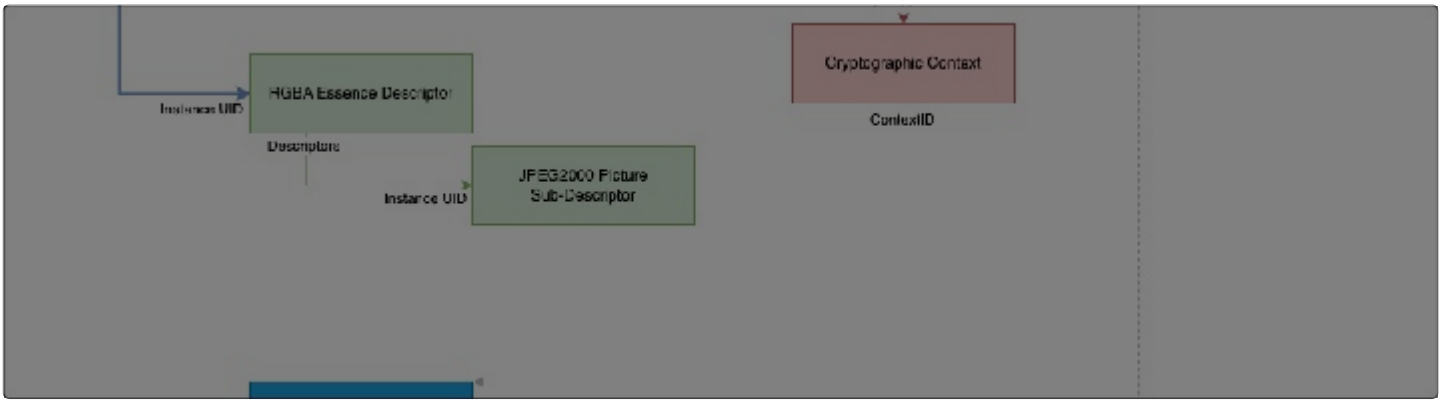
MXF : PICTURE : JPEG2000 PICTURE SUB-DESCRIPTOR

Références	SMPTE 422 - MXF - Mapping JPEG2000 Codestreams into the MXF Generic Container SMPTE 429-4/ - DCP - MXF JPEG2000 Application SMPTE 429-3 - DCP - Sound And Pictures Track File [1] SMPTE 377-1 - MXF - File Format Specification - RGBA Essence Descriptor (p148) SMPTE 384M - MXF - Mapping of Uncompressed Pictures into the Generic Container ISO/IEC 15444-1 — JPEG 2000 Image Coding System — Part 1: Core Coding System ISO/IEC 15444-3 — JPEG 2000 Image Coding System — Part 3: Motion JPEG ISO/IEC 15444-12 — JPEG 2000 Image Coding System — Part 12: ISO Base ISO/IEC 15444-1/Amd 1:2006 - Profiles for Digital Cinema Applications (intégré à 15444-1)
Modèle KLV	Data Item

Universal Label	`060e2b34.02530101.0d010101.01012900 - RGBA Essence Descriptor`
	`060e2b34.02530101.0d010101.01015a00 - JPEG2000 Picture Sub-Descriptor`
	`060e2b34.01020101.0d010301.15010801 - Picture Essence, non-chiffré`
	`060e2b34.02040101.0d010301.027e0100 - Encrypted Essence (SMPTE)`
	`060e2b34.02040107.0d010301.027e0100 - Encrypted Essence (Interop)`

PRÉFACE



Le KLV **JPEG2000 Picture Sub-Descriptor** est défini dans le **SMPTE 422-2014** - MXF - Mapping JPEG2000 Codestreams into the MXF Generic Container (Chapitre 8.2 - JPEG 2000 Picture Sub-Descriptor) et dans **SMPTE 429-4** - MXF JPEG2000 Application

Ce KLV va vous donner des informations spécifiques à l'imagerie JPEG2000 : Certains items regorgent de valeurs très précises sur les entrailles du JPEG2000, comme l'item **Coding Style Default** (COD) et l'item **Quantization Default** (QCD).

