ORA4 objects in OCFL

Product description

An overview of how ORA4 objects will be stored in an OCFL-conformant file system. Alongside this document, the reader should have access to the ORA Data Model METS mapping document [ORA-METS], and the sample ORA4 OCFL storage root [ORA-OCFL-ROOT]. In addition, this document should be read alongside the OCFL specification [OCFL-SPEC].

About this document

Version: 0.3
Author: Tom Wrobel <thomas.wrobel@bodleian.ox.ac.uk>
Created: 27 March 2019
Last modified: 24 April 2019

Background

As part of the ORA4 Digital Preservation Solution, ORA4 will seek to store a human-readable, stable, digital preservation copy of an object. The most appropriate format for this is in a file system that conforms to the Oxford Common File System specification (OCFL).

OCFL is "an application-independent approach to the storage of digital objects in a structured, transparent, and predictable manner. It is designed to promote long-term access and management of digital objects within digital repositories." [OCFL-SPEC]. The OCFL specification describes the way in which multiple versions of an object are stored, and provides a set of rules which enable an object to be validated. The history of an object, including its current state, is stored in an inventory.json file in the object's root.

OCFL concepts

OCFL has the concept of a 'storage root' directory. This storage root contains the OCFL objects, each in their own directory.

OCFL does not prescribe where object directories are stored, only that any directories between the storage root and an OCFL object are empty. However, it is advised that whatever the logical structure of an OCFL storage root, that the path from the storage root to an OCFL object is intelligible from its OCFL object identifier [OCFL-IMPLEMENTATION-NOTES].

In addition, OCFL makes a series of important requirements: OCFL mandates that once a file is written it is immutable, and that once an object version is created, that this version is immutable.

ORA OCFL storage root and file system structure

ORA OCFL objects will be stored in a directory corresponding to their UUID. For an ORA object with the pid "ora_ abcdef01-abcd-abcd-abcd-abcdef013456", the ORA object directory will be "abcdef01-abcd-abcd-abcd-abcdef013456".

The path to this directory will be a pair-tree directory composed of the first four pairs of bytes in the UUID. For example `ab/cd/ef/01/abcdef01-abcd-abcd-abcd-abcdef0123456/` (see also [ORA-OCFL-ROOT]).

The storage root will contain an ocfl_layout.json file detailing the layout of the file system, as well as a copy of the final version of this document as a text file (README.txt) a copy of the ORA Data Model (ora_data_model.xls), and a copy of the ORA Data Model to METS documentation (ora-data-model-to-mets-2.0.xml [ORA-METS]).

An ORA OCFL storage root containing one ORA4 object (ora_ abcdef01-abcd-abcd-abcd-abcdef013456) would therefore have the following structure:

```
[storage_root]
      ├── 0=ocfl_1.0
      ├── ocfl_1.0.txt
      ├── ocfl_layout.json
      ├── ora_data_model.xls
      ├── ora-data-model-to-mets-2.0.xml
      ├── README.txt
      └── ab
            └── cd
                  └── ef
                        └── 01
                              └── abcdef01-abcd-abcd-abcd-abcdef013456
```

OCFL object structure

Each version of an OCFL object will contain a METS file describing the object, its metadata, the binary files associated with the object, and their metadata. This METS file will conform to the specification laid down in [ORA-METS].

The METS file will be named {UUID}.ora{datamodel-major-version}.mets.xml, e.g. abcdef01-abcd-abcd-abcd-abcdef013456.ora2.mets.xml

The METS file will list file locations relative to the OCFL version's 'logical file path' (see [OCFL SPEC] Terminology): the a METS file will use this as the value of the the Flocat[@xlink:href] property for that file. For example (see also sample METS files in objects in the [ORA-OCFL-ROOT]:

```
    <mets:fileGrp>
        <mets:file ID="FILENAME2" MIMETYPE="mimetype">
            <mets:FLocat xmlns:xlink="http://www.w3.org/1999/xlink"
LOCTYPE="URL" xlink:href="file:///test-file-3.txt"/>
        </mets:file>
    </mets:fileGrp>
```

A simple ORA OCFL object will therefore have the following structure:

```
abcdef01-abcd-abcd-abcd-abcdef013456
      ├── 0=ocfl_object_1.0
      ├── inventory.json
      ├── inventory.json.sha512
      └── v1
            ├── inventory.json
            ├── inventory.json.sha512
            └── content
                  ├── abcdef01-abcd-abcd-abcd-abcdef013456.ora2.mets.xml
                  └── test-file.txt
```

A multiple versioned ORA OCFL object will a version directory for each version. Each version MUST contain a METS file describing that version. In the following example:

1. An initial object was created with one file, *test-file.txt*
2. Two additional files were subsequently deposited, *test-file-2.txt* and *test-file-3.txt*
3. One of the files was found to be an incorrect file, and was removed from the object (the contents of an object in any given version are managed in the object's inventory.json file, see the [OCFL-SPEC].

```
abcdef01-abcd-abcd-abcd-abcdef013456
      ├── 0=ocfl_object_1.0
      ├── inventory.json
      ├── inventory.json.sha512
      └── v1
            ├── inventory.json
            ├── inventory.json.sha512
            └── content
                  ├── abcdef01-abcd-abcd-abcd-abcdef013456.ora2.mets.xml
                  └── test-file.txt
      └── v2
            ├── inventory.json
            ├── inventory.json.sha512
            └── content
                  ├── abcdef01-abcd-abcd-abcd-abcdef013456.ora2.mets.xml
                  ├── test-file-2.txt
                  └── test-file-3.txt
      └── v3
            ├── inventory.json
            ├── inventory.json.sha512
            └── content
                  └── abcdef01-abcd-abcd-abcd-abcdef013456.ora2.mets.xml
```

Purging a file

In order to purge a file, it will be necessary to create a new object as if that file had never existed. This new object will replace the existing one, see [OCFL-IMPLEMENTATION-NOTES] File Purging.

