ASC Framing Decision List (ASC FDL)

Advanced Data Management Subcommittee

THE AMERICAN SOCIETY OF CINEMATOGRAPHERS

Specification v0.0

01. May, 2022

Table of Contents

1. Introduction

ASC FDL's are a set of instructions for how to view media in any application. The ASC FDL provides a mechanism to document framing decisions through all phases of a project's life cycle, from pre-visualization through post-production. The FDL can exist in the form of a sidecar JSON file, or embedded into another data structure like camera original files. Any time an application is rendering media to go to another department or person, an accompanying set of ASC FDL data should be created to inform how to view the newly generated content. This ASC FDL data can be applied to view the intended framing.

2. Scope

This document specifies format definitions and operations for the ASC Framing Decision List (ASC FDL), for the exchange and creation of framing data. This document also contains the information required to implement an "ASC FDL-compliant" software/hardware system.

The ASC FDL is intended for use in media production workflows and has been optimized to support the ability for a department or person to be able to easily recreate framing made by others, while limiting human error and the traditional operational burden. A common example is when on-set framing decisions have been made by an Image Author that then need to be conveyed to the vendor processing material for editing and review. Often, the processes used for re-creating that on-set framing decision are manual and prone to human error.

While ASC FDL is able to track framing decisions per shot, tracking per frame is currently out of scope.

3. Conformance Notation

Normative text is text that describes elements of the design that are indispensable or contains the conformance language keywords: "shall", "should", or "may". Informative text is text that is potentially helpful to the user, but not indispensable, and can be removed, changed, or added editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except: the Introduction, any section explicitly labeled as "Informative" or individual paragraphs that start with "Note:"

The keywords "shall" and "shall not" indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keywords, "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document. The keyword “reserved” indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword “forbidden” indicates “reserved” and in addition indicates that the provision will never be defined in the future.

The following font/font color formatting is used throughout this document:

- JSON formatting has been used to show examples of ASC FDL code
- Purple is used for JSON elements, attributes and values, e.g. `aspect ratio`

4. References

- JavaScript Object Notation (JSON)
- SMPTE ST 2114:2017, Unique Digital Media Identifier (C4 ID)
- Unicode Transformation Format 8-bit (UTF-8)

5. Concepts and Semantics

5.1 ASC FDL

An ASC FDL is a self contained file with an .fdl extension. The file is formatted utilizing JSON and does not require a specific directory structure. FDL data can also be written into camera files, or shared within other file types that support it.

5.2 Locating ASC FDL Files

Requiring specific folder structures, file names or how an application may locate an FDL is not in scope for this specification. It is by design that FDL files can be managed in any directory structure a user may wish, along with any filename the user wishes. It is up to each implementation how they would like the user to select any FDL to be used on any shot.

5.3 Applying ASC FDL Files

ASC FDL's are not intended to be a set of render instructions, but rather instructions for how to view media within an application. It is recommended that any time an application is applying an ASC FDL to media, the Canvas Dimensions should be compared against the source materials resolution. If these do not match, the user should be warned that they are trying to apply an FDL that does not match the Canvas Dimensions resolution.

6. ASC FDL File Properties

6.1 Schema

An ASC FDL Manifest File is a JSON document. The namespace prefixes used in JSON Schema definitions herein are not normative values and implementations shall perform correctly with any JSON compliant prefix values.

6.2 Character Encoding

ASC FDL Manifests shall be encoded using UTF-8 character encoding.

6.3 Naming of the ASC FDL Files

Any implementation when creating ASC FDL's, shall only utilize a limited set of characters including the numbers 0-9, letters a-z and the following special characters: "-", "_", "+", ".". Spaces shall not be permitted within the name of an ASC FDL.

Here is an example set of characters that shall not be permitted @ # $ % ^ & * ( ) ` ; : < > ? , [ ] { } ' " | / \ ~

7. Classes

Description:

ASC FDL files are organized in various sections, each containing its own set of attributes. The ASC FDL classes consist of: Header, Framing Intent, Context, Canvas, Framing Decision, Canvas Template.

7.1 Header

```
"uuid": "BCD142EB-3BAA-4EA8-ADD8-A46AE8DC4D97",
 "version": {"major": 0, "minor": 1},
 "fdl_creator": "ASC FDL Committee",
```

7.1.1 UUID

```
"uuid": "BCD142EB-3BAA-4EA8-ADD8-A46AE8DC4D97",
```

Description:

The UUID field is a globally unique string that differentiates an individual ASC FDL file from any other. The format of the uuid must use the canonical textual representation. The 16 octets of a UUID are represented as 32 hexadecimal (base-16) digits, displayed in 5 groups separated by hyphens, in the form 8-4-4-4-12 for a total of 36 characters (32 alphanumeric characters and 4 hyphens). As an example:

afe122be-59d3-4360-ad69-33c10108fa7a

The UUID attribute is one of two fields used to determine which FDL should be applied to which shot: asc_fdl>uuid and framing_decision>id.

The ASC FDL UUID attribute will utilize the ISO/IEC 9834-8:2008 specification, enabling applications to produce128-bit identifiers that are either guaranteed to be globally unique, or are globally unique with a high probability. For more information please see: ISO/IEC 9834-8:2008

7.1.2 Version

```
"version": {"major": 0, "minor": 1},
```

Description:

The ASC FDL specification may be updated over a course of time and each updated version officially released will have a version number. All ASC FDL files should contain the implementer's ASC FDL version within the header of the ASC FDL file.

Data Type: Int,Int

Required Field: Yes

7.1.3 FDL Creator

```
"fdl_creator": "ASC FDL Committee",
```

Description:

This field can take a string indicating who created the FDL document. A user or implementation may choose to include the user or the software name/version that was used to create it. If any software receives an FDL and adds data to it, the fdl_creator field should represent the most recent modifying author when creating a new version of the FDL.

Data Type: String

Required Field: No

Default Value: Blank

7.2 Framing Intents

```
"framing_intents": [{
      "label": "2.39:1 Framing",
      "id": "FDLSMP01",
      "aspect_ratio": {"width": 239, "height": 100},
      "is_primary": true,
      "protection": 0.05
    }
```

Description:

Creating a framing intent is the first key step to creating an FDL. It represents the intended aspect ratio, unbounded by the constraints of any camera or device. As an example: 2.39:1. This is the region within which a cinematographer will compose content intended for the viewing audience. An FDL may contain multiple framing intents.

Required child elements:

7.2.1 `label`:

The label field in an FDL is a human entered field, with a 16 character maximum length, for any user to have a title for their Framing Intent. This field may be useful for any user that has multiple framing intents within a single FDL. An implementation may choose to show this field in their interface for allowing a user to choose which framing intent from the selected FDL they would like to apply.

The label field shall only utilize a limited set of characters, please see Appendix G for details.