Ces informations peuvent être retrouvées en analysant le JPEG2000 stocké dans le container **Picture Essence** (voir ci-dessous) ou dans le container **Encrypted Essence** (voir chapitre MXF chiffré)

LES MÉTADONNÉES

Voici un exemple de rendu des métadonnées avec le fichier 2D.mxf :

```
# mxf-analyzer -f "assets/MXF/2D.mxf" --filter "JPEG2000 Picture Sub-Descriptor" -vv
╔═══════════════════════════════════╦═══════════════════════════════════════════════════════════════════════════════════════════════════
║   3C0A - Instance UID             ║  6054268f.8fdf47ba.98db7167.5d3f704b
║   FFFE - Rsiz                     ║  4K D-Cinema Profile (04h)
║   FFFD - Xsiz                     ║  4096
║   FFFC - Ysiz                     ║  2160
║   FFFB - XOsiz                    ║  0
║   FFFA - YOsiz                    ║  0
║   FFF9 - XTsiz                    ║  4096
║   FFF8 - YTsiz                    ║  2160
║   FFF7 - XTOsiz                   ║  0
║   FFF6 - YTOsiz                   ║  0
║   FFF5 - Csiz                     ║  3
║   FFF4 - Picture Component Sizing ║  3 elements (3 bytes each) =>
║                                   ║  - {Ssiz : 11, XRSiz : 1, YRSiz : 1}
║                                   ║  - {Ssiz : 11, XRSiz : 1, YRSiz : 1}
║                                   ║  - {Ssiz : 11, XRSiz : 1, YRSiz : 1}
║   FFF3 - Coding Style Default     ║  - Scod: 0x01
║                                   ║  - SGcod: ProgressionOrder: 0x04
║                                   ║  - SGcod: NumberLayers: 0x0001
║                                   ║  - SGcod: MultipleComponentTransformation: 0x01
║                                   ║  - SPcod: NumberOfDecompositionLevels: 0x06
║                                   ║  - SPcod: CodeBlock-Width: 0x03
║                                   ║  - SPcod: CodeBlock-Height: 0x03
║                                   ║  - SPcod: CodeBlock-Style: 0x00
║                                   ║  - SPcod: WaveletTransformation: 0x00 (Irreversible)
║                                   ║  - SPcod: PrecinctSizeSubband: 0x77
║                                   ║  - SPcod: PrecinctSizeResolutionsLevels: 6 elements => 0x88, 0x88, 0x88, 0x88, 0x88, 0x88
║   FFF2 - Quantization Default     ║  - Sqcd-all: 0x42
║                                   ║  - Sqcd[ 1]: 0x7f34
║                                   ║  - Sqcd[ 2]: 0x7ef1
║                                   ║  - Sqcd[ 3]: 0x7ef1
║                                   ║  - Sqcd[ 4]: 0x7eae
║                                   ║  - Sqcd[ 5]: 0x76f1
║                                   ║  - Sqcd[ 6]: 0x76f1
║                                   ║  - Sqcd[ 7]: 0x76ae
║                                   ║  - Sqcd[ 8]: 0x6f02
║                                   ║  - Sqcd[ 9]: 0x6f02
║                                   ║  - Sqcd[10]: 0x6ee0
║                                   ║  - Sqcd[11]: 0x674d
║                                   ║  - Sqcd[12]: 0x674d
║                                   ║  - Sqcd[13]: 0x6767
║                                   ║  - Sqcd[14]: 0x5003
║                                   ║  - Sqcd[15]: 0x5003
║                                   ║  - Sqcd[16]: 0x5044
║                                   ║  - Sqcd[17]: 0x57d2
║                                   ║  - Sqcd[18]: 0x57d2
║                                   ║  - Sqcd[19]: 0x5760
╚═══════════════════════════════════╩═══════════════════════════════════════════════════════════════════════════════════════════════════
```

Rsiz indique le type de profile JPEG2000 (*decoder capabilities*) :

Valeur	Profile Cinema
`0x0003`	2K D-Cinema Profile
`0x0004`	4K D-Cinema Profile

Il en existe d'autres, notamment pour IMF.

