KLV : KEY: L'IDENTIFIANT DE TYPE (OU UNIVERSAL LABEL)

La normalisation de la clef est définie dans la documentation **SMPTE 298 : Universal Labels (UL) for Unique Identification of Digital Data**.

En résumé : **C'est une clef unique par type de KLV**, mais pas unique par KLV : vous pouvez avoir plusieurs fois la même clef dans un MXF. Les exemples parfaits sont les KLV pour les frames : vous aurez plusieurs milliers de fois la même clef pour chaque KLV contenant une frame.

Un **Universal Label (UL)** a une certaine forme :

```
Représentation hexadécimale brute = 060e2b34020501010d01020101020400
Représentation au format SMPTE    = 060e2b34.02050101.0d010201.01020400
```

Il en existe énormément dont voici une poignée ci-dessous :

Key / Universal Label (UL)	Type de KLV
`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
...	

Comme on le constate, chaque **Universal Label (UL)** identifie un type de KLV.

On remarquera que l'entête (`060e2b34`) est toujours identique. Puis, après, et suivant le type du KLV, cela change un peu :-)

Après l'entête, tous les octets sont importants car ils sont comme des paramètres, des options ou des statuts du KLV. Par exemple, les **Partitions Packs** possèdent plusieurs identifiants avec une base commune. Ces paramètres permettent de déterminer si c'est un Header, un Body ou un Footer, si ces derniers sont complets ou incomplets, ouverts ou fermés, etc.

On va prendre notre exemple des **Partition Packs** et mettre en avant leurs différences :

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**

Pour le cas des **Partition Pack**, vous remarquerez que seuls leurs avant-avant-derniers octets (en gras) changent, ils représentent respectivement :

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

La valeur juste après (`04`) détermine le statut du **Partition Pack** : Avec `04` , il est considéré comme **Closed & Complete**.

Chaque KLV a ses propres spécificités : Il faudra lire chaque documentation SMPTE pour comprendre et récolter ces précieux paramètres et ainsi identifier des **Universal Labels** pour chaque type de KLV (c'est comme des Pokemons).

Vous pouvez également récupérer un listing des différents **Universal Labels** auprès de ces registres, mais ils ne vous indiqueront pas à quoi correspondent chaque octet :

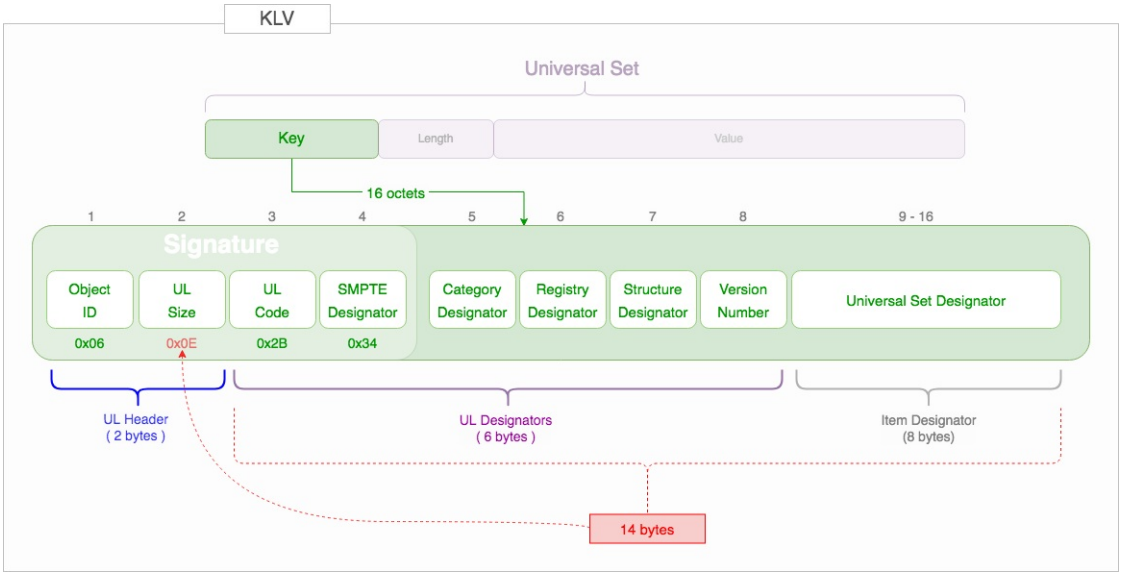
- https://registry.smpte-ra.org/apps/pages/published/
- https://registry.smpte-ra.org/view/published/published_registers/
- https://registry-page.isdcf.com/uls/
- Compilation des SMPTE Universal Labels de la RP224-2003 (attention, peut-être deprecated)

Nous allons voir en détail comment est composé un **Universal Label (UL)**, l'intérieur d'une clef. Notez que savoir cela n'est pas essentiel et ne vous empêchera aucunement l'interprétation des données dans **Value**. C'est un bonus pour votre savoir.

L'INTÉRIEUR DE KEY

Comme nous l'avons vu, après l'entête de la clef (`060e2b34`), chaque octet a une caractéristique propre.

