PKL : CODES & FICHIERS

compléter les codes

LES FICHIERS

Fichier	Description
`PKL.min.xml`	PKL minimale
`PKL.xml`	PKL complète
`PKL.template-xmlsec.sh`	Script simple avec xmlsec1 et utilisant le template PKL ci-dessous à refaire
`PKL.template-xmlsec.xml`	PKL template sans la signature et pouvant être utilisé pour créer Digest et Signature
`PKL-ST429-8.xsd`	Fichier de définition XML pour la PKL
`PKL-getAssetsInfos.py`	Script Python permettant de récupérer les informations des assets

VALIDATION XML

Schéma XML de référence pour la PKL : PKL-ST429-8.xsd

```
$ xmllint --noout --schema PKL-ST429-8.xsd PKL.xml
PKL.xml validates
```

Si vous voyez `validates`, c'est que votre PKL est parfaitement syntaxiquement valide (à ne confondre avec sa validation cryptographique)

Voici un exemple d'une mauvaise PKL et du type de message d'erreur xmllint :

```
$ xmllint --noout --schema PKL-ST429-8.xsd bad-PKL.xml
bad-PKL.xml:13: element BadTag: Schemas validity error :
    Element '{http://www.smpte-ra.org/schemas/429-8/2007/PKL}BadTag':
       This element is not expected. Expected is
          ( {http://www.smpte-ra.org/schemas/429-8/2007/PKL}OriginalFileName ).
bad-PKL.xml fails to validate
```

Notez que dans l'entête du fichier PKL-ST429-8.xsd, celui-ci intègre deux imports de fichiers XSD :

```
<xs:import
     namespace="http://www.w3.org/2000/09/xmldsig#"
     schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-schema.xsd"/>
<xs:import
     namespace="http://www.w3.org/XML/1998/namespace"
     schemaLocation="http://www.w3.org/2001/03/xml.xsd"/>
```

Ces deux imports sont utilisés pour valider le reste du schéma XML car PKL-ST429-8.xsd ne va valider que les parties spécifiques à la PKL, le reste étant normé par du XML de base et du XML Signature, il faudra donc les fichiers XSD normés.

Les outils tels que xmllint vont aller importer automatiquement ces fichiers pour compléter leur analyseur. Cependant, si vous n'avez pas Internet, il vous suffit de télécharger à part et de déposer à la racine les deux fichiers (xml.xsd + xmldsig-core-schema.xsd), puis de modifier le schéma de PKL-ST429-8.xsd pour pointer vers les fichiers locaux. Exemple :

```
<xs:import
     namespace="http://www.w3.org/2000/09/xmldsig#"
     schemaLocation="xmldsig-core-schema.xsd"/>
<xs:import
     namespace="http://www.w3.org/XML/1998/namespace"
     schemaLocation="xml.xsd"/>
```

RÉCUPÉRATION DE L'IDENTIFIANT (ID) D'UNE PKL

AVEC XMLLINT :

```
$ xmllint --xpath '//*[local-name()="PackingList"]/*[local-name()="Id"]/text()' PKL.xml
urn:uuid:ce5c22c8-a640-428c-9004-2107e1f3c94e
```

AVEC DU SHELL - VERSION COURTE :

```
$ grep -oE '<Id>urn:uuid:[A-Za-z0-9-]+</Id>' PKL.xml | head -n1 | awk -F'<|>' '{ print $3 }'
urn:uuid:ce5c22c8-a640-428c-9004-2107e1f3c94e
```

AVEC DU SHELL - VERSION LONGUE :

```
$ grep -oE '<Id>urn:uuid:[A-Za-z0-9]{8}-[A-Za-z0-9]{4}-[A-Za-z0-9]{4}-[A-Za-z0-9]{4}-[A-Za-z0-9]{12}</Id>' PKL.xml | h
urn:uuid:ce5c22c8-a640-428c-9004-2107e1f3c94e
```

AVEC PYTHON :

```
>>> from lxml import etree
>>> with open("PKL.xml", "rb") as xml:
    tree = etree.fromstring(
        text = xml.read()
    )
    tree.xpath("/*[local-name()='PackingList']/*[local-name()='Id']/text()")
[urn:uuid:ce5c22c8-a640-428c-9004-2107e1f3c94e]
```

RÉCUPÉRATION DES INFORMATIONS DES ASSETS

AVEC XMLLINT :

Depuis l'élement **AssetList** :

```
$ xmllint --xpath '//*[local-name()="AssetList"]//*[local-name()="Id"]/text()' PKL.xml
urn:uuid:3bd3d849-117b-46b0-bc45-3d3228c987c6
urn:uuid:3433a00f-4bc8-4c16-b33c-b0b0d65711af
urn:uuid:4aa03fde-da81-4451-baaa-4d85bf4773d0
```

