Designing a Migration Path - Oxford Common File Layout Review

Introduction

The Oxford Common File Layout (OCFL) is a community-based effort to develop “an application-independent approach to the storage of digital information in a structured, transparent, and predictable manner. It is designed to promote long-term object management best practices within digital repositories.” The original idea for this effort came from a Fedora Camp hosted at the University of Oxford in September, 2017, so it has roots in the Fedora community even though the OCFL is not dependent on Fedora in any way. However, if Fedora were to persist repository resources as specified by the OCFL, it has the potential to greatly impact Fedora, particularly when it comes to data migrations. One of the primary motivating use cases of the OCFL is to reduce the need for data migrations; both because moving large amounts of data is time-consuming and because such migrations risk data corruption and loss. The OCFL decouples the structure of files on disk from the application(s) that manage the files by defining a file and folder hierarchy to which applications must conform. Thus, the files can be left in place and new software applications can implement the OCFL specification in order to manage the files.

The OCFL Specification

An OCFL Object is “a group of one or more content files and administrative information, that are together identified by a URI. The object may contain a sequence of versions of the files that represent the evolution of the object's contents.” An OCFL object should contain all the data and metadata necessary to represent the intellectual entity and, in the case of disaster, rebuild it completely simply by reading the files on disk. An OCFL object has a specific structure that applications must conform to:

```
[object_root]
    0=ocfl_object_1.0
    inventory.json
    inventory.json.sha512
    v1
        inventory.json
        inventory.json.sha512
        content
            ... content files ...
```

This structure includes a file announcing the object’s conformance with a particular version of the OCFL, a directory for each version of the object, and an inventory of the object’s contents, including checksums.

Impact on Fedora and Migrations

There are two primary ways the OCFL could be implemented in Fedora: 1) as the structure of a native persistence layer beneath the Fedora API or 2) as an exported backup of the repository that could be used to rebuild the repository in the event of a disaster. Both approaches will be described and analysed below.

Implementing the OCFL in the Fedora Persistence Layer

The recently specified Fedora API separates the services Fedora offers from the underlying software that stores and preserves the contents of the repository. This provides an opportunity for the Fedora community to design and develop an alternate software application to the currently used ModeShape application that natively stores and manages the contents of a Fedora repository as OCFL objects. This would allow repository data to remain largely unchanged over time as new versions of Fedora are deployed over the files and folders on disk.

This approach has several advantages: primarily, it eliminates or at least substantially mitigates the need for data migrations. As repositories manage greater and greater amounts of data, migrations become more costly and time consuming. While migrations might still be necessary in the case of storage hardware upgrades or failures, the cycle should be slower compared to standard repository upgrades and migrations. Additionally, it would be possible to completely rebuild the repository from the data on disk in the case of a disaster; currently this is not possible because the metadata is stored in a separate database. Finally, repository contents would be both machine and human readable, which is a high priority for preservation specialists.

The primary disadvantage to this approach is performance and scale. File systems are reliable but slow compared to databases and other modern storage technologies. Reading and writing to disk is a relatively slow operation, and one that is difficult to scale. Fortunately, these issues can be mitigated via a modular application design that uses fast technologies such as databases and caches to power repository access while a slower, asynchronous operation writes the data to disk as OCFL objects in the background. This approach would preserve the benefits of scalable and performant technologies while adding the value of an OCFL-conformant storage layer.

OCFL as a Fedora Export

The second implementation possibility is an export tool that would transform natively stored Fedora data into OCFL objects and export the contents to an external location. A manual or automatic operation could execute the export at appropriate times and the exported files could be used to update an existing export such that it exactly mirrors the contents of the live repository.

This approach has all the same advantages around rebuilding the repository from files on disk and providing machine and human readability, and it would be easier to implement because it doesn't require any changes to the native repository storage layer. An Import/Export Utility also already exists, and this tool could be modified to support OCFL export.

Unfortunately, this approach would have no impact on migrations. The native repository storage layer would remain unchanged, so the data would still need to be migrated in the case of a major repository upgrade.

Conclusion

The Oxford Common File Layout represents a great opportunity for the open source preservation community to standardize on an shared approach to structural file storage. Fedora could benefit from this initiative in several ways, most notably repository rebuild capabilities and self-describing, human readable persistence. In terms of migrations, the greatest benefit can be achieved by implementing an OCFL-compliant application under the Fedora API. While this approach has several design challenges to overcome, and would take significant community resource investment, it has the potential to make future repository migrations far less onerous.