

Références	SMPTE 422-2014 - MXF - Mapping JPEG2000 Codestreams into the MXF Generic Container SMPTE 429-4-2006 - DCP - MXF JPEG2000 Application SMPTE 429-3-2007 - DCP - Sound And Pictures Track File ¹ SMPTE 377-1-2011 - MXF - File Format Specification - RGBA Essence Descriptor (p148) SMPTE 384M-2005 - MXF - Mapping of Uncompressed Pictures into the Generic Container
Modèle KLV	Data Item
Universal Label	060e2b34.01020101.0d010301.15010801 - Picture Essence, non-chiffré 060e2b34.02530101.0d010101.01012900 - RGBA Essence Descriptor 060e2b34.02530101.0d010101.01015a00 - JPEG2000 Picture Sub-Descriptor

PRÉFACE

Dans un DCP, les images sont stockées une par une dans leurs KLVs spécifiques placés dans le **Body** : chaque frame va être séparée des autres et stockée dans un KLV **Picture Essence**.

Le SMPTE a fait le choix de stocker chaque image du film séparément et sans avoir de lien entre la précédente et la suivante.

Dans des encodages de type H264/265/MPEG (et autres), en simplifiant, pour permettre de gagner le plus possible en termes de poids, une bonne partie des frames sont des [extrapolations des images précédentes](#), et à certains intervalles, une image référence complète s'intercale. Imaginez cela comme si vous aviez une photo fixe, pour créer du mouvement, vous rajoutez des calques transparents par-dessus avec seulement les modifications des calques inférieurs.

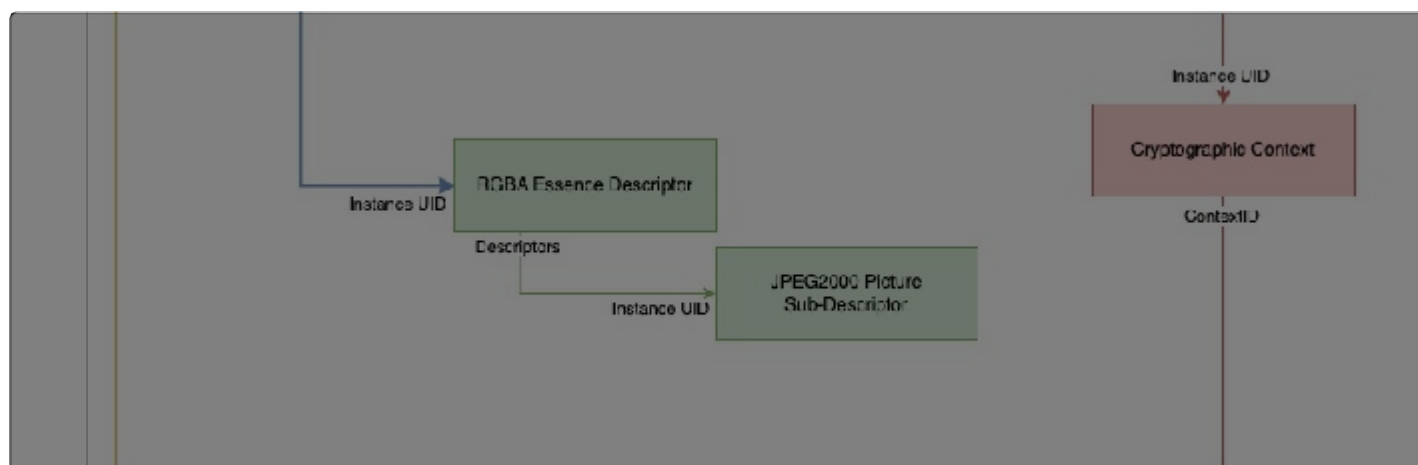
Ce choix technique n'a pas été fait pour un DCP, car si une frame saute, cela va générer des artefacts inacceptables en projection cinéma. Il fallait donc que chaque frame soit indépendante : si une frame saute, cela ne sera pas trop grave (toute proportion gardée), la suivante sera présente pour compenser.

Notez que dans ce chapitre, nous allons parler exclusivement d'un MXF non-chiffré, donc les KLV seront d'un certain type ([Data Item](#)) et chaque frame sera en clair. Dans la version chiffrée, les KLV seront légèrement différents, vous retrouverez l'explication de ces différences dans le chapitre [Cryptographie des MXF](#)

LES KLV HEADERS

Pour stocker du JPEG2000, on a besoin de quelques KLV supplémentaires dans l'entête du MXF :

RGBA Essence Descriptor & JPEG2000 Picture Sub-Descriptor dont voici la relation :



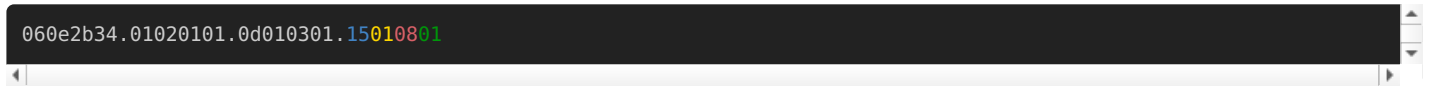
Vous retrouverez les chapitres pour ces deux KLV headers :

- [RGBA Essence Descriptor](#)
- [JPEG2000 Picture Sub-Descriptor](#)