Data Type: String

Required Field: No

7.2.2 id:

The Framing Intent id field is meant to provide a means of identification for a framing intent. This id is not universally unique, but no other framing intent id within a single FDL will use the same id. The field will have 8 alphanumeric characters that can be specified by the implementing software, as long as it is unique within the ASC FDL.

Example: `FDLSMP01`

Data Type: String

Required Field: Yes

7.2.3 aspect_ratio:

The aspect ratio field represents the image author's original intention. An aspect ratio will be displayed with the width and height, separated by a colon. (Width:Height). As an example: 220:100 or 2048:858.

Data Type: int:int

Required Field: Yes

7.2.4 is_primary:

It is possible that a production will require more than one framing intent within an FDL. The ASC FDL includes an "is_primary" attribute to identify which framing intent should be applied by default. Within an individual ASC FDL only one primary framing intent shall be permitted. If the ASC FDL only contains one framing intent, the default value will be True.

Any implementation modifying an already existing ASC FDL will check if another primary `framing_intent` exists. If it does and the user has not specified which should be the primary, it will populate any newly added framing intent's is_primary field as false. Any ASC FDL Implementation will prompt an error message when attempting to create an FDL with more than one primary framing intent.

Data Type: Boolean

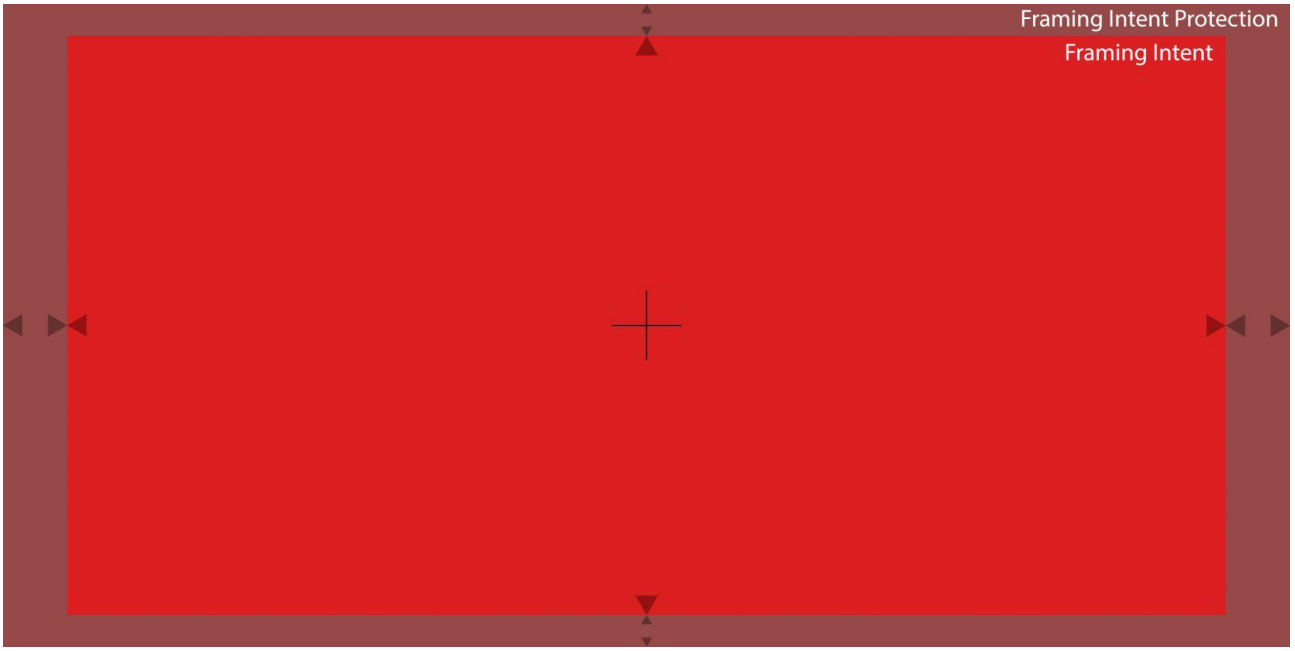
Allowed Values: True, False

Required Field: Yes

7.2.5 protection:

A framing intent may include an area for protection. This area is called Framing Intent Protection and matches the aspect ratio of the framing intent.

A protection value of 0 means there will be no Framing Intent Protection. A protection value of 0.05 results in a 5% protection area outside of the framing intent.



The default value for the protection will be 0.

The `protection` aspect ratio will always match the aspect ratio of the associated `framing_intent`.

Unless manually overridden by the user, implementations should default to center the protection to the associated framing decision.

Data Type: Float

Required Field: Yes

Default Value: 0

7.3 Contexts

```
"contexts": [{
    "label": "ArriLF OG",
    "content_creator": "ASC FDL Committee",
```

Description:

The Context class provides the ability for image authors to provide additional information on the origin of the ASC FDL. Context attributes will not be referenced by any other function within the ASC FDL. The field's purpose is for users to manage and organize their FDL data. As an example, an image author may choose to have framing data specific to a certain camera, editorial delivery, visual effects plates, etc.

Required child elements:

7.3.1 label:

The label field is available for anyone to categorize/manage their FDL data. This is a field that would be manually entered by the user generating the FDL. As an example: ArriLF OG. It will have a 16 character maximum limit and shall utilize the character encoding parameters as described in section 7.3 of this specification.

Data Type: String

Required Field: No

7.3.2 context_creator:

The context creator attribute will be populated by the application that has generated the FDL. This field represents which user or implementation has generated this specific context within the ASC FDL. There could be different context_creator data in each context if multiple authors contributed to an ASC FDL. It is up to the implementation as to how it formats this attribute's values. This field shall utilize the character encoding parameters as described in section 8.3 of this specification.

Data Type: String

Required Field: Yes

7.4 Canvases

```
"canvases": [{
      "label": "Open Gate RAW",
      "id": "ArriLFOG-20220310",
      "source_canvas_id": "ArriLFOG-34256345",
      "dimensions": {"width": 4448, "height": 3096},
      "effective_dimensions": {"width": 4006, "height": 2788},
      "effective_anchor_point": {"x": 222, "y": 155},
      "photosite_resolution": {"width": 4448, "height": 3096},
      "physical_dimensions": {"width": 36.7, "height": 25.54},
      "anamorphic_squeeze": 1.0,
```

Description:

This region defines the active coordinate system of an application, file, or video stream. An application, file, or video stream could contain additional area outside of the defined canvas, but applying an FDL to utilize that region would require a different canvas. As an example, if a camera recorded a file, the file's recorded resolution will be utilized as the canvas. If you then record a second file in a different resolution, you will have a new canvas. A canvas can only be generated once the system creating it understands what the recorded/generated

area is going to be.

Required child elements:

7.4.1 label:

Each canvas within an ASC FDL allows users to enter a human written label. As an example: `Open Gate RAW`. The canvas label field has a 32 character limitation. This field shall utilize the character encoding parameters as described in section 8.3 of this specification.