Voici une représentation schématisée de l'intérieur du «**K**» d'un KLV :



La taille d'un **Universal Label (UL)** est de 16 octets et est subdivisible en 3 grandes parties :

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

Ce tableau permet de voir rapidement les différentes parties d'une clef :

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															

Normalisée, les 4 premiers octets débuteront **toujours** par la séquence hexadécimale : `0x06` , `0x0e` , `0x2b` , `0x34` . **C'est sa signature** - et cela permet ainsi d'identifier rapidement ces KLV [1].

Si vous regardez l'**UL Header** - 2 premiers octets : nous avons **Object ID** + **UL Size** qui possèdent un format qui va vous paraître familier : nous débutons avec un identifiant puis une taille et on termine avec d'autres données... Oui, c'est bien le format KLV qui charpente aussi l'intérieur de la clef :-)

Ainsi, le second octet (**UL Size**) est la taille des données de **Key** : `0x0E` représente 14 octets : cela correspond donc aux octets suivants : De **SMPTE Designator** au dernier **Item Designator #8**.

Voici un tableau descriptif de chaque octet de l'**Universal Label (UL)** :

N°	Nom du champ	Description	Données	
1	Object ID	Son identifiant	`0x06`	UL Header
2	UL Size	Taille restante de la clef	`0x0E`	UL Header
3	UL Code	Sous-identifiant ISO + ORG	`0x2B`	UL Designator
4	SMPTE Designator	Sous-identifiant SMPTE	`0x34`	UL Designator
5	Category Designator	Types de données	`0x01` - `0x7E`	UL Designator
6	Registry Designator	Universal Sets	`0x??`	UL Designator
7	Structure Designator		`0x??`	UL Designator
8	Version Number	Version du registre	`0x??`	UL Designator
9-16	Universal-Set Designator	Paramètres & Options	....	Item Designator

CATEGORY DESIGNATOR

Le **Category Designator** permet de connaître rapidement le type de données que nous aurons dans la partie **Value**, pour faire rapide sur les principales catégories :

- **Dictionnaries** sont les données brutes
- **Groups (Sets ou Packs)** sont les données structurées

Voici un tableau récapitulatif des différentes **Category Designator** (5) accompagné du suivant, le **Registry Designator** (6), qui permet d'être plus précis sur le type de données (en vert, les principaux éléments que nous aurons dans un MXF de DCP) :

	Category Designator (5)	Registry Designator (6)	
Dictionnaries	**0x01**	**Données brutes**	
	0x01	0x01	Metadata Dictionary
	0x01	0x02	Essence Dictionary
	0x01	0x03	Control Dictionary
	0x01	0x04	Types Dictionary
Groups (Sets ou Packs)	**0x02**	**Données structurées**	
	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	Reservé / Interdit
Wrappers & Containers	**0x03**		
	0x03	0x01	Simple Wrapper & Container
	0x03	0x02	Complex Wrapper & Container
SMPTE Labels	**0x04**		**Non utilisé comme clef de KLV**
Registered Private Information	**0x05**		
Reserved	**0x06**		
...	**...**		
Reserved	**0x7E**		

REGISTRY DESIGNATOR

Le **Registry Designator** identifie le registre spécifique dans une catégorie spécifique (par exemple, "**Essence Dictionary**" qui est une sous-catégorie dans "**Dictionaries**"). Voir le tableau juste au-dessus pour voir les différents **Registry Designator**.

Vous aurez remarqué la présence de plusieurs **Local Sets** avec trois **Registry Designator** différents. De base dans la norme KLV (2004), **Local Set** a un code de **Registry Designator** à `0x03`. Dans la norme MXF (2011), les différents types de KLV compatibles avec le type **Local Sets** portent les codes de **Registry Designator** `0x53` ou `0x13`.

Le pourquoi de ces différents **Registry Designator** pour **Local Set** ? On va pas trop s'attarder dessus car nous le verrons dans la partie MXF : KLV : Values - Local Sets ainsi que dans les notes [2] mais pour résumer rapidement cela aura un impact sur les données stockées dans la partie **Value**. Seul le dernier **Local Set** `0x53` est utile pour nous : c'est le **Registry Designator pour les types de KLV des MXF DCP**.

A noter que nous avons le même type de comportement avec le **Registry Designator** de Variable-Length Pack mais que nous n'aurons qu'un **Registry Designator** à `0x04` [3]

STRUCTURE DESIGNATOR

Le **Structure Designator** identifie le variant dans le **Registry Designator**. Selon la norme, il est utilisé comme un simili-numéro de version majeur (Major number), allant de `0x01` à `0x7F`. Dans la quasi-majorité, cette valeur sera toujours à `0x01`.