Xsiz et **Ysiz** est la taille de la grille JPEG2000 de référence (voir le chapitre JPEG2000 pour comprendre)

XOsiz et **YOsiz** sont les positions de décalage de l'image dans la grille de référence. Ce ne sera jamais le cas, on sera toujours avec une origine à {0, 0}.

XTsiz et **YTsiz** est la taille de la tile de référence dans la grille de référence. (idem, si vous ne comprenez pas le principe de tile, voir le chapitre JPEG2000 pour comprendre)

XTOsiz et **XTOsiz** sont les positions de décalage pour le tile de référence dans la grille de référence. Elles seront toujours aussi à {0, 0}

Csiz est le nombre de composant (Component) dans l'image. En simplifiant, sous ce terme barbare se cache juste la désignation pour le X'Y'Z' : Nous aurons que 3 éléments pour spécifier chaque composante du X'Y'Z.

Picture Component Size sera toujours de 3 valeurs de 3 octets chacun dont les valeurs seront 11, 1, 1 (0x0b, 0x01, 0x01). Ce sont des valeurs Ssiz, XRSiz, YZSiz que vous découvrirez dans la norme JPEG2000 (ISO/IEC 15444-1 Annex A.5.1)

- **Ssiz** représente la précision (Component sample precision). La valeur 11 peut être bizarre alors qu'on parle d'une précision de 12 bits. C'est parce que la norme JPEG2000 instaure que la véritable valeur est "valeur + 1". Autre élément important: l'octet est divisé en 2 parties : le premier bit spécifie si les valeurs des composants seront signés ou non signés (signed/unsigned). Par chance, dans notre cas, on aura toujours le premier bit à 0, donc unsigned.

Hexa	Binaire	Décimale	Valeur réelle
`0x0b`	`(0)0001011`	(0 = unsigned) - 11 octets	11 + 1 = 12 octets unsigned
`0x8b`	`(1)0001011`	(1 = signed) - 11 octets	11 + 1 = 12 octets signed

Vous remarquerez qu'avec 2 valeurs hexadécimales `0x0b` et `0x8b`, nous avons la même valeur de bitdepth (`11+1` octets). Seul le premier bit change pour savoir si nous aurons que des nombres uniquement positifs (unsigned int) ou des nombres négatifs et positifs (signed int).

- **XRsiz** représente la séparation horizontale d'un échantillon du énième composant par rapport à la grille de référence (plus d'infos dans JPEG2000). C'est le nombre de bit, nous serons toujours à 1.
- **YRsiz** représente la séparation verticale d'un échantillon du énième composant par rapport à la grille de référence (plus d'infos dans JPEG2000). C'est le nombre de bit, nous serons toujours à 1.

Coding Style Default sont des valeurs pour les paramètres JPEG2000, il va décrire le style de codage, le nombre de niveaux de décomposition, et d'autres éléments sur l'image et ses tiles.

Les bits se décomposent en groupe par thème d'option. À l'intérieur, chaque bit définit des options, des paramètres ou des informations sur le JPEG2000.

Voici quelques informations rapides (vous trouverez de plus amples informations dans le chapitre JPEG2000) :

Renommer les différents acronymes pour les faire correspondre au chapitre JPEG2000

> Les documentations DCI/SMPTE et ISO (pour le JPEG2000) utilisent parfois des acronymes légèrement différents pour désigner les mêmes choses, par exemple POD (Progression Order Change - Default) et POC (Progression Order Change)