Data Type: String

Required Field: Yes

7.4.2 id:

The id is unique to each `canvas` inside of a given FDL, but may not be globally unique to other FDL files. The canvas id is a system generated value made up from the context label, a dash and a newly generated 8 digits (numerical). Therefore: `[context>label][-][8 digits]`

Example: `ArriLFOG-34256345`

The id field will not contain any spaces. Therefore if the context label had spaces, any application that complies with the ASC FDL specification shall remove these spaces within the canvas id field.

Data Type: String

Required Field: Yes

7.4.3 source_canvas_id:

ASC FDL's may be generated from original camera files, or derivatives from the original camera files. Therefore the source canvas id attribute has been created to allow a user to see the canvas that was used when the original ASC FDL was created. As an example, a user may have an ASC FDL for the source camera file. They may then render plates to be delivered to a VFX vendor with a smaller resolution. Along with these plates, a new FDL may be delivered. The new ASC FDL's canvas id will represent the new canvas (VFX plates) that were generated. However the source canvas id would reference the initial FDL.

If there is no prior knowledge of a past generation `canvas`, the `source canvas id` should be the same value as `canvas id`.

Data Type: String

Required Field: Yes

7.4.4 dimensions:

Any canvas within an ASC FDL will have a width and height defined. The dimensions field will be formatted as: `"width": 4448, "height": 3096`

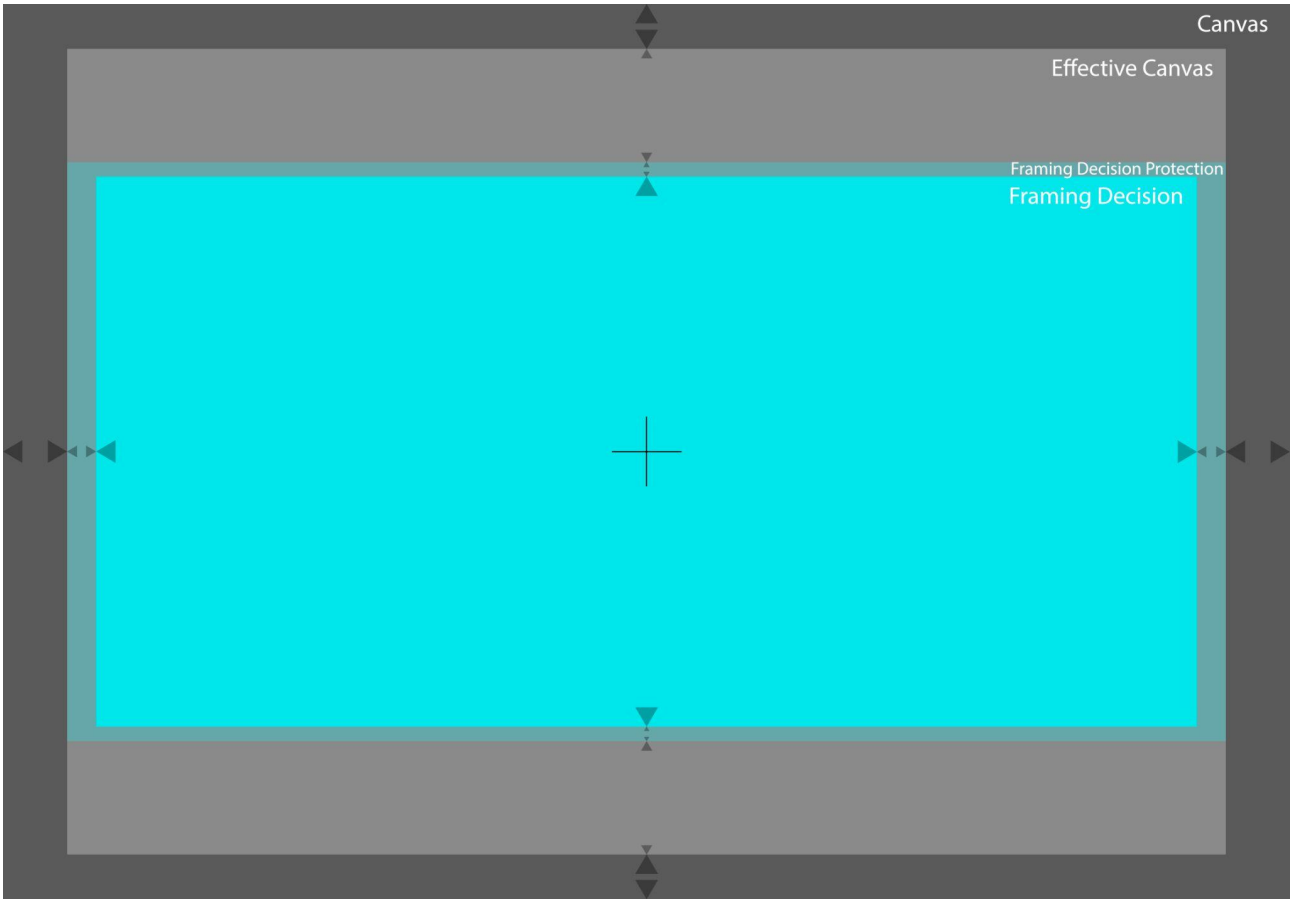
This dimensions characters shall be limited to only digits (numerical)

Data Type: dimensions_int

Required Field: Yes

7.4.5 effective dimensions:

A Canvas can be effectively constrained to prevent a Framing Intent and its Framing Intent Protection from being applied outside an intended area. This is called an effective canvas. As an example, when vignetting results from a lens that doesn't cover your camera's sensor, a user may choose to constrain the usable canvas to the region not affected by vignetting. When the user applies the framing decision within, it will not have the vignetting in frame.



The effective dimensions attribute will define the width and height of this canvas constraint and will be written as: `width": 4006, "height": 2788`

