PKL : CODES & FILES

complete the code

THE FILES

Filename	Description
`PKL.min.xml`	Minimal PKL (example)
`PKL.xml`	Complete PKL (example)
`PKL.template-xmlsec.sh`	Simple script using xmlsec1 alongside the PKL template below to be redone
`PKL.template-xmlsec.xml`	Template PKL - without the signature and which can be used to create Digest and Signature
`PKL-ST429-8.xsd`	Definition File (xsd) for PKL
`PKL-getAssetsInfos.py`	Python script that retrieves asset detail

XML VALIDATION

XML Schema reference for PKL : PKL-ST429-8.xsd

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

If you see `validates` : your PKL is syntactical perfect and valid (don't confuse with cryptographic validation)

Here is an example of a bad PKL and the error output message from 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
```

Note that the file header PKL-ST429-8.xsd integrates two imports of XSD files :

```
<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"/>
```

These both imports are used to validate the rest of XML schema, because PKL-ST429-8.xsd validates only the specific part of the PKL. The rest follows the XML basis standard and XML Signature standard. Therefore, the standardized XSD files will be required.

Tools such as xmllint will automatically import these files to complete their parser. However, if you don't have internet access, simply download the two files separately (xml.xsd + xmldsig-core-schema.xsd) and place them at the root directory. Then, modify the PKL-ST429-8.xsd schema to point to the local files. Example:

```
<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"/>
```

GET IDENTIFIER (ID) OF PKL

USING XMLLINT :

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

USING SHELL - SHORT FORM :

```
$ 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
```

USING SHELL - LONG FORM :

```
$ 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 | head -n1
urn:uuid:ce5c22c8-a640-428c-9004-2107e1f3c94e
```

USING 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]
```

GET DETAIL OF EACH ASSETS

USING XMLLINT :

From the element **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
```

From the element **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
```

From the root :

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

Warning : in this example, with **Id**, you will get ALL **Id** from the XML file.

USING PYTHON :

You can find the code below here :

```
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-b0b0d6571
# [assets ] AnnotationText    =  ['CPL: DCP-INSIDE-CRYPTE_TST-2D-24_C_FR-XX_51_4K_20220102_SMPTE_OV']
# [assets ] Hash              =  ['hnBSgENJXOvI6bfpat7GA1VImss=', 'ACd4Aky39E608RNnVfAOisPICZ4=', 'YOdZjnEy0JynObowRG9FLXx8tD4
# [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-b0b0d657

# == 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']
```