

HEG-796-22-030

PREMIS et SHACL

Jan Krause-Bilvin

2022-05-02

## Thème de cette session

- 
- Ontologie de préservation (**PREMIS**)
  - Schémas de graphes RDF (**SHACL**)
- 

## Cours précédents

- 
- Linked Data Platform (LDP):
    - Ressources (ldp:Ressource) de type RDF et non-RDF
    - Conteneurs (ldp:Container), peuvent être emboîtés.
    - Manipulation via verbes HTTP:
      - GET: lire
      - POST / PUT : créer / mettre à jour
      - DELETE : effacer
  - Les conteneurs LDP permettent de délimiter les ressources représentant des objets (métier, archivistiques).
    - Combinaison d'ontologies, p.ex. RiC-O, SKOS et autres
    - Fedora Commons accepte tout turtle valide (ressources RDF)
- 

## PREMIS

---

PREservation Metadata : Implementation Strategies (PREMIS) permet de représenter:

- les objets (p.ex. records),
- les événements (de préservation),
- les agents (personnes, logiciels) impliqués dans ces événements,
- les droits.

en RDF.

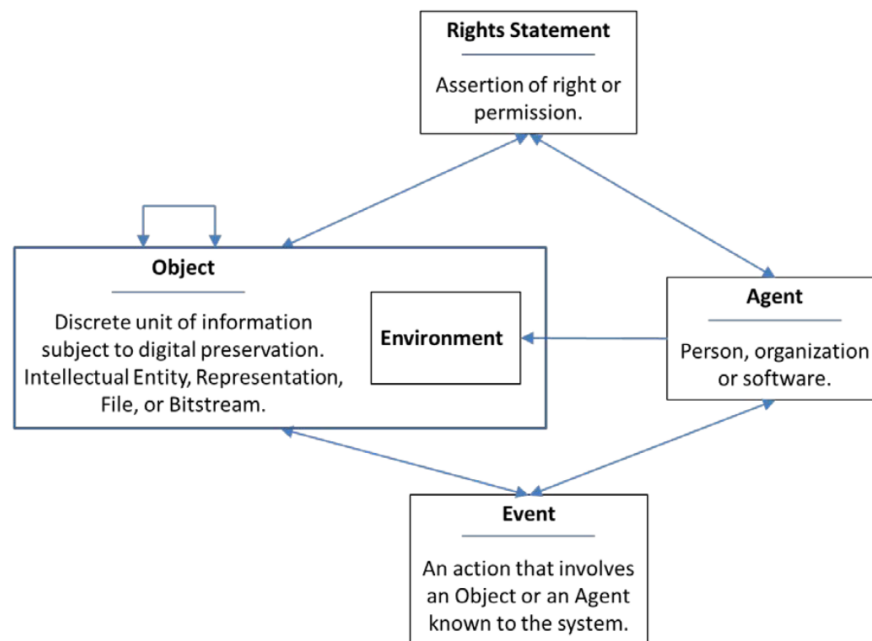


FIGURE 1 – PREMIS overview

Une représentation fine des objets numériques est disponible.

Exemple / demonstration : **docuteam Packer** (logiciel libre)