- **Scod (1 octet)** : Définition des paramètres du style de la codification (Style Code) pour tous les composants (components). Exemple ici, nous avons `0x01` = `b00000001` désigne trois paramètres :
  - *Entropy coder with precincts*
  - *No SOP marker segments used*
  - *No EPH marker used*
- **SGcod (4 octets)** : Définition des paramètres du style de la codification désignée dans **Scod** :
  - **Progression Order** (`0x04` = `b00000100`) = *Component-position-resolution level-layer progression* (**CPRL**)
  - **NumberLayers** (`0x0001`) indique le nombre de layer dans le JPEG2000, il sera toujours à 1 (single tile).
  - **Multiple Component Transformation** (`0x01` = `b00000001`) = *Component transformation used on components 0, 1, 2 for coding efficiency*
- **SPcod (5 octets et plus)** : Définition des paramètres du style de la codification désignée dans **Scod** :
  - **Number Of Decomposition Levels** (`0x06`) est le nombre de niveau dans la décomposition (à ne pas confondre avec resolution level):
    - Pour un 2K, nous aurons 5 niveaux.
    - Pour un 4K, nous aurons 6 niveaux.
  - **CodeBlock** : Dimensions et style des code blocks (doh!). Avec un CodeBlock-Style à `0x00`, nous avons les informations suivantes :
    - *No selective arithmetic coding bypass*
    - *No reset of context probabilities on coding pass boundaries*
    - *No termination on each coding pass*
    - *No vertically causal context*
    - *No predictable termination*
    - *No segmentation symbols are used*
  - **Wavelet Transformation** : Avec `0x00`, il est indiqué que notre JPEG2000 est irréversible (Transformation 9/7): le JPEG2000 permet de conserver les informations de l'image d'origine, ce qui peut permettre de la reconstruire, donc sans destruction (avec la transformation 5/3). Ici, ce n'est pas le cas, il y aura destruction durant la compression.
  - **Precinct Size** : Le premier paramètre correspond à une valeur liée au LL-Subband, puis les autres valeurs (`0x88…`) correspondent à chaque niveau de résolution successive dans l'ordre. On constatera que nous avons 6 valeurs, comme le nombre défini dans notre **Number Of Decomposition Levels** (en 2K, nous n'aurions eu que 5 valeurs)

Quantization Default donne les informations (et surtout valeurs) de la quantization utilisée pour la compression de tous les composants (components). La structure est de cette forme :

- Un **Sqcd** (1 octet / 8 bits) : pour tous les composants (components).
- Un ou plusieurs **Sqcd** (2 octets / 16 bits * x) : un par sub-band. Appelée parfois SPqcd. C'est un code binaire sur 16 bits où les 5 premiers bits est le mantisse, les 11 suivants est l'exposant. (16 bits pour la transformation 9/7 ou seulement 8 bits avec la transformation 5/3, avec une mantisse en 3 bits et un exposant en 5 bits, vous ne verrez ce cas que dans le cadre de l'IMF, et non du DCP)

