KLV : KEY: THE TYPE IDENTIFIER (OR UNIVERSAL LABEL)

The normalization of the key is defined in the **SMPTE 298 : Universal Labels (UL) for Unique Identification of Digital Data** document.

In summary, **it's a unique key for each type of KLV** but not unique for each KLV. We can have the same key multiple times in a MXF file. A perfect example is the KLV for the frame: the same key for each KLV that contains a frame.

A **Universal Label (UL)** follows a specific format :

```
Hexa Format   =  060e2b34020501010d01020101020400
SMPTE Format  =  060e2b34.02050101.0d010201.01020400
```

They exist a lot of Universal Label, here are a few examples below:

Key / Universal Label (UL)	KLV Type
`060e2b34.02050101.0d010201.01020400`	Partition Pack - Header
`060e2b34.02050101.0d010201.01030400`	Partition Pack - Body
`060e2b34.02050101.0d010201.01040400`	Partition Pack - Footer
`060e2b34.02050101.0d010201.01050100`	Primer Pack
`060e2b34.02530101.0d010101.01013000`	Identification
`060e2b34.02530101.0d010101.01013600`	Material Package
`060e2b34.02530101.0d010101.01010f00`	Sequence
`060e2b34.02530101.0d010101.01011100`	Source Clip
`060e2b34.02050101.0d010201.01110100`	Random Index Pack
`060e2b34.02040107.0d010301.027e0100`	Encrypted Essence
`060e2b34.01020101.0d010301.15010801`	Picture Essence
...	

As we can see, each **Universal Label (UL)** identifies a type of KLV.

We can notice that the header (`060e2b34`) is always identical. After that, depending on the type of the KLV, some changes occur :-)

After the header (`060e2b34`), all octets are importants because they act like parameters, options, or status for the KLV. For example, the **Partition Packs** have many identifiers with a common base. Their parameters determine whether it's a Header, a Body or a Footer, and wheter they are complete or incomplete, open or closed, etc.

We can take our example of **Partition Packs** and highlight the difference :

Universal Label (UL)	Type
`060e2b34.02050101.0d010201.01` **`02`** `0400`	Partition Pack - **Header**
`060e2b34.02050101.0d010201.01` **`03`** `0400`	Partition Pack - **Body**
`060e2b34.02050101.0d010201.01` **`04`** `0400`	Partition Pack - **Footer**

In the case of **Partition Pack**, we observe that only the second-to-last octets change (in bold), they respectively represent :

- `02` : `Header`
- `03` : `Body`
- `04` : `Footer`

The value just after (`04`) determines the status of the **Partition Pack** : With the value `04` , the Partition Pack is considered **Closed & Complete**

Each KLV has its own specific characteristics: it will be necessary to read each SMPTE document to understand and collect these precious parameters, and thus identify **Universal Labels** for each type of KLV (they're like Pokémons).

You can also get a list of multiple **Universal Labels** from the following registries, but they won't tell you what each octet corresponds to :