Data Type: width:int, height:int

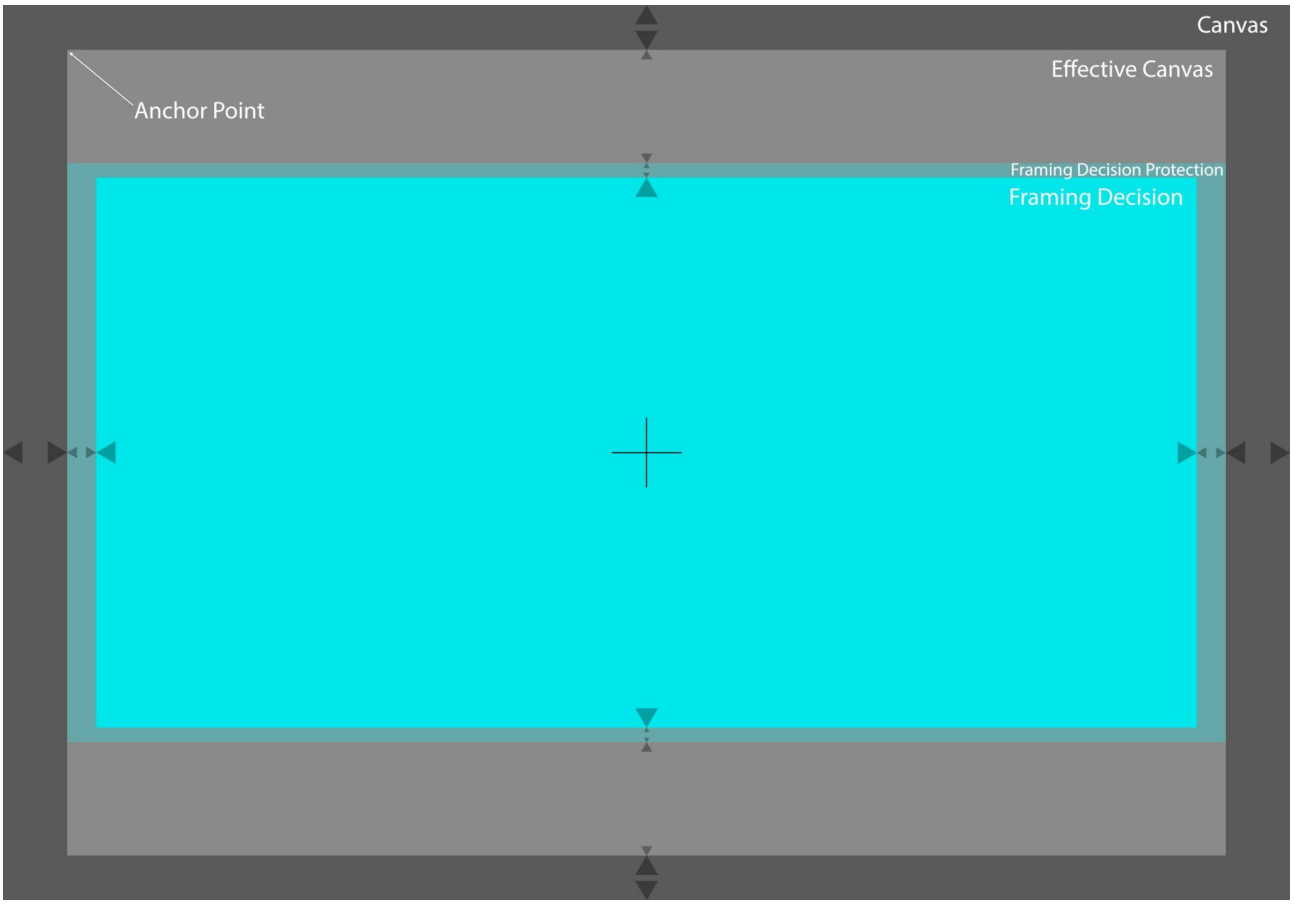
Units: Pixels

Required Field: No

7.4.6 effective_anchor_point:

If an effective canvas is to be placed within a canvas, any implementation will require an understanding of the area to be used and where to position this area within the canvas. The effective anchor point documents where the top left pixel of the effective canvas should be inside the canvas. Similarly to the framing decision anchor points, we're using float versus int values for these variables to avoid rounding issues when scaling.

The effective anchor point will be written as an example
`"x": 222, "y": 155}`



Data Type: x:float, y:float

Required Field: Only required if there is an effective_dimension

Units: Pixels

7.4.7 photosite_resolution:

We encourage camera manufacturers to provide this data when a camera has generated an ASC FDL canvas. We do not require any implementation to generate photosite dimensions for non physical camera generated canvases. As an example, if a camera generated a canvas it would ideally capture this attribute. However if an ASC FDL was generated with a new canvas for a VFX Plate, this attribute would not be expected to be filled. Therefore, a source canvas should have photosite resolution, but subsequent child canvases downstream would likely not.

As an example;
Arri Alexa Mini - Recording Mode: 4K UHD
Sensor Active Image Area:
3200 x 1800 photosites
Recording File Image Content:
3840 x 2160 pixels

Data Type: dimensions_int

Required Field: Optional

Units: Photosites

Default Value: **Blank**

7.4.8 physical_dimensions:

We encourage camera manufacturers to provide this data when a camera has generated an ASC FDL canvas. We do not require any implementation to generate physical dimensions for non physical camera generated canvases. As an example, if a camera generated a source canvas it would ideally capture this attribute. However if an ASC FDL was generated with a new canvas for a VFX Plate, this attribute is not mandatory. Any implementation that is reading an FDL that originally contained physical dimensions, could now generate new physical dimension values. The physical dimensions will use at least 4 decimal places to ensure accuracy to one tenth of a micron.

we should also provide an example

Data Type: float, float

Required Field: Optional

Unit Type: Millimeters

Default Value: **Blank**

7.4.9 anamorphic_squeeze:

Any application reading an FDL will need to understand if the canvas it is reading is squeezed or not. The anamorphic_squeeze attribute will match the image deformation numbering system that lens manufacturers use. As an

example, 1.3 would mean the image is currently squeezed by a ratio of 1.3:1. Or 2.0 would indicate the image has been squeezed by a factor of 2:1. The squeeze is specifically a horizontal squeeze factor.

All applications reading an ASC FDL will apply the squeeze factor before any scaling, to ensure consistency between applications.

