?

Durability for content must be supported by durability for the machinery that holds and processes it.

Why specify the API?

A well-specified API allows systems that use Fedora to evolve carefully and without nasty surprises.

Don't we have a specified API?

No.

Don't we have a specified API?

We have extensive human-readable documentation, but not formal specification.

- Documentation describes, but does not prescribe.
- That means expectations, but not guarantees, and that means… trouble!

Don't we have a specified API?

What is the true “specification” of Fedora software?

Whatever a given version of the software actually *does*!

What should we do?

If we want a durable repository API, we should think about how we provide durability to other “kinds of Fedora”.

Fedora themselves tell us a lot about that.

What is “Fedora”?

Three kinds of Fedora:

- Fedora, the information architecture
- The Fedora community
- Fedora Commons software

Fedora, the information architecture

- “Object”, “Datastream”, “Disseminator”
- Has evolved over the years
- Only well-understood by Fedora software

The Fedora community

- A “repository” of tested praxes, human and institutional relationships
- Has also evolved over the years
- Centered on the information architecture and its value of durability

Fedora Commons software

- Built by the community as the premier implementation of the information architecture

Aspects of durability

- For the model
- For the community
- For the software

Durability in the model

Arises from clarifying and publishing ontological claims (content modeling, relationships between resources)

Durability in the community

Arises from sharing ontological claims and practices, and engaging with larger communities

Durability in the software

Arises from well-known practices for good software engineering

- Modularization
- Versioning
- Testing

We want these durabilities for our API.

What are we doing?

- The API specification comprises a core and extension modules, organized in logical packages
- Each module will contain a formal specification and automated test suite, versioned together
- Most modules will also include a formal ontology

What are we doing?

- Test suites will provide testability for any new Fedora implementation, and
- guarantee interoperability

What are we doing?

- All of the APIs assume RDF over HTTP.
- The ontologies are being made available in RDFS/OWL.

The Core

- LDP + the Core ontology
- Does CRUD with the Fedora model for content

Packages: Workflow APIs

- The Core Module
- Transactions
- Versioning
- Locking?

Packages: Administration APIs

- Backup/Restore

Packages: Other guys

- Fixity
- Your contributions

Where we are

https://wiki.duraspace.org/display/FF/API+review+and+discussion

https://wiki.duraspace.org/display/FF/API+Partitioning