CPL : ASSETTYPE : GENERIC

PRÉFACE

AssetType **Generic** est le **papa** de tous les assets.

Un exemple de **GenericAssetType** appliqué sur un faux AssetType appelé **Example** :

```
<Example>
    <Id>urn:uuid:3bd3d849-117b-46b0-bc45-3d3228c987c6</Id>   ----.
    <AnnotationText language="en">Example</AnnotationText>       |
    <EditRate>24 1</EditRate>                                    |--- Generic
    <IntrinsicDuration>24</IntrinsicDuration>                    |
    <EntryPoint>0</EntryPoint>                                   |
    <Duration>24</Duration>                                  ----'
</Example>
```

DESCRIPTION

Le **GenericAssetType** intègre des éléments de base que vous trouverez dans **tous** les autres AssetTypes :

Nom	Format	Exemple		Info
Id	UUID URN	`urn:uuid:3bd3d849-117b-46b0-bc45-3d3228c987c6`	**Obligatoire**	
AnnotationText	Texte	`Picture 1`	Optionnel	
EditRate	Rational Number	`24 1`	**Obligatoire**	
IntrinsicDuration	Integer	`14400` (24 images * 60 secondes * 10 minutes)	**Obligatoire**	Minimum: 24 (1 sec)
EntryPoint	Integer	`0` (aucune image pour l'offset)	Optionnel	
Duration	Integer	`14400` (24 images * 60 secondes * 10 minutes)	Optionnel	Minimum: 24 (1 sec)

Rappel : Le champ **Id** n'est pas un identifiant aléatoire, il sera dans cette CPL, mais également dans la PKL et l'AssetMap. Dans le cas de nos assets MXF, l'identifiant se trouve au sein du MXF, dans le KLV Source Package dans le champ **Package UID**, vous trouverez des méthodes rapides dans le chapitre spécifique MXF : Codes & Fichiers.

Nous allons maintenant étudier les différences fondamentales entre **EditRate**, **IntrinsicDuration**, **EntryPoint** et **Duration**.

- **AnnotationText** : déjà vu dans AssetMap ou PKL.
- **EditRate** définit la fréquence des images par secondes. De base, nous sommes à 24 images/secs.
  Voici une liste non exhaustive des différents EditRate rencontrés et de leurs cas d'usages probables :

EditRate	Cas d'usage probable
24 1	24 fps (cas le plus courant)
25 1	25 fps
30 1	30 fps
48 1	HFR 48 fps et/ou 3D 24 fps
50 1	HFR 50 fps et/ou 3D 25 fps
60 1	HFR 60 fps et/ou 3D 30 fps
96 1	HFR 96 fps et/ou 3D 48 fps
100 1	HFR 100 fps et/ou 3D 50 fps
120 1	HFR 120 fps et/ou 3D 60 fps - cas rare, exemple avec GeminiMan en 120 fps natif
Hors scope	
24000 1001	c'est du 23.97 fps - cas rare, ne pas reproduire, c'est dégueulasse
30000 1001	c'est du 29.97 fps - cas encore plus rare, même conclusion que dessus.

Notez que si vous étudiez le format IMF, vous pouvez tomber sur des EditRates ou des FrameRates encore plus bas comme 16/1, 200/11 (18.18), 20/1 ou encore 240/11 (21.81).

Pourquoi HFR et/ou 3D ?

Il est parfois difficile d'identifier le cas d'usage en regardant seulement l'**EditRate**. Il vous faudra également regarder du côté du paramètre **FrameRate** que nous verrons plus tard.

Par exemple, 100 fps peut être utilisé soit pour faire du HFR - auquel cas, nous aurons 100 images par seconde, soit de la 3D, auquel cas, il y aura 50 images par oeil.

Il est également possible de faire du HFR et 3D en même temps, prenons le cas `120 1` : si c'est de la 3D, nous avons donc du 60 images par oeil et le 60 fps fait partie du HFR.

Il existe même un cas de figure très particulier, par exemple pour faire du 120 fps en 3D-HFR avec des DCP 2D : Pour cela, on fait un DCP avec un EditRate de 120 fps mais on en fait 2 : un pour chaque oeil et un pour chaque projecteur. Cette méthode a été utilisée sur la projection de GeminiMan en véritable 120 FPS HFR-3D (donc 120 images par oeil).