Data Type: float

Required Field: Yes

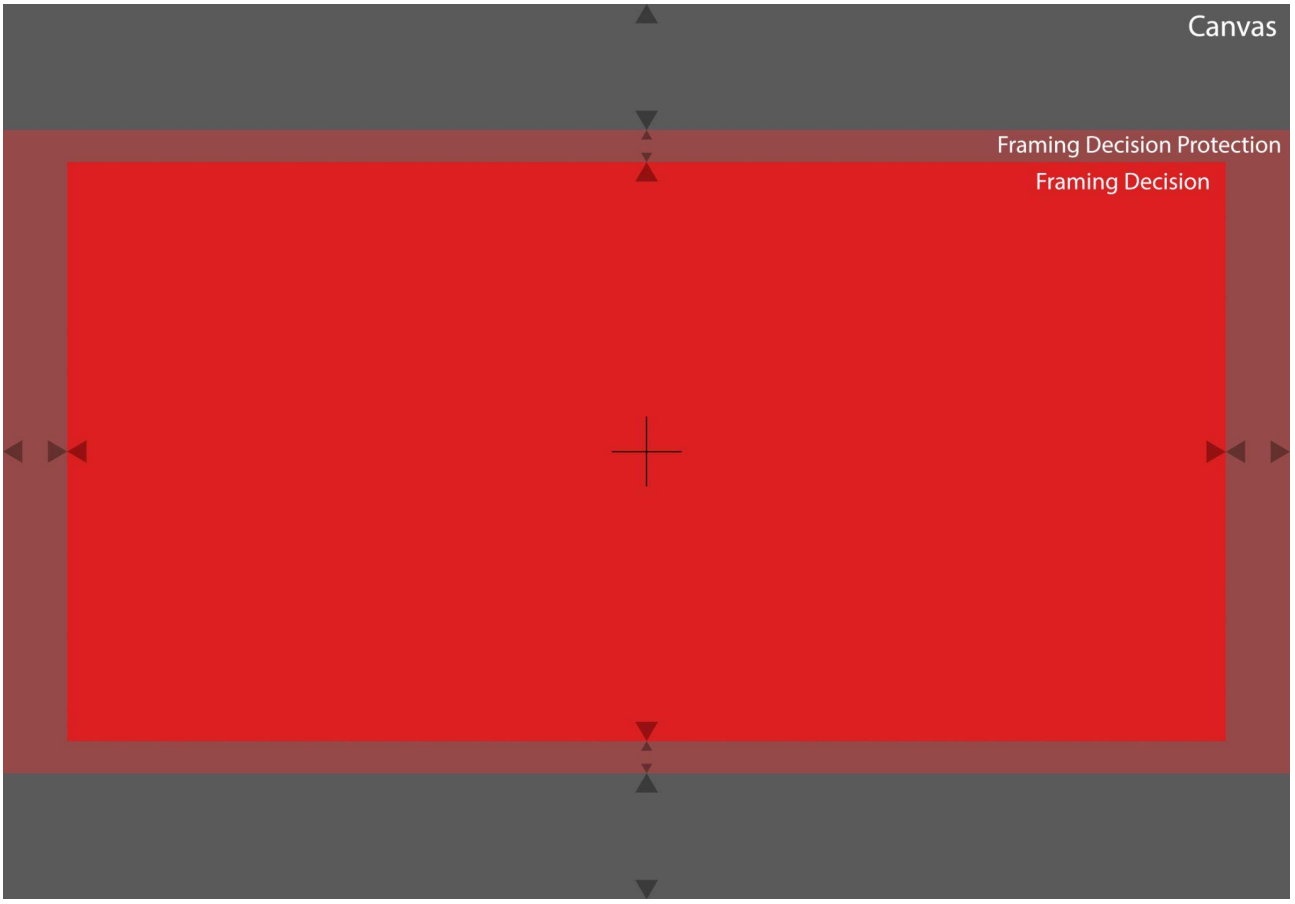
Default Value: **1.0**

7.5 Framing Decisions


```
"framing_decisions": [{
          "id": "ArriLF-20220310-FDLSMP01",
          "framing_intent_id": "FDLSMP01",
          "framing_intent_label": "2.39:1 Framing",
          "anchor_point": {"x": 222.00, "y": 547.00},
          "dimensions": {"width": 4004, "height": 2002},
          "protection_dimensions": {"width": 4448, "height": 2224},
          "protection_anchor_point": {"x": 0, "y": 436}
```


Description:

When the initial framing intent was created, it did not have any attributes that documented its position within a canvas, nor anything that defined its actual size. It was just a defined aspect ratio. When a Framing Intent and its Framing Intent Protection are now applied to a Canvas**,** two decisions are produced: a framing decision and a framing decision protection. The framing decision which is connected to a specific canvas will have a defined set of coordinates, so any application reading an ASC FDL can understand where the intended frame resides.



Required child elements:

7.5.1 id:

Each framing decision will have its own id field that will be unique to the ASC FDL, but not universally unique among other ASC FDL's.

The formatting of the framing decision id will be:

`[canvas>id] [-] [framing_intent>id]`

As an example: `ArriLFOG-20220310-FDLSMP01`

Data Type: String

Required Field: Yes

7.5.2 framing_intent_id:

Including the `framing_intent_id` inside the `framing_decision` class is intended to allow for any implementation to infer which `framing_intent` this `framing_decision` is connected to.

Data Type: String

Required Field: Yes

7.5.3 framing_intent_label:

Including the framing_intent_label inside the framing_decision class is intended to allow for any human to infer which framing_intent this framing_decision is connected to.

Data Type: String

Required Field: Yes

7.5.4 anchor_point:

If an application is given a dimension for the framing decision in pixels, it still needs to understand where to position this within the canvas. The framing decision anchor point specifies where the top left pixel of the framing decision is, in relation to the top left of the canvas, or the canvas protection if one was used.

The anchor point uses float versus int values to avoid scaling ambiguities and rounding issues when scaling.

The formatting of the framing decisions anchor point will be as follows: x-axis value, y-axis value.

As an example: `"x": 222.00, "y": 547.00`

The first value is the number of pixels horizontally from the left side of the canvas, unless an effective canvas has been used. If an effective canvas has been used, the anchor point would count from the left side of the effective canvas.

The second value uses the same process, but now for the y-axis.