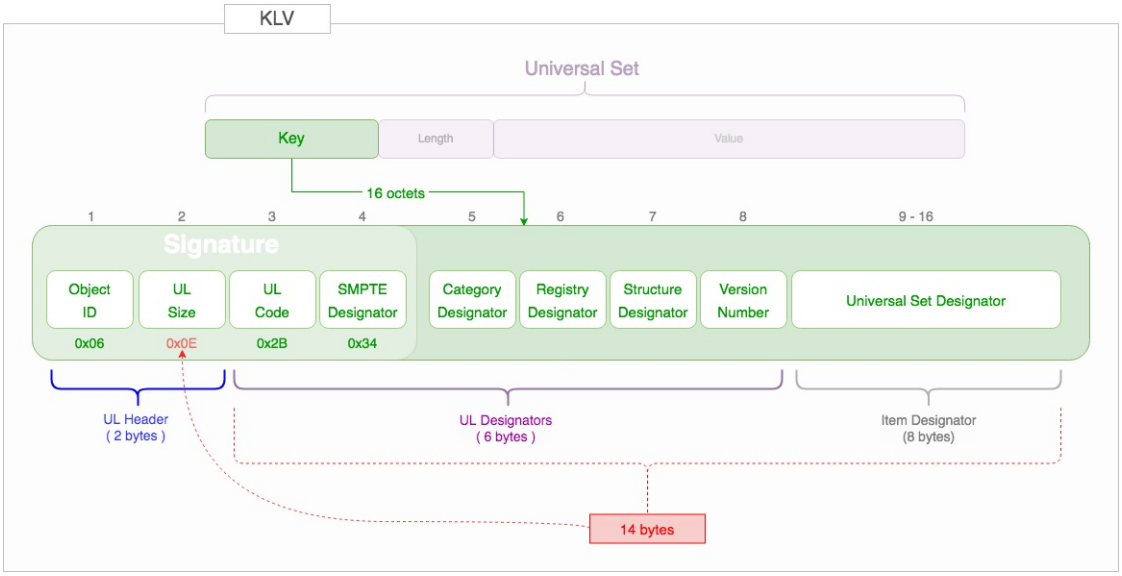
- https://registry.smpte-ra.org/apps/pages/published/
- https://registry.smpte-ra.org/view/published/published_registers/
- https://registry-page.isdcf.com/uls/
- SMPTE Universal Labels compilation from RP224-2003 (warning, may be deprecated)

We will see in detail how a **Universal Label (UL)** is composed, inside a key. Note that knowing all of this is not essential and won't stop you from interpreting data in the **Value** field. It's just a bonus to your knowledge.

INSIDE OF KEY

As we can see, after the key header (`060e2b34`), each octet has its own characteristics.

Here is a schematic representation of the inside of the **K** in a KLV :



The size of the **Universal Label (UL)** is 16 octets and it is subdivisible into three main part :

- **UL Header** (2 octets)
- **UL Designators** (6 octets)
- **Item Designator** (8 octets)

The following table provides a quick overview of the different parts of the key :

UL Header		**UL Designators**						**Item Designator**							
2 octets		6 octets						8 octets							
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Object ID	UL Size	UL Code	SMPTE Designator	Category Designator	Registry Designator	Structure Designator	Version Number	#1	#2	#3	#4	#5	#6	#7	
Signature															
0x06	0x0E	0x2B	0x34												
4 octets															

As standardized, the first four octets **always** start with the hexadecimal sequence: `0x06` , `0x0e` , `0x2b` , `0x34` . That is its signature. And this allows us to quickly identify these KLVs [1].

If you take a look at the **UL Header** (first two octets) : we have **Object ID** + **UL Size** which have a format that will look familiar. We start with an identifier, then a size, and finally other data... Yes, that's indeed the KLV format, which also structures the inside of the key :-)

Thus, the second octet (**UL Size**) indicates the size of the **Key**'s data : `0x0E` represents 14 octets, which correspond to the following octets, from **SMPTE Designator** to the last **Item Designator #8**

Here is a descriptive table of each octet in **Universal Label (UL)** :

N°	Name of the field	Description	Data	
1	Object ID	Its identifier	`0x06`	UL Header
2	UL Size	Remaining size of the key	`0x0E`	UL Header
3	UL Code	Subidentifier ISO + ORG	`0x2B`	UL Designator
4	SMPTE Designator	Subidentifier SMPTE	`0x34`	UL Designator
5	Category Designator	Data Type	`0x01` - `0x7E`	UL Designator
6	Registry Designator	Universal Sets	`0x??`	UL Designator
7	Structure Designator		`0x??`	UL Designator
8	Version Number	Registry version	`0x??`	UL Designator
9-16	Universal-Set Designator	Parameters & Options	....	Item Designator

CATEGORY DESIGNATOR

Category Designator allows us to quickly identify the type of data contained in the **Value** part. To be brief about the main categories :

- **Dictionnaries** are raw data
- **Groups (Set or Packs)** are structured data

Here is an overview table of the different **Category Designator** (5) accompanied by the next **Registry Designator** (6), which allows for more precision about the type of data (in green, the main elements in an MXF from a DCP) :

