

HEG-796-22-040

AIPs et OCFL

Jan Krause-Bilvin

2022-05-11

## Sessions précédentes

- 
- Linked Data Platform (LDP):
    - Ressources (ldp:Ressource) de type RDF et non-RDF
    - Conteneurs (ldp:Container), peuvent être emboîtés
  - Les conteneurs LDP permettent de délimiter les ressources représentant des objets
    - Combinaison d'ontologies, p.ex. RiC-O, PREMIS, SKOS.
  - PREMIS permet de modéliser et organiser la préservation
  - SHACL permet de définir des “shapes”
    - comment les ontologies sont utilisées et combinées
    - de valider la conformité à la définition
- 

## Cette session

- Archival Information Packages (AIP) : paquets d'information archivistiques (OAIS).
  - Oxford Common File Layout (OCFL) : une spécification pour le stockage des paquets d'information.
  - Lien entre OCFL et LDP.
- 

## AIPs

---

Rappel - schéma d'un système de préservation OAIS :

---

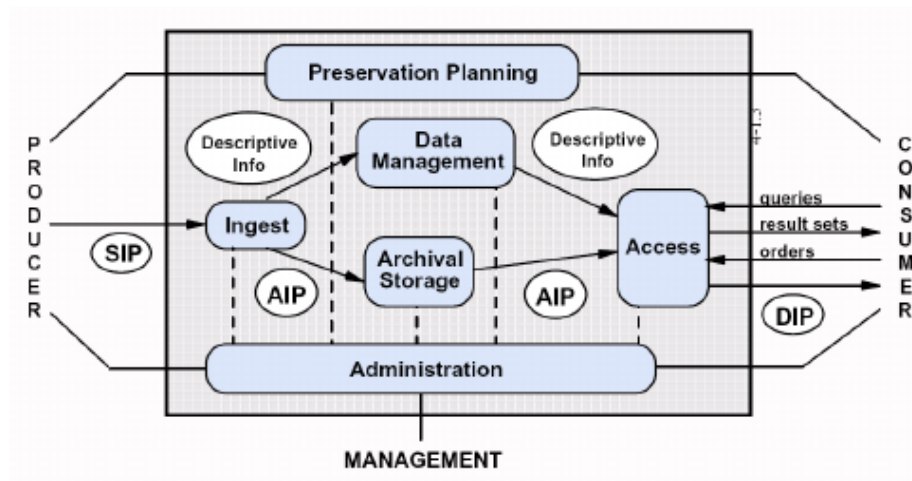


FIGURE 1 – OAIS schema

- Les AIPs sont autonomes : ils contiennent toutes les données et métadonnées d’une “unité archivistique”.
- Au coeur du module de stockage OAIS, ils sont structurés de façon à ce que celui-ci puisse remplir ses fonctions.

---



---

### Cardinalité ContainerLDP - AIP

---

En théorie, cette cardinalité peut être de  $m, n$ .

1. un objet (container LDP) peut être archivé dans  $m$  AIP
2. un AIP peut archiver  $n$  objets (containers LDP)

Nous allons examiner l’option 2.

---

Dans Fedora Commons, par défaut, chaque container LDP est stocké comme un seul AIP.

Mais, il est possible d’attribuer la propriété “archival unit” à un container LDP.

Dans ce cas, ce container et tout ses enfants (définis par *ldp:contains*) seront “physiquement” stockés dans le même AIP.