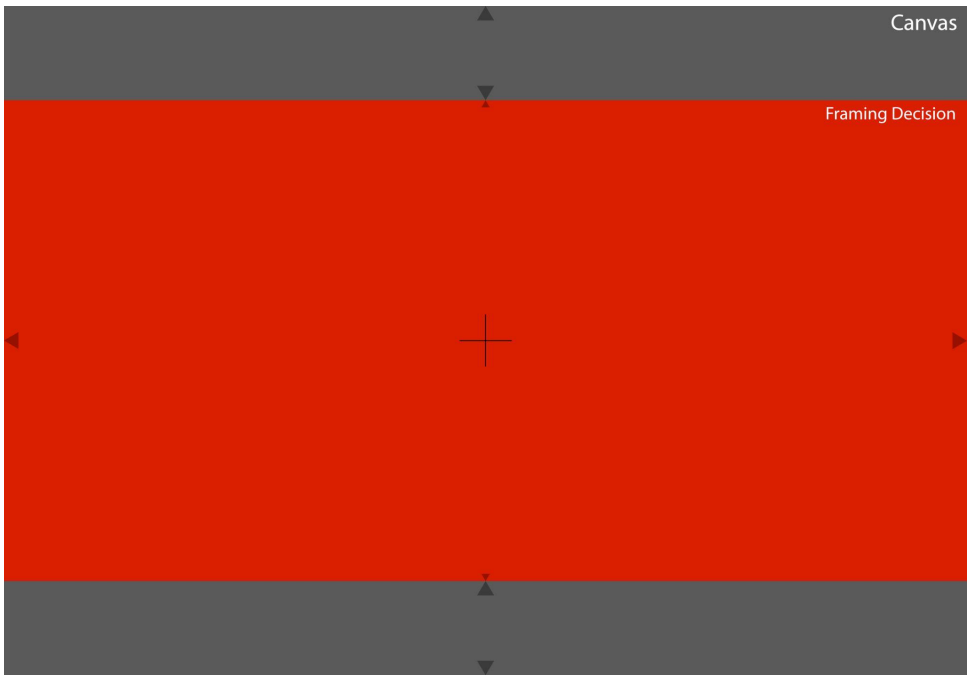
Data Type: x:float, y:float

Required Field: Yes

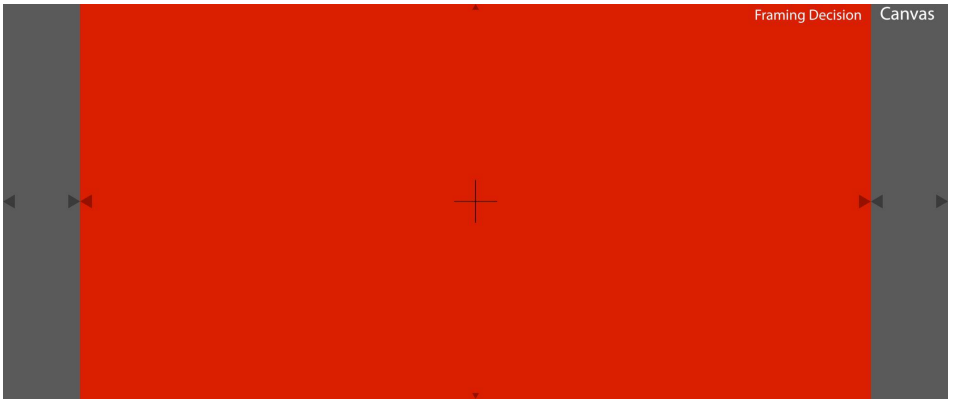
Units: Pixels

7.5.5 dimensions:

The framing decision dimensions will specify the width and height of the framing decision now that it has been placed within a canvas. When generating an ASC FDL, any implementation shall by default place the framing decision to fit within the canvas, not cropping any of the resulting framing decision. As an example, if the canvas had an aspect ratio of 1.43, and the framing intent was 2.0, the resulting framing decision would letterbox the canvas:



Alternatively, if the canvas had an aspect ratio of 2.39, it would be pillarboxed:



The dimensions shall be formatted as float number values, for higher scaling precision. When the float values need to become integers for display purposes in any implementation, we recommend the nearest integer value.

When a source canvas has been scaled to a new dimension, it is possible that the framing decision dimension will not match the framing intent perfectly. When this new FDL is read by any implementation, the framing decision dimension should take precedence.

The formatting of this data will be: “width”: value, ”height”: value

As an example:

```
"width": 4004, "height": 2002
```

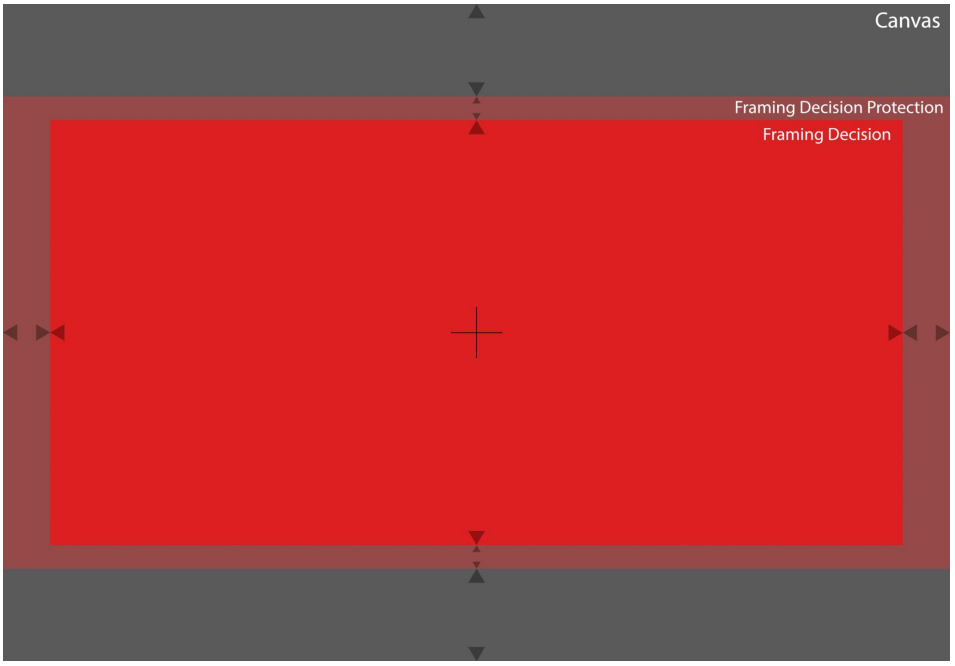
Data Type: number, number

Required Field: Yes

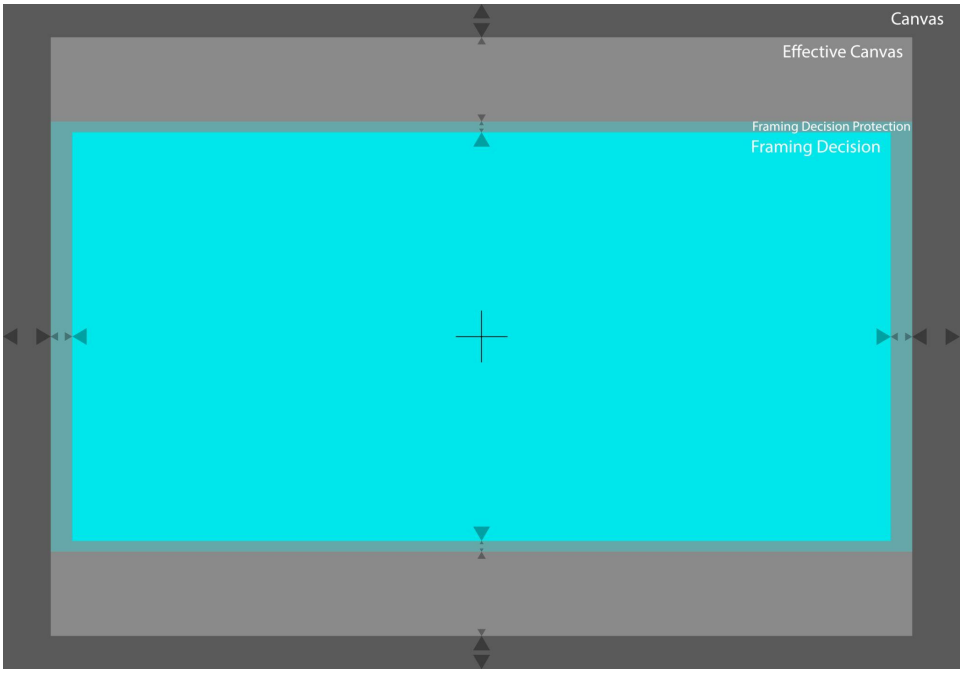
7.5.6 protection_dimensions:

Similarly to the framing Intent resulting in a framing decision once placed within a canvas, the framing intent protection will result in a framing decision protection once placed within a canvas. This area may be utilized as a safety in post production to allow for slight reframing, stabilization and more.

If a framing decision had a protection, the protection would be fit into the canvas:



If an effective canvas exists, the framing decision protection would be placed within it:



In the event that a user has chosen a framing decision dimension that conflicts with an existing framing intent protection, it is permissible to override the framing intent protection to utilize the framing decision dimension.