	Category Designator (5)	Registry Designator (6)	
Dictionnaries	**0x01**	**Raw data**	
	0x01	0x01	Metadata Dictionary
	0x01	0x02	Essence Dictionary
	0x01	0x03	Control Dictionary
	0x01	0x04	Types Dictionary
Groups (Sets or Packs)	**0x02**	**Structured data**	
	0x02	0x01	Universal Set
	0x02	0x02	Global Set
	0x02	0x03	Local Set
	0x02	0x13	Local Set
	0x02	0x53	Local Set
	0x02	0x04	Variable Length Pack
	0x02	0x05	Defined Length Pack
	0x02	0x06	Reserved / Forbidden
Wrappers & Containers	**0x03**		
	0x03	0x01	Simple Wrapper & Container
	0x03	0x02	Complex Wrapper & Container
SMPTE Labels	**0x04**		**Not used as KLV key**
Registered Private Information	**0x05**		
Reserved	**0x06**		
...	**...**		
Reserved	**0x7E**		

REGISTRY DESIGNATOR

Registry Designator identifies the specific registry within a specific category. (for example, **Essence Dictionary** which is a subcategory of "**Dictionaries**"). See the table above for the different **Registry Designator**.

You may have noticed the presence of several **Local Sets** with three different **Registry Designator**. By default in the KLV standards (2004), **Local Set** has a **Registry Designator** code of `0x03`. In the MXF standards (2011), the different types of KLV that comply with **Local Sets** have the **Registry Designator** codes of `0x53` or `0x13`.

Why are there these different **Registry Designator** for the **Local Set** ? We won't spend too much time on this because we'll cover it in the MXF chapter MXF : KLV : Values - Local Sets along with the notes [2], but to quickly summarize, this will have an impact on the data stored in the **Value** part. Only the last **Local Set** `0x53` is useful for us : it's the **Registry Designator for the KLV types in the MXF DCP**.

Note that we have the same behavior with the **Registry Designator** for Variable-Length Pack, but we only have one **Registry Designator** set to `0x04` [3]

STRUCTURE DESIGNATOR

Structure Designator identifies the variant within the **Registry Designator**. According to the standard, it uses as a pseudo major version number ("Major number"), ranging from `0x01` to `0x7F`. Most of the time, this value is `0x01`.

VERSION NUMBER

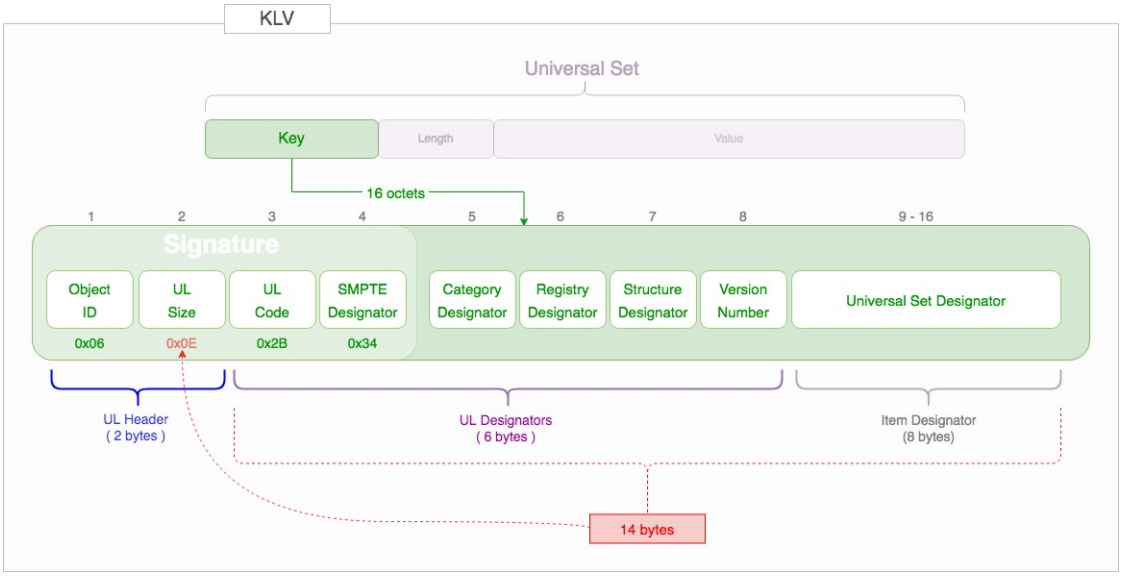
Version Number gives the version of **Item Designator** (the following octets)

ITEM DESIGNATOR

Item Designator (or **Universal-Set Designator**) is the unique identifier for this item within its category. It's on the last 8 octets [4] of our **Universal Label**. We will look at this in more detail in the chapter "KLV Types".