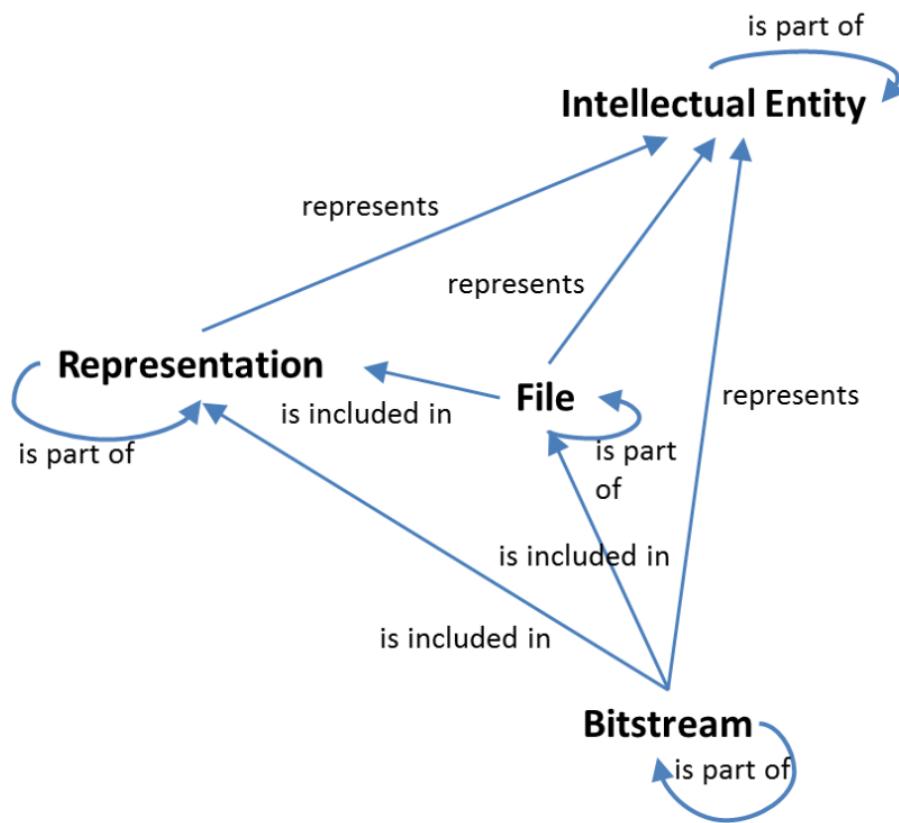
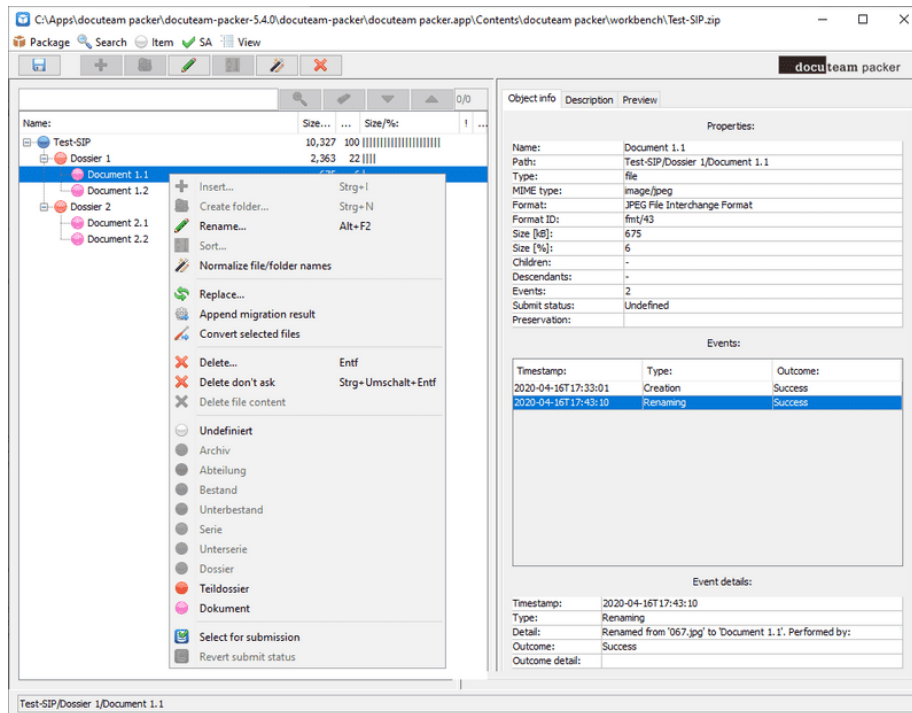


FIGURE 2 – PREMIS overview



Exemples de concepts PREMIS:

- premis:hasMessageDigest (checksum)
- premis:signature (signature numérique)
- premis:hasSize (taille en octets)
- premis:originalName (nom original)
- premis:rights (droits)
- premis:formatRegistry (format de fichier)
- premis:creatingApplication (application)
- premis:environmentDesignation (système)
- premis:inhibitors (inhibiteurs t.q. DRM ou chiffrement, cf. **DLCM**)
- premis:hasCompositionLevel (composition)

PREMIS peut être combiné aux ontologies descriptives pour assurer la préservation numérique. Par exemple:

PREMIS	RiC-O
-	RecordSet RiC-E03
Intellectual entity	Record RiC-E04
Representation	Instantiation RiC-E06

PREMIS	RiC-O
File	-
Datastream	-

Mais comment faire en pratique?

## SHACL (Core)

### SHape and Constraint Language (W3C)

- Il s'agit d'un langage de validation de graphe RDF.
- Les graphes sont composés de noeuds (ensembles de triplets).
- Validation porte sur la structure et le contenu des noeuds.

Exemple:

```
ex:Alice
  a ex:Person ;
  ex:ssn "987-65-432A" .
```

```
ex:Bob
  a ex:Person ;
  ex:ssn "987-65-432B" ;
  ex:birthDate "1971-07-07"^^xsd:date ;
```

SHACL

```
ex:PersonShape
  a sh:NodeShape ;
  sh:targetClass ex:Person ;      # toutes les pesonnes
  sh:property [                  # _:b1
    sh:path ex:ssn ;             # contraintes ex:ssn
    sh:maxCount 1 ;
    sh:minCount 1 ;
    sh:datatype xsd:string ;
  ] ;
  sh:property [                  # _:b2
    sh:path ex:birthDate ;
    sh:maxCount 1 ;
    sh:datatype xsd:date ;
  ] ;
```

---

Nous allons nous focaliser sur

- Nombre d’occurences:
    - *sh:minCount* , *sh:maxCount*
  - Type de noeud:
    - *sh:NodeKind sh:IRI* , *sh:NodeKind sh:BlankNode*
  - Type de donnés:
    - *sh:datatype xsd:date*, *sh:datatype xsd:string*
  - Format
    - *sh:maxLength 50*
    - *sh:pattern “^\d{3}\.\d{4}\.\d{4}\.\d{2}\$”*
- 

### Mode de validation fermé

Pour un noeud donné, le mode fermé (*sh:closed true*), requiert que chaque triplet satisfasse au moins une condition énoncée.

Par défaut, le mode est ouvert. En d’autres termes, les triplets non concernés pas les conditions sont ignorés.

---

### Démonstration

---

Validateurs en ligne:

- [SHACL Play](#)
- [SHACL.js](#)

Ou avec un module python (fournit avec le TP3).

Syntaxe pour valider le RDF *dossier.ttl* en utilisant le SHACL *shacl.ttl*:

```
python shacl.py dossier.ttl shacl.ttl
```

---