---

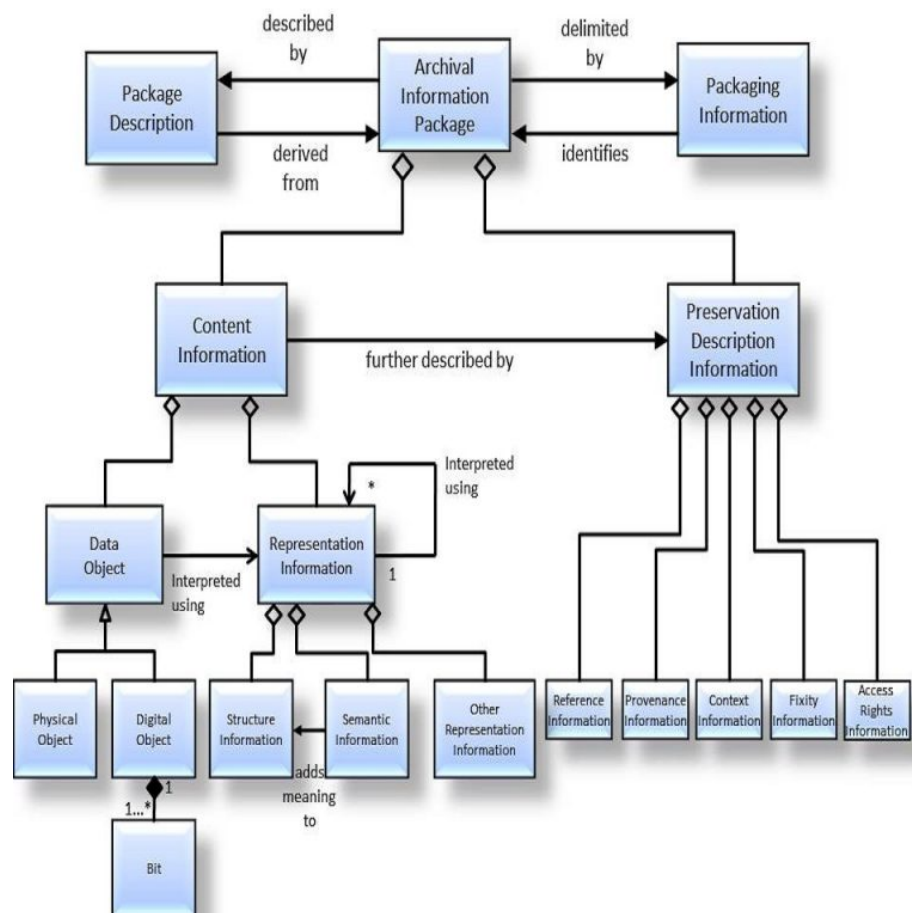


FIGURE 2 – OAIS schema

## Créer un archival unit dans l'interface de Fedora

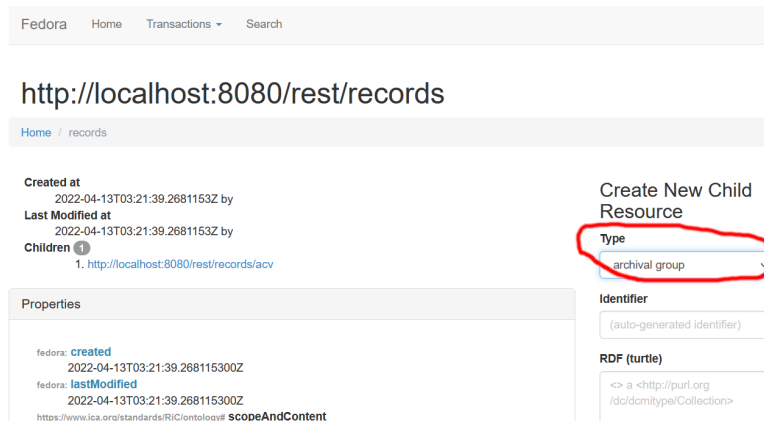


FIGURE 3 – Fedora Commons: new archival group

## Créer un “archival unit” via l'API

```
import requests
url = 'http://localhost:8080/rest/records/acv/D9999'
headers = {"Content-Type": "text/turtle",
           "Link": '<http://fedora.info/definitions/v4/repository#ArchivalGroup>;rel="type"'}
auth = ('fedoraAdmin', 'fedoraAdmin')
data = """ <> <rico:title> 'Ceci est le titre'.
          <> <rico:scopeAndContent> 'Voilà la description'.
          """

r = requests.put(url, auth=auth, data=data.encode('utf-8'), headers=headers)
print( 'Status:', r.status_code )
print( r.text )
```

Noter la partie “link” dans les headers.

## OCFL

Oxford Common File Layout est une spécification. Selon [ocfl.io](http://ocfl.io) :

This Oxford Common File Layout (OCFL) specification describes 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.

---

Toujours selon [ocfl.io](https://ocfl.io), ses bénéfices :

- ***Completeness***, so that a repository can be rebuilt from the files it stores
  - ***Parsability***, both by humans and machines, to ensure content can be understood in the absence of original software
  - ***Robustness*** against errors, corruption, and migration between storage technologies
  - ***Versioning***, so repositories can make changes to objects allowing their history to persist
  - ***Storage diversity***, to ensure content can be stored on diverse storage infrastructures including conventional filesystems and cloud object stores
- 

En pratique, OCFL définit:

- la hiérarchie de stockage
    - i.e. l'organisation des paquets (objets OCFL) sur le media
  - le format des paquets (objets OCFL)
    - i.e. la structure des paquets eux-mêmes
- 

## La hiérarchie de stockage OCFL

Elle doit être déterministe. Dans le cas de Fedora Commons, la règle par défaut pour calculer le chemin des paquets est la suivante:

```
hash := sha256( fedoraId )
chemin := hash[0:3]/hash[3:6]/hash[6:9]/hash
```

Exemple:

```
fedoraId = records/acv/dossiers/D1
chemin = 536/2a8/fe0/5362a8fe0af7fd17596d076f943f179...
```

---

## La structure des paquets OCFL

- un répertoire par version v1, v2, v3, ...
- dans chaque répertoire de version, il y a:
  - un fichier d'inventaire, *inventory.json*, comprenant:
    - maifeste: liste de tous les fichiers avec leur chemin en regard avec leur checksum
    - pour chaque version:

- liste des fichiers composant la version avec référence au manifeste via le checksum
  - un répertoire *content* avec les fichiers ajoutés ou modifiés dans cette version
- 

Exemple 1:

Voir [Exemple\\_OCFL\\_inventory.json](#) et [Exemple\\_OCFL\\_structure.png](#) :

- v1 : création du paquet avec les fichiers *RiC.ttl* et *IMG\_20210228\_092707.jpg*
  - v2 : ajout du fichier *cal.txt* et renommage de *IMG\_20210228\_092707\_renomme.jpg*
  - v3 : suppression de *IMG\_20210228\_092707\_renomme.jpg*
  - v4 : modification de *cal.txt*
- 

Exemple 2:

```
[object root]
├── 0=ocfl_object_1.0
├── inventory.json
├── inventory.json.sha512
├── v1
│   ├── inventory.json
│   ├── inventory.json.sha512
│   └── content
│       ├── empty.txt
│       ├── foo
│       │   └── bar.xml
│       └── image.tiff
├── v2
│   ├── inventory.json
│   ├── inventory.json.sha512
│   └── content
│       └── foo
│           └── bar.xml
└── .
```

FIGURE 4 – OCFL package: nested content

---

Démo :

Exemple d'un dossier dans Fedora Commons.

---