L'**EditRate** et le FrameRate travaillent de pair pour définir le type de projection :

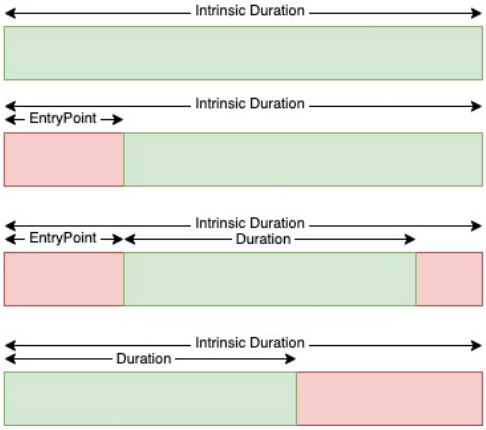
EditRate	FrameRate	
24	24	Film projeté en 24 fps
48	48	Film projeté en 48 fps
24	48	Film en 24 fps mais projeté en 48 fps, cas d'usage : la 3D (24 images par oeil)
48	96	Film en 48 fps mais projeté en 96 fps, cas d'usage : la 3D (48 images par oeil)

Tous les **EditRate** d'une CPL doivent être identiques. Il n'est donc pas possible de faire une Reel en 24, puis une Reel en 48, etc... (adieu VFR)

- **IntrinsicDuration** est la durée totale de l'asset en nombre d'images : **c'est la durée réelle contenue dans le MXF** (mais pas forcément la durée utilisée en playback)
- **EntryPoint** est l'endroit où débute l'asset lors de la lecture en nombre d'images. Vous pouvez avoir un asset qui peut être plus long que les autres assets, il est donc possible de faire soit démarrer à un certain point, soit l'arrêter à un autre (avec Duration)
- **Duration** est la durée réelle du playback de l'asset en nombre d'images. C'est elle qui détermine ce qui va être réellement projeté.

En voyant **IntrinsicDuration**, **EntryPoint** et **Duration**, vous pourriez être perdu, et c'est normal : Ces trois paramètres permettent de faire un semblant de montage avec les différents MXF. Selon les paramètres spécifiés, vous pouvez démarrer un MXF a un certain endroit ou l'arrêter à un autre.

Voici les différents cas de figure en image pour mieux comprendre les implications de chacun :



En vert est la zone de l'asset qui va être joué.

De base, le player va lire **IntrinsicDuration**, il sait que l'asset va durer une certaine durée. S'il n'existe ni **EntryPoint** ni **Duration**, il considère qu'il doit lire la totalité de l'asset : de la première frame à la dernière frame de l'asset.

S'il existe un **EntryPoint**, le player va démarrer à un offset particulier indiqué dans **EntryPoint**.

- S'il n'existe aucune **Duration**, il considère qu'il doit lire la totalité du reste de l'asset.
- S'il existe une **Duration**, le player va donc jouer à partir de l'**EntryPoint** et jusqu'à la fin de la **Duration**.

En combinant **EntryPoint** et **Duration**, le player peut donc lire qu'une toute petite partie de l'asset.

Dans la grande majorité des cas, la **Duration** est égale à **IntrinsicDuration** et l'**EntryPoint** est à 0.

Toutes les **Duration**s d'une **Reel** doivent être égales (à l'exception des éléments de support textuel (sous-titres par exemple)). Et la

Duration de référence est déterminée par le **MainPicture** (ou **MainStereoscopicPicture** si présent)

CONCLUSION

Pour résumer, **GenericAssetType** est le parent de tous les assets types, les autres asset types seront tous ses enfants et auront donc une partie ou la totalité des éléments du **GenericAssetType**.

CHAPITRES ANNEXES

- AssetType Generic : Le parent de tous ⬅ vous êtes ici
  - AssetType TrackFile : Le parent de Picture, Sound, Subtitle, Caption et Data.
    - AssetType PictureTrackFile : MainPicture et MainStereoscopicPicture
    - AssetType SoundTrackFile : MainSound
    - AssetType SubtitleTrackFile : MainSubtitle et ClosedSubtitle
    - AssetType CaptionTrackFile : MainCaption et ClosedCaption
    - AssetType DataTrackFile : AuxData
  - AssetType Marker : MainMarkers
  - AssetType CompositionMetadata : CompositionMetadaAsset