KLV : KEY-LENGTH-VALUE

Each brick of an MXF is named a **KLV** and is represented as follows :

A KLV IS A DATA STRUCTURE WITH THREE PARTS

KLV is the contraction of "*Key, Length, Value*" : it is a data structure defined in following standards :

- **SMPTE 336 : Data Encoding Protocol Using Key-Length-Value**
- **SMPTE 377 : MXF - File Format Specification**, Chapitre 4.3. KLV Coding.

This structure is quite simple, it is divided into three parts: it starts with a key, then the size of the data and finally with the actual data.

- The **Key** is an **unique identifier** that defines the type of data in the **Value** part.
- The **Length** is **the size of the data stored** in the **Value** part.
- The **Value** is the **actual data**

For example, a KLV can be represented in the following schematic form:

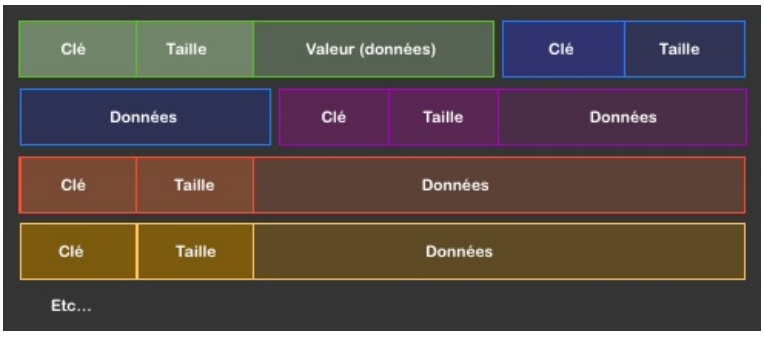
Key	Length	Value
`MyKey`	`32`	`Lorem ipsum dolor sit amet odio.`

As follows :

- The key has as an identifier `MyKey` .
- The size of data is `32 octets` long.
- The sentence `Lorem ipsum dolor sit amet odio.` is the data and is 32 octets long.

This example is extremely simplified to aid understanding. We will be come back to its true SMPTE / DCI form later.

Visually, this is what it looks like inside an MXF with several KLV :



As you can see, we always have a sequence of :

```
Key, Size, Data.
Key, Size, Data.
Key, Size, Data.
...etc...
```

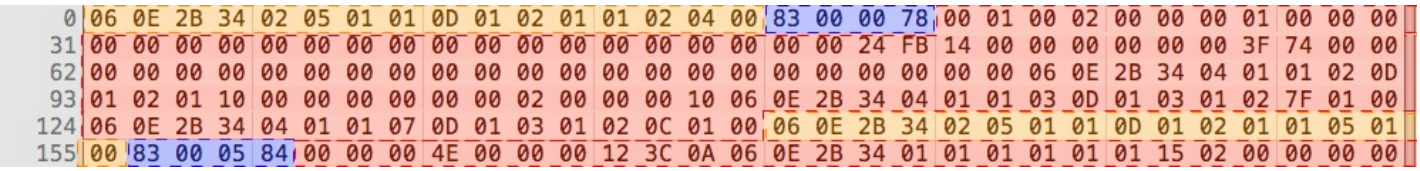
Each KLV has a variable size depending on several criteria including the size of the data.

To quickly describe each part of the KLV of a specific MXF from a DCP :

- **Key** is a specific SMPTE identifier (known as **Universal Label** or **UL**)
  - Its size is 16 octets
  - It starts with the hexadecimal value `060e2b34`
  - For example, the key `060e2b34020501010d01020101020400` is an identifier for a **Partition Pack - Header** which defines an MXF header.
- **Length** determines the data size of the **Value** field.
  - The size of **Length** field is itself variable - from 1 to 8 octets.
  - The first octet determines the size of the **Length** field. [1]
  - The following octets represent the actual size of the data in the **Value** part. [2]
  - For example, with a **Length** field value of `83000078` :
    - The first octet `0x83` indicates that the total size of the **Length** field is 4 octets (we will see why later)
    - The three following octets are therefore `0x00` , `0x00` , and `0x78` . They define the size of the **Value** field. Converted to decimal, `0x78` gives `120` (octets)
- **Value** is the usable data and is made up of :
  - Either raw data
  - Or structured data that can be interpreted using the appropriate documentation

Here is the raw vieuw of the beginning of an MXF file in hexadecimal form :

- The **Key**s are in orange
- The **Length**s are in blue
- The **Value**s are in red



We just quickly saw what KLVs are : A key field, a size field and a data field. Nothing more.

If you understand this principle, you already understand half of how an MXF works :)

We will come back to each part in more depth later.

But before that, a quick explanation of the terms **Universal Set** and **Universal Label** used in SMPTE documentation, because we will use these terms regularly.

UNIVERSAL SET & UNIVERSAL LABEL : WHAT THE HECK IS THAT ?

In the SMPTE Cinematic Universe, we love the sweet little names :

- **KLV** are named **Universal Set**
- The keys / SMPTE identifiers (**Key** field) are named **Universal Label** (or **UL**).

```
KLV -> Universal Set
Key -> Universal Label
```

Here's a simplified diagram:

But why these names ?

Because there are also equivalents of the **Universal Set** and **Universal Label** but for other KLV stored in the **Value** part, a kind of baby KLV. These baby KLV are named **Local Set** and their keys **Local Tag**.

If we summarize the differents between the two :

	KLV	**Baby KLV**
Name of the KLV	Universal Set	Local Set
Name of the key	Universal Label	Local Tag

Don't ask me why the **Local Tag** wasn't named "Local Label", probably the creators were probably drunk.

But we will come back for this concept of "Baby KLV" later in the **Value** chapter.

Now, let's take a look at the Key part.

NOTES

1. Except for exceptions, see chapter MXF-KLV-Length ↩
2. Except for exceptions, see chapter MXF-KLV-Length ↩