The framing decision protection dimension allows float number values for higher scaling precision.

When the float values need to become integers for display purposes in any implementation, we recommend the nearest integer value.

The formatting of this data will be: “width”: value, “height”: value

As an example:
`"width": 4448, "height": 2224`

Data Type: number, number

Unit Type: Pixels

Required Field: Yes

7.5.7 protection_anchor_point:

If a protection dimension is utilized, any implementation will require an understanding of the area within the canvas to be used. Even if an application is given a dimension in pixels for this area, it still needs to understand where to position it within the canvas. Therefore the framing decision protection anchor point specifies where the top left pixel of the framing decision protection is, in relation to the top left of the canvas, or effective canvas if one was used.

The framing dimension protection anchor point will use pixels as the unit type. If fractional pixels are calculated, rounding will need to occur to ensure any ASC FDL created has whole numbers in this framing decision protection anchor point.

The `protection anchor point` originates from the top left of the `canvas` coordinate system (0,0).

The first value is the number of pixels horizontally from the left of the `canvas>dimensions` (or effective canvas dimensions if one was used) to the edge of the `framing decision>protection dimensions` (x-axis).

The second value is the number of pixels vertically from the top of the `canvas>dimensions` (or effective canvas dimensions if one was used) to the top of the `framing decision>protection dimensions` (y-axis).

Data Type: x:float, y:float

Required Field: Yes

Unit Type: Pixels

7.6 Canvas Template

```
"canvas_templates": [{
      "label": "VFX Pull",
      "id": "VX220310",
      "target_dimension": {"width": 3840, "height": 2160},
      "target_anamorphic_squeeze": 1.00,
      "fit_source": "framing_decision.dimensions",
      "fit_method": "width",
      "alignment_method_vertical": "center",
```

```
        "alignment_method_horizontal": "center",
        "alignment_offset": {"x": 0, "y": 0},
        "preserve_from_source_canvas": "canvas.dimensions",
        "round": {"even": true, "mode": "up"}
        "maximum_dimension": {"width": 5000, "height": 3496},
        "pad_to_maximum": "true",
```

Description:

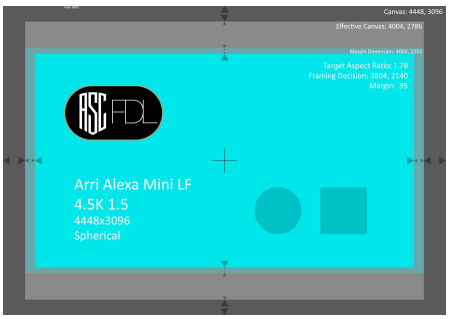
The canvas template is an entire section (class) within an ASC FDL that provides the ability for users to normalize framing across all camera formats that may have been used during a production.

The canvas template provides a set of framing instructions that map source canvases into newly defined canvases. As an example, if various cameras and resolutions were captured during a production, the VFX Supervisor or Picture Finishing Facility may want to ensure that all plates generated for VFX work are normalized into a common container before going to vendors. However they do not know all of the various camera formats that have been captured on set. The user could create a canvas template within an ASC FDL and provide instructions for any application to place the sources into this newly defined canvas.

As an example:

Source 01

Source 02

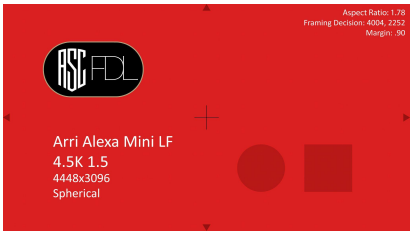


Source 03

Result 01

Result 02

Result 03



An ASC FDL file does not require a Canvas Template. Many workflows may never utilize this enhanced feature. However this functionality has been added to the specification for sophisticated jobs looking to normalize framing.

An ASC FDL with a canvas template may, or may not contain source FDL data. However, if an ASC FDL contains the source file's FDL data (canvas, framing decision, etc) this data can be used for the implementation's decision making on normalizing framing.

For ASC FDL files that do not contain the source files framing decisions, implementations will need to look for ASC FDL data for each source file they intend to use the canvas template on.

Any required attribute within the canvas template is only a requirement if the canvas template class exists within the FDL.

7.6.1 Applying Canvas Templates

For a canvas template to be used by any application, the application would need to know which source FDL data (canvas, framing decision) it should be utilizing. It is not within scope of this specification to mandate how an implementation will request a user to point the application to specific FDLs for this source data.

Required child elements:

7.6.2 `label`:

A human entered label can be added to the Canvas Template. As an example “VFX Pull”. The canvas template label field has a 32 character limitation. This field shall utilize the character encoding parameters as described in section 8.3 of this specification.

Data Type: String

Required Field: Optional

7.6.3 `id`:

Each Canvas Template will have an id for tracking and identification purposes. This id is unique within the FDL, but not universally unique among other ASC FDL’s.

The id will have 8 characters and will only utilize alphanumeric values.

The id data will be formatted as follows:

```
"id": "value",
```

As an example:

```
"id": "VX220310",
```

Data Type: String

Required Field: Yes

7.6.4 `target_dimension`:

For a Canvas Template to be utilized, the user will choose a specific set of dimensions. Using the fit source attribute they can specify what specifically to fit into this target dimension. As an example, a user may choose a target dimension of:

`{"width": 3840, "height": 2160}` and a fit source of Framing Decision. That would mean the user wants to fit the framing decision into the target dimension of 3840:2160.

The target_dimension data will be formatted as follows:

```
"target_dimension": {"width":value, "height": value},
```

As an example:

```
"target_dimension": {"width": 3840, "height": 2160},
```

For an ASC FDL to utilize a canvas template, this field must be populated. If it is left blank, any application reading the ASC FDL should flag this field as missing and the canvas template cannot be used.

Data Type: int,int

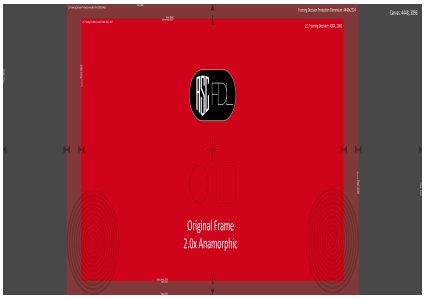
Required Field: Yes

Unit Type: Pixels

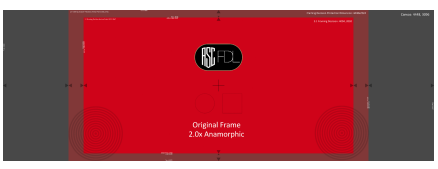
7.6.5 `target_anamorphic_squeeze`:

The target anamorphic squeeze field represents the squeeze factor we are forcing on any incoming media when placed into this new canvas. As an example, if 1.0 is written as the target anamorphic squeeze; that means the user wants everything to be normalized to a squeeze factor of 1.0. Therefore media that was originally captured squeezed at 2.0 would be horizontally stretched to 1.0. If other media was captured at 1.3, it would be horizontally stretched to become 1.0. If other media was shot spherically at 1.0, it would remain 1.0.

