Designing a Migration Path - Fedora API Review

Introduction

Since the release of Fedora 4.0, Fedora has provided a robust REST-API based primarily on the Linked Data Platform (LDP) as the pattern for resource management operations (Create, Read, Update, and Delete; or CRUD). Over time, the other services provided by the API (versoning, authorization, fixity, and messaging) have been aligned with other relevant web standards. However, these services and standards were subject to change, and the API was not versioned separately from the underlying implementation, so there was no clear way to distinguish between API changes and software updates without reading detailed release notes. The API specification changes this paradigm such that the services Fedora offers and the standards with which it complies are now formally documented and versioned separately from the underlying application. This provides stability and predictability for clients, while also unlocking the potential to replace the underlying application with a different implementation that satisfies different use cases. This approach may also provide benefits for repository migrations because tools can be written against the API, which will change less often than the software application.

The Fedora API

The Fedora API is primarily an extension of the LDP 1.0 specification, but it also contains the following:

- reconciliation of [LDP] and the version identification and navigation scheme delineated in the Memento specification [RFC7089],
- integration with the Web Access Control proposal [SOLIDWEBAC],
- design for the publication of event notifications, and
- interaction patterns to support binary resource fixity verification.

The primary goal of the API specification is to clearly define the behaviours of Fedora implementations to promote stability and interoperability with client applications. A client should be able to be developed against the specification without knowing anything about the underlying implementation, and that same client should work with a different Fedora API implementation without making any major changes.

The main components of the API specification are as follows:

Resource Management - Linked Data Platform (LDP)

LDP defines interaction patterns for clients and servers on the web using linked data (RDF). LDP defines two primary types of entities: containers (composed of RDF) and binaries (not composed of RDF; essentially files). Fedora supports these entity types, along with some more specific types of containers, and uses these concepts for resource management. Everything a user creates, reads, updates, or deletes in a Fedora repository is either a container or a binary; containers are used to represent intellectual objects and binaries are typically files uploaded from a desktop of server (PDFs, images, videos, etc.). The native response format for requests made against these resources is RDF.

Versioning - Memento

Fedora supports versioning for both containers and binaries; a new version can be created as part of any resource management operation, and previous versions can be restored. These interactions are provided in accordance with the Memento specification. Memento provides a mechanism for navigating previous versions of a web resource using standard HTTP operations on the web. A resource that supports Memento provides an HTTP link to a TimeGate resource that manages all the previous versions (called Mementos) of that resource. A client can issue a request (including a date and time) to the TimeGate, which will respond with a link to a Memento that most closely matches the date and time of the request. Thus, a client can issue a request for a Memento of a Fedora resource and get an appropriate response without knowing anything about the underlying Fedora software implementation.

Authorization - Web Access Control

Fedora provides authorization but not authentication. Authentication, which validates a user's credentials (typically a username and password) and either allows or denies access, is assumed to take place at a layer above the Fedora API. Once a user has been authenticated, Fedora provides authorization support: the user will have different levels of access (read, write, delete) for different resources (or groups of resources) depending on what policies have been set. The authorization policies are based on Web Access Control, a standard for creating and managing policies using RDF. Administrators can control access to repository resources by creating and managing these policies.

Notifications - Activity Streams

Fedora emits notifications after every durable change to a repository resource. Any create, update, or delete operation is considered a durable change. These notifications are messages constructed in accordance with the Activity Streams 2.0 standard. While Fedora doesn't perform any further operations on these notifications after they are sent, the notifications can be used to power external integrations. These integrations are fundamentally asynchronous, and can be used to perform useful background operations such as indexing for search, generating derivatives for binary resources, and triggering exports to other applications and services.

Fixity - HTTP Headers

Fedora supports binary resource fixity in two ways: transmission fixity and persistence fixity. For transmission fixity, a checksum may be included with a binary file when it is uploaded to Fedora. In this case, Fedora will calculate the checksum of the file and reject the upload if there is a mismatch. If the checksums match, Fedora will store the file along with its checksum as an RDF property. For persistence fixity, a client can request the checksum of a binary resource stored in Fedora at any time. This checksum can then be compared against a known value (e.g. the checksum stored as a property on the binary resource).

Impact on Migration

The Fedora API specification can support a path to migration because migration tools can be built against the API. For example, a tool could take exported data from Fedora 3.x, convert it to a format compatible with Fedora 4.x and higher, and import the data via the Fedora API. This has the advantage of providing a stable interface to build tools against; the API will change more slowly than the underlying application, so tools that work against the API will continue to be useful as Fedora undergoes application upgrades over time.

The primary disadvantage to API-based migrations is performance. Transferring large amounts of data over HTTP can be slow, and the REST API performs a number of operations during object creation, particularly for objects with many internal relationships, that can create an ingest bottleneck. There are ways to mitigate these disadvantages, however, and they may not be relevant for institutions with relatively small collections or longer timelines to complete a migration. One mitigation strategy is to convert XML metadata to RDF properties, thus cutting down on the number of individual files that need to be created.

Another disadvantage is the nature of application upgrades over time. The application underlying the Fedora API is likely to change more quickly than the API itself, and the API specification does nothing to mitigate these changes over time. While the API specification does make migrations easier from a tooling perspective, the difficulties of moving large amounts of data will only become greater as the amount of data managed by an institution grows. One possible mitigation strategy would be to leverage Fedora's external content capabilities to store some or all binary content outside Fedora's native filesystem on some other hardware than is unlikely to change very quickly.

Conclusion

The Fedora API specification describes the services Fedora provides in alignment with a set of modern web standards. The specification represents a stable, independently versioned interface on top of the Fedora software application that offers stability for clients and flexibility for alternate implementations that service different use cases. In terms of migrations, tools can be built against the Fedora API to support data imports from previous versions of Fedora, and going forward the API specification will make it much easier to support API-based migrations in future versions of Fedora. While there are disadvantages in terms of performance and future application changes, the API offers a clear benefit to anyone pursuing a migration to a current version of Fedora.