VERSION NUMBER

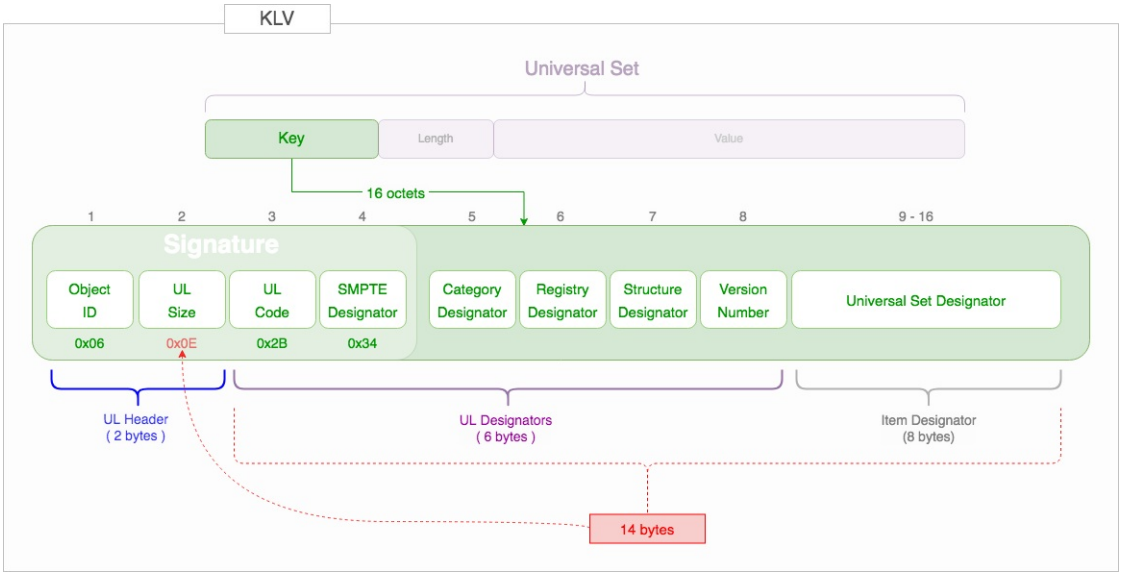
Le **Version Number** donne la version de l'**Item Designator** (donc de la suite)

ITEM DESIGNATOR

Le **Item Designator** (ou **Universal-Set Designator**) est l'identifiant unique pour cet item dans sa catégorie, il est sur les 8 derniers octets [4] de notre **Universal Label**. Nous verrons ce passage plus en détail dans la section "Les types de KLV".

CONCLUSION

Revoyons une dernière fois ce schéma de **Key** :



Maintenant que vous connaissez tout ceci, vous pouvez le ranger quelque part.

Hormis de savoir à quoi correspondent **Category** et **Registry** pour connaitre rapidement le type de données, cela n'aura pas beaucoup d'impact - ou presque - sur votre façon de traiter les données : ces informations essentielles seront disponibles dans chaque norme décrivant la donnée et comment la traiter :)

Gardez juste en tête que **un identifiant** = **Universal Label (UL)** = **un type de données**.

Voyons maintenant la partie **Length d'un KLV**

NOTES

1. Nous verrons par la suite qu'il y a des KLV principaux, les **Universal Sets** et des sous-KLV - placés dans les parties **Value** - appelés **Local Sets** ↩
2. Variable entre 1 octet et 4 octets mais pour notre cas, c'est 2 octets seulement. Cela est déterminé par la valeur du **Registry Designator**. Si vous vous souvenez, nous avions plusieurs **Local Set** avec des valeurs de **Registry Designator** comme `0x03`, `0x13` et `0x53`. ↩

   Cependant, ce n'est qu'une infime partie des valeurs possibles. Dans la documentation, il en existe beaucoup plus. Chaque valeur représente une taille **Local Tag** et une taille de **Length** pour l'item en question.

   Dans les MXF de DCP, **nous n'aurons que la valeur** `0x53` qui correspond à un **Length** et **Local Tag** à 2 octets (16 bits) chacun.

   A titre informatif, voici un tableau récapitulatif des différentes valeurs du **Registry Designator** pour les autres **Local Set**, seul le vert nous sera utile (nous ne verrons jamais les autres) :

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

3. Variable entre 1 octet et 4 octets mais pour notre cas, c'est un BER. Cela est déterminé par la valeur du **Registry Designator**. ↩

   A titre informatif, voici un tableau récapitulatif des différentes valeurs du **Registry Designator** pour les autres **Variable-Length Pack**, seul le vert nous sera utile (nous ne verrons jamais les autres) :

6 premiers octets Universal Label	Registry Designator (6)	Taille Length	Informations
060e2b34.0204	0x04	BER short/long	Toute taille
060e2b34.0224	0x24	1 octet	Taille jusqu'à 255
060e2b34.0244	0x44	2 octets	Taille jusqu'à 65535
060e2b34.0264	0x64	4 octets	Taille jusqu'à (2^{32} - 1)

4. La norme est coquine, elle indique que la taille de **Item Designator** peut être entre 1 à 8 octets. Si c'est le cas, elle sera complétée qu'avec des zéros pour faire ... 8 octets - En gros, **Item Designator** sera **toujours** de 8 octets :) ↩