Toutes ces informations sont stockées également dans les métadonnées du JPEG2000.
Voici un output de notre parseur JPEG2000 avec le fichier J2C de notre précédent MXF :

```
# jpeg2000-parser.py "2D.j2c" --verbose

[SOC] Start of codestream (0xff4f)

[SIZ] Image and tile size (0xff51)
size: 47 bytes
data: 45 bytes : 000400001000000008700000000000000000000010000000087000000000000000000030b01010b01010b0101
---- Provides information about the uncompressed image such as the width and height of the reference grid,
---- the width and height of the tiles, the number of components, component bit depth,
---- and the separation of component samples with respect to the reference grid
SIZ - rsiz   : Profile 4
SIZ - Xsiz   : 4096 px
SIZ - Ysiz   : 2160 px
SIZ - XOsiz  : 0 px
SIZ - YOsiz  : 0 px
SIZ - XTsiz  : 4096 px
SIZ - YTsiz  : 2160 px
SIZ - XTOsiz : 0 px
SIZ - YTOsiz : 0 px
SIZ - Csiz   : 3 components
SIZ - Component 1 - ssizDepth : 11 -> 12 bits          Components Parameters (00001011)
SIZ - Component 1 - xRsiz     : 1 bit(s)               Horizontal separation of a sample
SIZ - Component 1 - yRsiz     : 1 bit(s)               Vertical separation of a sample
SIZ - Component 2 - ssizDepth : 11 -> 12 bits          Components Parameters (00001011)
SIZ - Component 2 - xRsiz     : 1 bit(s)               Horizontal separation of a sample
SIZ - Component 2 - yRsiz     : 1 bit(s)               Vertical separation of a sample
SIZ - Component 3 - ssizDepth : 11 -> 12 bits          Components Parameters (00001011)
SIZ - Component 3 - xRsiz     : 1 bit(s)               Horizontal separation of a sample
SIZ - Component 3 - yRsiz     : 1 bit(s)               Vertical separation of a sample

[COD] Coding style default (0xff52)
size: 19 bytes
data: 17 bytes : 0104000101060303000077888888888888
---- Describes the coding style, decomposition, and layering that is
---- the default used for compressing all components of an image  or a tile
COD - Scod   : 01                  Binary Parameters: 00000001
COD - Progression order : 04               Binary Parameters: 00000100
COD - Number of Layers  : 1
COD - Multiple Component Transform : 01 00000001
COD - Decomposition levels : 06
COD - CodeBlockSize : 32 x 32
COD - CodeBlock style : 00000000
COD - CodeBlock style parameter bit7 : No selective arithmetic coding bypass
COD - CodeBlock style parameter bit6 : No reset of context probabilities on coding pass boundaries
COD - CodeBlock style parameter bit5 : No termination on each coding pass
COD - CodeBlock style parameter bit4 : No vertically stripe causal context
COD - CodeBlock style parameter bit3 : No predictable termination
COD - CodeBlock style parameter bit2 : No segmentation symbols are used
COD - CodeBlock style parameter bit1 : Unknown parameter
COD - CodeBlock style parameter bit0 : Unknown parameter
COD - TransformType : 9-7 irreversible wavelet (00000000)
COD - PrecinctSize 1 : 128 x 128    (0111:0111)
COD - PrecinctSize 2 : 256 x 256    (1000:1000)
COD - PrecinctSize 3 : 256 x 256    (1000:1000)
COD - PrecinctSize 4 : 256 x 256    (1000:1000)
COD - PrecinctSize 5 : 256 x 256    (1000:1000)
COD - PrecinctSize 6 : 256 x 256    (1000:1000)
COD - PrecinctSize 7 : 256 x 256    (1000:1000)

[QCD] Quantization default (0xff5c)
size: 41 bytes
data: 39 bytes : 427f347ef17ef17eae76f176f176ae6f026f026ee0674d674d67675003500350445 7d257d25760
---- Describes the quantization default used for compressing all components not defined by a QCC marker segment.
---- The parameter values can be overridden for an individual component by a QCC marker segment in either the main or tile-part header
QCD - Sqcd (Scalar coefficient dequantization), Quantization style for all components: 01000010
QCD - Sqcd binary parameters (bit1-3): Number of guard bits 0–7: 0b010 -> 2 bits
QCD - Sqcd: 0b00010 -> Scalar explicit
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band  0:
    -> SPqcd    : 0111111100110100  (0x7f34)
    -> Mantissa : 01111             : 15
    -> Exponent :      11100110100 : 1844
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band  1:
    -> SPqcd    : 0111111011110001  (0x7ef1)
    -> Mantissa : 01111             : 15
    -> Exponent :      11011110001 : 1777
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band  2:
    -> SPqcd    : 0111111011110001  (0x7ef1)
    -> Mantissa : 01111             : 15
    -> Exponent :      11011110001 : 1777
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band  3:
    -> SPqcd    : 0111111010101110  (0x7eae)
    -> Mantissa : 01111             : 15
    -> Exponent :      11010101110 : 1710
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band  4:
```

```
    -> SPqcd    : 0111011011110001  (0x76f1)
    -> Mantissa : 01110             : 14
    -> Exponent :      11011110001 : 1777
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band  5:
    -> SPqcd    : 0111011011110001  (0x76f1)
    -> Mantissa : 01110             : 14
    -> Exponent :      11011110001 : 1777
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band  6:
    -> SPqcd    : 0111011010101110  (0x76ae)
    -> Mantissa : 01110             : 14
    -> Exponent :      11010101110 : 1710
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band  7:
    -> SPqcd    : 0110111100000010  (0x6f02)
    -> Mantissa : 01101             : 13
    -> Exponent :      11100000010 : 1794