Depuis les élements **Asset** :

```
$ xmllint --xpath '//*[local-name()="Asset"]//*[local-name()="Id"]/text()' PKL.xml
urn:uuid:3bd3d849-117b-46b0-bc45-3d3228c987c6
urn:uuid:3433a00f-4bc8-4c16-b33c-b0b0d65711af
urn:uuid:4aa03fde-da81-4451-baaa-4d85bf4773d0
```

Depuis la racine :

```
$ xmllint --xpath '//*[local-name()="Hash"]/text()' PKL.xml
hnBSgENJXOvI6bfpat7GA1VImss=
ACd4Aky39E608RNnVfAOisPICZ4=
YOdZjnEy0JynObowRG9FLXx8tD4=
```

Attention: dans ce dernier exemple, avec **Id**, vous récupérerez tous les **Id** du fichier XML.

AVEC PYTHON :

Vous retrouverez le code ci-dessous en téléchargement ici :

```
from lxml import etree

with open("PKL.xml", "rb") as xml:
    tree = etree.fromstring(
        text = xml.read()
    )

    # Method #1
    for element in ["Id", "AnnotationText", "Hash", "Size", "Type", "OriginalFileName"]:
        values = tree.xpath("*/*[local-name()='Asset']/*[local-name()='%s']/text()" % element)
        print(f"[assets ] {element:16s}  =  {values}")

    # Method 2
    for index, asset in enumerate(tree.xpath("*/*[local-name()='Asset']"), 1):
        for element in ["Id", "AnnotationText", "Hash", "Size", "Type", "OriginalFileName"]:
            value = asset.xpath(".//*[local-name()='%s']/text()" % element)
            print(f"[asset {index}] {element:16s}  =  {value}")


# == Output ============

# == Method 1 ==

# [assets ] Id               =  ['urn:uuid:3bd3d849-117b-46b0-bc45-3d3228c987c6', 'urn:uuid:3433a00f-4bc8-4c16-b33c-b
# [assets ] AnnotationText   =  ['CPL: DCP-INSIDE-CRYPTE_TST-2D-24_C_FR-XX_51_4K_20220102_SMPTE_OV']
# [assets ] Hash             =  ['hnBSgENJXOvI6bfpat7GA1VImss=', 'ACd4Aky39E608RNnVfAOisPICZ4=', 'YOdZjnEy0JynObowRG9
# [assets ] Size             =  ['989547', '889695', '13102']
# [assets ] Type             =  ['application/mxf', 'application/mxf', 'text/xml']
# [assets ] OriginalFileName  =  ['jp2k_3bd3d849-117b-46b0-bc45-3d3228c987c6_video.mxf', 'wav_3433a00f-4bc8-4c16-b33c-

# == Method 2 ==

# [asset 1] Id               =  ['urn:uuid:3bd3d849-117b-46b0-bc45-3d3228c987c6']
# [asset 1] AnnotationText   =  []
# [asset 1] Hash             =  ['hnBSgENJXOvI6bfpat7GA1VImss=']
# [asset 1] Size             =  ['989547']
# [asset 1] Type             =  ['application/mxf']
# [asset 1] OriginalFileName  =  ['jp2k_3bd3d849-117b-46b0-bc45-3d3228c987c6_video.mxf']
# [asset 2] Id               =  ['urn:uuid:3433a00f-4bc8-4c16-b33c-b0b0d65711af']
# [asset 2] AnnotationText   =  []
# [asset 2] Hash             =  ['ACd4Aky39E608RNnVfAOisPICZ4=']
# [asset 2] Size             =  ['889695']
# [asset 2] Type             =  ['application/mxf']
# [asset 2] OriginalFileName  =  ['wav_3433a00f-4bc8-4c16-b33c-b0b0d65711af_audio.mxf']
# [asset 3] Id               =  ['urn:uuid:4aa03fde-da81-4451-baaa-4d85bf4773d0']
# [asset 3] AnnotationText   =  ['CPL: DCP-INSIDE-CRYPTE_TST-2D-24_C_FR-XX_51_4K_20220102_SMPTE_OV']
# [asset 3] Hash             =  ['YOdZjnEy0JynObowRG9FLXx8tD4=']
# [asset 3] Size             =  ['13102']
# [asset 3] Type             =  ['text/xml']
# [asset 3] OriginalFileName  =  ['CPL_4aa03fde-da81-4451-baaa-4d85bf4773d0.xml']
```