LES KLV BODY : PICTURE ESSENCE

Chaque frame sera conservé dans un KLV "Picture Essence" unique : nous allons donc avoir un ou plusieurs - des milliers pour un film complet - de KLV "Picture Essence"

Son **Universal Label** sera `060e2b34.01020101.0d010301.15010801` qui se décomposera ainsi :

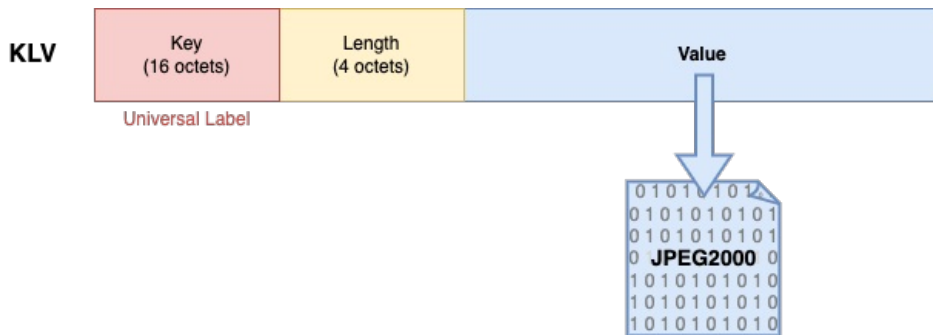


Chaque octet de fin représente un statut :

- **15** indique un **Generic Container Picture Item** ¹, en d'autre terme, un container générique pour stocker des images.
- **01** indique le nombre d'image dans le container. Ici, nous n'aurons qu'une seule image dans ce container.
- **08** indique un statut **Not Clip-wrapped JPEG 2000 Picture Element**.
- **01** indique une sorte de numéro d'index pour l'image, pas d'inquiétude, la valeur sera toujours 1.

Un KLV **Picture Essence** est le plus simple : il suffit de lire l'ensemble de la **Value** du KLV qui intègre la frame entière : des entêtes JPEG2000 jusqu'aux données de l'image : vous aurez l'ensemble du fichier JPEG2000 sans distinction. Il n'y aucun traitement à appliquer, aucune conversion. Le fichier JPEG2000 sera donc une image en **X'Y'Z 12 bits**.

Vous pouvez enregistrer directement le contenu de la Value du KLV dans un fichier [JPEG2000 J2C](#).



CODE : LIRE ET EXTRAIRE LES KLV PICTURE ESSENCE

[Code source](#) pour lire les KLV **Picture Essence** et les extraire :

```

import sys

# Conversion en int
def to_int(length : bytes = b'') -> int:
    return int.from_bytes(length, byteorder='big')

if len(sys.argv) < 2:
    print("Usage: %s <mxfile>" % sys.argv[0])
    sys.exit(1)

mxfile = sys.argv[1]

with open(mxfile, "rb") as file:

    while True:

        # Key : Universal Label
        key = file.read(16)

        # End of file
        if not key:
            break

        # Length (BER format) (quick & dirty, see KLV chapter why)
        length = to_int(file.read(4)[1:]) # BER format - read last 3 bytes

        # Value
        value = file.read(length)

        # Filter by KLV "Picture Essence"
        if key.hex() != "060e2b34010201010d01030115010801":
            continue

        # Show each KLV
        print("{key} - {length:>6d} bytes - {value}...".format(
            key = key.hex(),
            length = length,
            value = value[0:16].hex()
        ))

        # write Plaintext to file
        filename = "output_%d.j2c" % file.tell()
        print(f"Extract {filename}")
        with open(filename, "wb") as f:
            f.write(value)

```

Le code source n'est pas plus compliqué que cela, il suffit de passer de KLV en KLV puis de s'arrêter au KLV **Picture Essence**, de lire le **Length** et de lire le **Value**.

Un avantage avec le KLV **Picture Essence JPEG2000**, c'est que l'ensemble du fichier JPEG2000 est stocké dans **Value**, il suffit d'extraire la donnée brute et de sauvegarder dans un fichier pour avoir un fichier JPEG2000 complet et prêt à l'utilisation.

CODE : ECRITURE NOS PROPRES KLV PICTURE ESSENCE

Coming soon...

CRYPTOGRAPHIE

Notez que tout ce que vous aurez lu dans ce chapitre - ainsi que sur les KLV liés [RGBA Essence Descriptor](#) & [JPEG2000 Picture Sub-Descriptor](#) - s'appliqueront au chapitre sur la [Cryptographie dans les MXF](#) (à l'exception que le KLV **Essence Container** sera un **Encrypted Essence Container**)

COLORIMÉTRIE XYZ

ENCODAGE JPEG2000

Voir le chapitre [JPEG2000](#)

STÉRÉOSCOPIE (3D)

Voir le chapitre [Essence Picture : Stereoscopique \(3D\)](#)

HIGH FRAME RATE (HFR)

Voir le chapitre [Essence Picture : High Frame Rate \(HFR\)](#)

NOTES

- =====
1. « *Track Files shall use the MXF Generic Container (GC) SMPTE 379M. Track Files shall use GC frame-based essence mappings* » -- SMPTE 429-3-2007 - DCP - Sound And Pictures Track File ↩