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band  8:
    -> SPqcd    : 0110111100000010  (0x6f02)
    -> Mantissa : 01101             : 13
    -> Exponent :      11100000010 : 1794
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band  9:
    -> SPqcd    : 0110111011100000  (0x6ee0)
    -> Mantissa : 01101             : 13
    -> Exponent :      11011100000 : 1760
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band 10:
    -> SPqcd    : 0110011101001101  (0x674d)
    -> Mantissa : 01100             : 12
    -> Exponent :      11101001101 : 1869
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band 11:
    -> SPqcd    : 0110011101001101  (0x674d)
    -> Mantissa : 01100             : 12
    -> Exponent :      11101001101 : 1869
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band 12:
    -> SPqcd    : 0110011101100111  (0x6767)
    -> Mantissa : 01100             : 12
    -> Exponent :      11101100111 : 1895
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band 13:
    -> SPqcd    : 0101000000000011  (0x5003)
    -> Mantissa : 01010             : 10
    -> Exponent :      00000000011 : 3
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band 14:
    -> SPqcd    : 0101000000000011  (0x5003)
    -> Mantissa : 01010             : 10
    -> Exponent :      00000000011 : 3
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band 15:
    -> SPqcd    : 0101000001000100  (0x5044)
    -> Mantissa : 01010             : 10
    -> Exponent :      00001000100 : 68
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band 16:
    -> SPqcd    : 0101011111010010  (0x57d2)
    -> Mantissa : 01010             : 10
    -> Exponent :      11111010010 : 2002
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band 17:
    -> SPqcd    : 0101011111010010  (0x57d2)
    -> Mantissa : 01010             : 10
    -> Exponent :      11111010010 : 2002
QCD - SPqcd (Exponent+Mantissa), Quantization step size value sub-band 18:
    -> SPqcd    : 0101011101100000  (0x5760)
    -> Mantissa : 01010             : 10
    -> Exponent :      11101100000 : 1888

[POC] Progression Order Change (0xff5f)
size: 16 bytes
data: 14 bytes : 00000001060304060000001070304
---- Describes the bounds and progression order for any progression order other than default in the codestream ----
POC - [0] RSpoc, Resolution index for the start of a progression: 0x00
POC - [0] CSpoc, Component index for the start of a progression: 0x00
POC - [0] LYEpoc, Layer index for the end of a progression: 0x0001
POC - [0] REpoc, Resolution index for the end of a progression: 0x06
POC - [0] CEpoc, Component index for the end of a progression: 0x03
POC - [0] Ppoc, Progression order: 0x04
POC - [1] RSpoc, Resolution index for the start of a progression: 0x06
POC - [1] CSpoc, Component index for the start of a progression: 0x00
POC - [1] LYEpoc, Layer index for the end of a progression: 0x0001
POC - [1] REpoc, Resolution index for the end of a progression: 0x07
POC - [1] CEpoc, Component index for the end of a progression: 0x03
POC - [1] Ppoc, Progression order: 0x04

[TLM] Tile-part lengths, main header (0xff55)
size: 34 bytes
data: 32 bytes : 005000000043e100000014a90000000caf0000002ecc000000063900000001b5
---- Describes the length of every tile-part in the codestream
---- Each tile-part’s length is measured from the first byte of the SOT marker segment
---- to the end of the data of that tile-part. The value of each individual tile-part length in the TLM
---- marker segment is the same as the value in the corresponding Psot in the SOT marker segment.
TLM - Ztlm: Index of this marker segment: 00 00000000 0
TLM - Stlm: Size of the Ttlm and Ptlm parameters: 01010000
TLM - Stlm Bit-Parameters: SP = 1; Ptlm parameter 32 bits
TLM - Stlm Bit-Parameters: ST = 1; Ttlm parameter 8 bits
```

```
TLM - Tile number of the ith tile-part : 0x00
TLM - Length from the beginning of the SOT marker: 17377 bytes
TLM - Tile number of the ith tile-part : 0x00
TLM - Length from the beginning of the SOT marker: 5289 bytes
TLM - Tile number of the ith tile-part : 0x00
TLM - Length from the beginning of the SOT marker: 3247 bytes
TLM - Tile number of the ith tile-part : 0x00
TLM - Length from the beginning of the SOT marker: 11980 bytes
TLM - Tile number of the ith tile-part : 0x00
TLM - Length from the beginning of the SOT marker: 1593 bytes
TLM - Tile number of the ith tile-part : 0x00
TLM - Length from the beginning of the SOT marker: 437 bytes

[CME] Comment and extension (0xff64)
size: 40 bytes
data: 38 bytes : 000143726561746564207769746820445653204360c69707374657220352e31302e302e313400
---- Comment, extension and unstructured data in the header ----
CME - Registration values 1: Text ISO 8859-1
CME - Text: Created with DVS Clipster 5.10.0.14

[SOT] Start of tile-part (0xff90)
size: 10 bytes
data: 8 bytes : 0000000043e10006
---- Marks the beginning of a tile-part and the index of its tile within a codestream
---- The tile-parts of a tile shall appear in order (see TPsot) in the codestream
---- but not necessarily consecutively.
SOT - Isot, Tile number: 0
SOT - Psot, Length from the beginning of the first byte of this SOT: 17377
SOT - TPsot, Tile-part instance: 0
SOT - TNsot, Number of tile-parts: 6

[SOD] Start of data (0xff93)
size: 39911 bytes
data: 39909 bytes : e7ff00a35ffc03fafe09efe09ccbad4ddbf3a0a9bf7b03003629b6e6f9bdb6ff(...)

[EOC] End of codestream (0xffd9)
```