The procedure will be as follows:

1. Generate a ‘tombstone’ json file with the title '{sha512}.purged.json', where the sha512 is the hash of the purged file. The tombstone json should include the following keys
    a. date: the date/time of the purge
    b. message: the reason for the purge
    c. user: an object with name and address keys, identifying the relevant user
    d. purged_file_hash: the sha512 of the purged file
    e. purged_file_hash_algorithm: fixed value (for ORA) ‘sha512’
    f. inventory_hash_at_purge: hash of the inventory.json file at the time of the purge, to enable accurate comparison with any version of the object where the file was not purged.
2. Identify the version directory from which the file must be removed
3. Remove the purged file from the content directory in that version
4. Place the tombstone file into the root the content directory for that version
5. Edit the inventory.json file for that version so that it is identical to the original except that the file deposited at that point was the marker file, and not the file that has been purged (i.e. dates, message, user, and all other files remain the same). Recreate the sha512 hash for that inventory.
6. Edit the inventory.json for all subsequent versions as in step 5
7. Create a final version, at the time of the purge, with no content directory and which removes the tombstone file from the state for the head version

Example:

```
abcdef01-abcd-abcd-abcd-abcdef013456
      ├── 0=ocfl_object_1.0
      ├── inventory.json
```

```
├── inventory.json.sha512
└── v1
    ├── inventory.json
    ├── inventory.json.sha512
    └── content
        ├── abcdef01-abcd-abcd-abcd-abcdef013456.ora2.mets.xml
        └── test-file.txt
└── v2
    ├── inventory.json
    ├── inventory.json.sha512
    └── content
        ├── abcdef01-abcd-abcd-abcd-abcdef013456.ora2.mets.xml
        ├── test-file-2.txt
        └── test-file-3.txt
└── v3
    ├── inventory.json
    ├── inventory.json.sha512
    └── content
        └── abcdef01-abcd-abcd-abcd-abcdef013456.ora2.mets.xml
```

Becomes:

```
abcdef01-abcd-abcd-abcd-abcdef013456
├── 0=ocfl_object_1.0
├── inventory.json
├── inventory.json.sha512
└── v1
    ├── inventory.json
    ├── inventory.json.sha512
    └── content
        ├── abcdef01-abcd-abcd-abcd-abcdef013456.ora2.mets.xml
        └──
        62e42efc6851e8cad1cd49e0dfc56e6e622e4616d2c8ee6065641b05a
        068cedcf72241242705bbd4e9866b4f34d785bfc3d9171c02111e3fa4
        7b759fd1bbb5d4.purged.json
└── v2
    ├── inventory.json
    ├── inventory.json.sha512
    └── content
        ├── abcdef01-abcd-abcd-abcd-abcdef013456.ora2.mets.xml
        ├── test-file-2.txt
        └── test-file-3.txt
└── v3
    ├── inventory.json
    ├── inventory.json.sha512
    └── content
        └── abcdef01-abcd-abcd-abcd-abcdef013456.ora2.mets.xml
└── v4
    ├── inventory.json
    └── inventory.json.sha512
```

This approach means that:

- We would know exactly which file was purged: the sha512 identifies the file while also not leaking ANY metadata (including filename)
- We would know when the file was purged (and, if relevant, who was responsible and the reason for the purge)

- The object history would otherwise be unchanged, and consistent with the original (dates could be maintained)
- We could track multiple purges on the same object while being able to audit the object's history
- We could see the original object's history (the original inventory.json data is preserved with the exception of the purged file)
- All versions of the object would be OCFL conformant (once these actions have been completed)
- A copy of the object from which the file had not been purged could be confidently compared with the object with a purged file (the inventory.json hash in the tombstone file would enable a secure link between the two objects)

There is a sample object in the sample objects suite with a purged file: 684f4a8a-1844-4f76-9b06-29816782c43bc.

Sample objects

A suite of sample objects will be maintained in the ORA Data Model github project [ORA-OCFL ROOT].

References

[OCFL-IMPLEMENTATION-NOTES]
Hankinson, A. et al. (2019), *Implementation notes*. URL: https://ocfl.io/0.2/implementation-notes

[OCFL-SPEC]
Hankinson, A. et al. (2019), *Oxford Common File Layout Specification 0.2 - Candidate Recommendation, 20 March 2019*. URL: https://ocfl.io/0.2/spec/

[ORA-METS]
Wrobel, T. (2019), *The ORA Data Model in METS*, URL: https://github.com/tomwrobel/ora_data_model/blob/master/ora_datamodel_to_mets-2.0.xml

[ORA-OCFL-ROOT]
URL: https://github.com/tomwrobel/ora_data_model/OCFL/storage_root