# Rationale

In 1998 when the original paper detailing the Fedora architecture was published there were many more open questions around the boundaries of responsibility of a Fedora repository and how the services the architecture could offer fit into the nascent repository application stack. In the subsequent 18 years, the maturation of both the software ecosystem surrounding a Fedora repository and the real-world requirements defined by the experience of our community's use cases have clarified Fedora's role in supporting the goals of preservation and access in the context of big data and performance at scale.

A Fedora repository is a digital object repository with Web-facing capabilities for publishing objects and backend facilities for managing and preserving those objects. The resources exposed on the web, must abide by established principles of RESTful HTTP as first-class participants in the read-write Web. Fedora must also enable bit-level preservation of the resources it manages. These access and preservation functions must maintain response performance while scaling the number of objects, the number of bytes, and the size of individual bitstreams, as well as the number of client requests.

It has become clear that attempting to address all of the requirements from the variety of Fedora installations results in an implementation that only partially satisfies any given institution, at extra cost to all. This has led to the organizational decision to return the focus of the Fedora repository to its essential, core capabilities while encouraging decoupled integration with external services. Examples of common external services include validation-on-ingest, bulk ingest/retrieval/edit, resource transformation, derivative generation, etc. This approach embraces the assumption that the Fedora repository is one piece in an institution's larger infrastructure ecosystem.

However, there is still enough diversity of requirements around scale, storage, and performance that the expectation that a single Fedora implementation will satisfy all requirements is optimistic at best. The reference Fedora implementation will target many Fedora use cases, but the option must be available to introduce alternate Fedora implementations without impacting repository client code or integrations. For example, applications that interact with a Fedora repository should not be concerned with the implementation idiosyncrasies required by the profile of the managed resources.

This speaks to the need for a well-specified application programming interface (API) that provides a stable layer of abstraction between Fedora clients and repository instances. In this way, alternate Fedora implementations suited for specific service profiles can all expose the same core of services to repository clients.

This separates two previously-merged concepts: Fedora as an API and Fedora as the implementation of that API. This can be thought of as analogous to how relational databases or RDF triplestores service common SQL and SPARQL-Query requests in the same way despite different backend implementations. A well-defined space of shared responsibility will exist between repository services and the clients that consume those services. Furthermore, the Fedora API specification will be precise enough to allow for automated verification by a technology compatibility kit (TCK); a suite of tests to be run against a prospective implementation. A Fedora TCK would provide a means by which alternate implementations of the Fedora API could be verified as "doing Fedora".

Although Fedora 4 has offered a production-ready REST API since late 2014, the Fedora API specification has yet to be formalized. The API specification is the most fundamental of several specifications for core services of a Fedora repository:

- API
    - Create/Read/Update/Delete (CRUD) for repository resources
    - Fixity checking and reporting
    - Resource versioning
    - Atomic batch operations
- Authorization
- Messaging service provider interface (SPI)

As these services are not specific to the Fedora community, we have the opportunity to take advantage of the efforts made by much larger communities of practice. To the degree that this is possible, the Fedora API specification will reuse existing standards, so that existing standards-based client libraries can be used to interact with the Fedora API. Likewise, on the server-side, existing open source components can be reused in Fedora implementations. The more closely Fedora's API specification aligns with standards, the more reuse is possible, resulting in less burden on the Fedora community for long-term maintenance. Additionally, the alignment with standards opens the door to greater interoperability with the broader Web, which will engender greater exposure and use of repository resources.

As noted, the rationale for defining a Fedora API specification includes stability for clients, freedom for server-side implementation, and Web interoperability. Currently, there is a single Fedora implementation. This implementation is built over an open source JCR implementation. Although it is possible to extract a Fedora API specification from a single implementation, having two or more implementations of the API is necessary to prove with greater certainty that the API is indeed a generalization; an abstraction from concrete reality. Extracting an API from a single example creates an API that is biased towards that example. For their respective benefits, it is conceivable that alternate Fedora implementations will emerge based on such technologies as Apache HBase, Apache Cassandra, Apache Marmotta, various triplestores, or Posix filesystems. The advent of alternative implementations of the Fedora API specification will provide confirmation of the utility and generality of the API abstraction.

Driven by a firmer understanding of the Fedora repository's responsibilities, a desire to limit the cost of client software, an interest in broader interoperability, and a need to support diverse use cases, the Fedora API specification represents a significant milestone for the project and the community.