CONCLUSION

Let's take a last look at this diagram of the **Key** :



Now that you know all of this, you can put it away somewhere.

Except for knowing what the **Category** and **Registry** correspond to in order to quickly identify the data type, this will have little — if any — impact on how you handle the data. These essential details will be available in each standard describing the data structure and how to process it :)

Keep in mind that **an identifier** = **Universal Label (UL)** = **a type of data**.

Now let's look at the chapter **KLV Length**.

NOTES

1. We will see later that there are main KLVs, called **Universal Sets** and sub-KLVs - stored within the **Value** part - called **Local Sets** ↩
2. Length is variable, from 1 octet to 4 octets, but in our case, it's only 2 octets. This is determined by the value of the **Registry Designator**. If you remember, we had several **Local Set** with different **Registry Designator** values, such as `0x03` , `0x13` and `0x53` . ↩

   However, this is only a tiny fraction of the possible values. In the documentation, there are many more. Each value represents a **Local Tag** size and a **Length** size for the item.

   In MXF DCP, **we have only the value** `0x53` , which corresponds to a **Length** size and a **Local Tag** size of 2 octets (16 bits) each.

   To your information, Here is a descriptive table of the different values of **Registry Designator** for the other **Local Set**. Only the green will be useful (We won't see the others) :

First 6 octets of Universal Label	Registry Designator (position 6)	Size of Local Tag	Size of Length	Informations
`060e2b34.0203`	`0x03`	1 octet	BER short/long	Any size
`060e2b34.020b`	`0x0b`	BER OID	BER short/long	Any size
`060e2b34.0213`	`0x13`	2 octets	BER short/long	Any size
`060e2b34.021b`	`0x1b`	4 octets	BER short/long	Any size
`060e2b34.0223`	`0x23`	1 octet	1 octet	Size up to 255
`060e2b34.022b`	`0x2b`	BER OID	1 octet	Size up to 255
`060e2b34.0233`	`0x33`	2 octets	1 octet	Size up to 255
`060e2b34.023b`	`0x3b`	4 octets	1 octet	Size up to 255
`060e2b34.0243`	`0x43`	1 octet	2 octets	Size up to 65535
`060e2b34.024b`	`0x4b`	BER OID	2 octets	Size up to 65535
`060e2b34.0253`	`0x53`	2 octets	2 octets	Size up to 65535
`060e2b34.025b`	`0x5b`	4 octets	2 octets	Size up to 65535
`060e2b34.0263`	`0x63`	1 octet	4 octets	Size up to (2^{32})-1 (4.294.967.295)
`060e2b34.026b`	`0x6b`	BER OID	4 octets	Size up to (2^{32})-1
`060e2b34.0273`	`0x73`	2 octets	4 octets	Size up to (2^{32})-1
`060e2b34.027b`	`0x7b`	4 octets	4 octets	Size up to (2^{32})-1

3. Length is variable, from 1 octet to 4 octets, but in our case, it's a BER encoding. This is determined by the value of the **Registry Designator**. ↩

   For your information, here is a descriptive table of the different values of **Registry Designator** for the other **Variable-Length Pack**. Only the green will be useful (We won't see the others)

first 6 octets Universal Label	Registry Designator (6)	Size Length	Informations
060e2b34.0204	0x04	BER short/long	Any size
060e2b34.0224	0x24	1 octet	Size up to 255
060e2b34.0244	0x44	2 octets	Size up to 65535
060e2b34.0264	0x64	4 octets	Size up to (2^{32} - 1)

4. The standard is naughty, it indicates that the size of **Item Designator** may be from 1 octet to 8 octets. If that's the case, it will be padded with zeros to make... 8 octets. To sum up, **Item Designator** is **always** 8 octets long. ↩