Toutes ces informations peuvent être difficilement compréhensibles, il faudra passer sur le chapitre JPEG2000 pour les comprendre.

STRUCTURES DES DONNÉES

Local Tag	Nom de l'attribut	Type	Taille (*)	Fixe/Variable SMPTE	Obligatoire
3C0A	Instance ID	UUID	16 octets	Fixe	**Obligatoire**
0102	Generation UID	UUID	16 octets	Fixe	Optionnel
0101	Object Class	AUID	16 octets	Fixe	Optionnel
dyn	ApplicationPlug-In Batch	Array[UL]	8+16n	Variable	Optionnel
2F01	Locators	Array[UL]	8+16n	Variable	Optionnel
dyn	SubDescriptors	Array[UID]	8+16n	Variable	Optionnel
(dyn)	Rsiz	UInt16	2 octets	Fixe	Obligatoire
(dyn)	Xsiz	UInt32	4 octets	Fixe	Obligatoire
(dyn)	Ysiz	UInt32	4 octets	Fixe	Obligatoire
(dyn)	XOsiz	UInt32	4 octets	Fixe	Obligatoire
(dyn)	YOsiz	UInt32	4 octets	Fixe	Obligatoire
(dyn)	XTsiz	UInt32	4 octets	Fixe	Obligatoire
(dyn)	YTsiz	UInt32	4 octets	Fixe	Obligatoire
(dyn)	XTOsiz	UInt32	4 octets	Fixe	Obligatoire
(dyn)	YTOsiz	UInt32	4 octets	Fixe	Obligatoire
(dyn)	Csiz	UInt16	2 octets	Fixe	Obligatoire
(dyn)	Picture Component Sizing	J2KComponentSizingArray	(4 + 4) + (3 * x) octets	Fixe	Obligatoire
(dyn)	Coding Style Default	J2KCodingStyleDefault	Variable	Variable	Optionnel
(dyn)	Quantization Default	J2KQuantizationDefault	Variable	Variable	Optionnel
(dyn)	J2CLayout	RGBALayout	Variable	Variable	Optionnel

HIÉRARCHIE DU FORMAT

Interchange Object → Generic Descriptor → (Generic Sub-Descriptor?) → JPEG2000 Picture Sub-Descriptor

LIRE CE KLV

Code source pour lire les headers :

```
import sys
import io

# 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 <mxf>" % sys.argv[0])
    sys.exit(1)

mxf_file = sys.argv[1]

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

    while True:

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

        # End of file
        if not key:
            break

        # Length (BER format)
        length = to_int(file.read(4)[1:])  # BER format - read last 3 bytes

        # Value
        value = file.read(length)

        # Filter by KLV : JPEG2000 Picture Sub-Descriptor
        if key.hex() != "060e2b34025301010d01010101015a00":
           continue

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

        # read each item
        data = io.BytesIO(value)

        # "item" is a baby KLV:
        #  localtag (key)  :  2 bytes
        #  item length     :  2 bytes (no BER format)
        #  item value      :  variable bytes

        while True:

            # get local tag (2 bytes)
            localtag = data.read(2)

            if not localtag:
                break

            # get item length (2 bytes, directly int, NOT Ber format)
            item_length = to_int(data.read(2))

            # get item value
            item_value = data.read(item_length)

            # Show each item
            print("{localtag} : {item_length:>2d} : {item_value}".format(
                localtag    = localtag.hex(),
                item_length = item_length,
                item_value  = item_value.hex()
            ))
```

Voici son exécution sur le MXF 2D.mxf :

```
$ ./mxf-klv-picture-jpeg2000-picture-subdescriptor.py "2D.mxf"

# KLV JPEG2000 Picture Sub-Descriptor
060e2b34025301010d01010101015a00 -    181 bytes - 3c0a00106054268f8fdf47ba98db7167...
3c0a : 16 : 6054268f8fdf47ba98db71675d3f704b
fffe :  2 : 0004
fffd :  4 : 00001000
fffc :  4 : 00000870
fffb :  4 : 00000000
fffa :  4 : 00000000
fff9 :  4 : 00001000
fff8 :  4 : 00000870
fff7 :  4 : 00000000
fff6 :  4 : 00000000
fff5 :  2 : 0003
fff4 : 17 : 00000003000000030b01010b01010b0101
fff3 : 17 : 0104000101060303000077888888888888
fff2 : 39 : 427f347ef17ef17eae76f176f176ae6f026f026ee0674d674d6767500350035045 7d257d25760
```

DUPLICATE RGBA Essence Descriptor

On voit qu'avec quelques lignes, on récupère l'ensembles des informations brutes :

- **Colonne 1** : le Local Tag - la clef spécifiant le type de valeur (comme pour la Key d'un KLV)
- **Colonne 2** : la taille en octets de sa value.
- **Colonne 3** : la valeur brute, sans aucune conversion.

Vous devrez maintenant convertir chaque valeur brute dans un format qui lui est propre. Et pour cela, il faudra vous baser sur le **Local Tag** en colonne 1.

Notez que pour les **Local Tag**, vous devrez lire le KLV Primer Pack afin de faire correspondre chaque **Local Tag** avec sa véritable désignation (Universal Label, Type, Taille de chaque sous-élément, Formatage, ..). Plus d'informations dans le chapitre Local Tag.

Même sans convertir, nous pouvons déjà identifier quelques éléments, comme des **Universal Label**.

Prenons un exemple, le `3201` qui a une taille de 16 octets et est - de toute évidence - un Universal Label. C'est le **Picture Essence Coding**, un identifiant qui va définir le type d'encodage et de compression utilisé sur l'essence stocké dans les **Picture Essence**. Pour notre cas, nous aurons (pour l'instant) deux UL disponibles :

- JPEG 2000 **2K** Digital Cinema Profile (`060e2b34.04010109.04010202.03010103`)
- JPEG 2000 **4K** Digital Cinema Profile (`060e2b34.04010109.04010202.03010104`)

Ici, nous voyons que notre UL correspond à JPEG2000 4K. Donc un encodage et compression JPEG2000 spécifique pour le Digital Cinema (JPEG2000 Codestream).

Il existe d'autres UL disponibles pour Picture Essence Coding.

Vous devrez donc, pour chaque valeur, faire correspondre son type et son formatage. En lisant chaque docs SMPTE, on découvre ces petits pokemons d'informations. Fort heureusement, une partie est compilée dans le chapitre Local Tag ou bien dans mxf-analyzer.