2.0:1 Source:



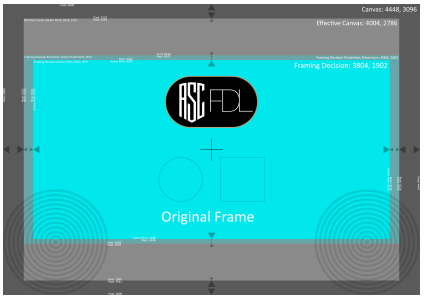
Result: 1.0:1



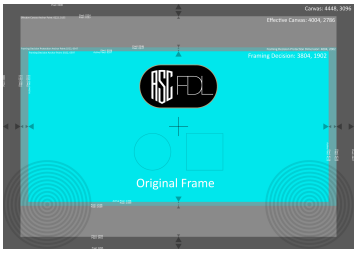
1.33:1 Source

Result: 1.0:1

1.0:1 source



Result: 1.0:1



If a value of 0 has been used within the target anamorphic squeeze field, all media should be left to their source squeeze factor defined in the source FDL's anamorphic squeeze attribute. This is similar to some applications using the terminology of Same As Source. Therefore, if an incoming acquisition was shot at 2.00 anamorphic, and the user set the target anamorphic squeeze to 0, it would remain 2.00.

The source's squeeze factor must be defined in order for the target anamorphic squeeze attribute to work. This is available as the anamorphic squeeze attribute in the source ASC FDL's canvas.

Any application performing this process will need to apply the squeeze factor before any Fit operation (scaling). The order of operations will be: Desqueeze and then Scale, to ensure consistency between applications.

This field's formatting will follow the image deformation numbering system typically used by lens manufacturers. As an example, 1.30 or 2.00. The formatting will specifically be:

```
"target_anamorphic_squeeze": value
```

As an example:

```
"target_anamorphic_squeeze": 1.00,
```

This field has a requirement of at least 2 decimal places, with a maximum of 10.

Data Type: Float

Required Field: Yes

Default Value: **0**

7.6.6 `fit_source`:

After determining a target dimension, the user will need to choose which region to fit into the target dimension from the source files. Available options will be:

- framing_decision.dimensions
- framing_decision.protection_dimensions
- canvas.dimensions
- canvas.effective_dimensions

For the canvas template to successfully target specific regions to place into the target dimension, the incoming media will require associated ASC FDL data.

The fit source field will be formatted as follows:

```
"fit_source": "value",
```

As an example:

```
"fit_source": "framing_decision.dimensions",
```

Data Type: Enum

Required Field: Yes

Allowed Values:

- framing_decision.dimensions
- framing_decision.protection_dimensions

- canvas.dimensions
- canvas.effective_dimensions

Default Value: canvas.dimensions

7.6.7 fit_method:

The fit method attribute specifies how to fit the fit source selection into the target dimension.

Any implementation shall apply the squeeze factor before any Fit (scaling). Therefore the order of operations shall be: Desqueeze > Fit/Scale

The fit method field will be formatted as follows:

```
"fit_method": "value",
```

As an example:

```
"fit_method": "width",
```

Data Type: Enum

Required Field: Yes

Allowed Values: Width, Height, Fit All

Default Value: Width

7.6.8 alignment_method_vertical:

The alignment method allows users to choose to offset their frame vertically, or horizontally. The alignment method vertical, allows the ability to specifically alter the position of the source vertically.

The alignment method vertical field will be formatted as follows:

```
"alignment_method_vertical": "value",
```

As an example:

```
"alignment_method_vertical": "center",
```

Data Type: Enum

Required Field: Optional

Allowed Values: top, center, bottom

Default Value if not specified: center

7.6.9 alignment_method_horizontal:

The alignment method allows users to choose to offset their frame vertically, or horizontally. The alignment method horizontal, allows the ability to specifically alter the position of the source horizontally.

The alignment method horizontal field will be formatted as follows:

```
"alignment_method_horizontal": "value",
```

As an example:

```
"alignment_method_horizontal": "center",
```

Data Type: Enum

Required Field: Yes

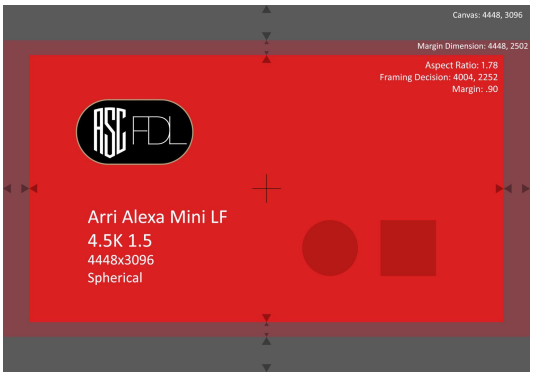
Allowed Values: left, center, right

Default Value: center

7.6.10 preserve_from_source_canvas:

Description:

Once a region from the source has been chosen to fit into the target dimension, the user may choose to preserve a larger area outside of the chosen fit source. Here is an example source image we'll be working with:



In the example above, the source canvas was 4448:3096 and the framing decision within was 4004:2502. Using the canvas template, the user may choose to fit the framing decision into the target dimension of 3840:2160. The user then has the option to either preserve the area outside of this framing decision, or cut it off. Here is an example if the user had chosen framing_decision.dimensions as the preservation:

Therefore nothing is maintained outside of the framing decision and the canvas remains 3840:2160. If the user had chosen canvas.dimensions, the resulting image would have expanded outward within a new canvas dimension of 4262:2968.

This field will be formatted as follows:

```
"preserve_from_source_canvas": "value",
```

As an example:

```
"preserve_from_source_canvas": "canvas.dimensions",
```

Data Type: Enum

Required Field: Yes

Allowed Values:

- none
- framing_decision.dimensions
- framing_decision.protection_dimensions
- canvas.dimensions
- canvas.effective_dimensions

Default Value: none

7.6.11 `maximum_dimension`:

The maximum dimension attribute will determine any newly generated canvas' maximum dimensions. In the example used above for the preserve from source attribute, if a user had chosen to preserve the canvas, but also set the maximum dimension to 4096:2160, the resulting canvas would have been cropped (not scaled) resulting in a 4096:2160.

However, if the user had not defined a maximum dimension, the resulting image would not have been cropped.

If the user as an example chose a maximum dimension of 5000:3496, the resulting image would still remain 4262:2968, as this attribute does not force any kind of scaling.

The maximum dimension attribute will always take priority over the preserve from source results.

The maximum dimension field does not have any requirements on the aspect ratio utilized.

The formatting for this attribute is as follows:

```
"maximum_dimension": {"width": value, "height": value}
```

As an example:
"maximum_dimension": {"width": 5000, "height": 3496}

Data Type: width: int, height: int

Required Field: Optional

Default Value: Blank

7.6.12 pad_to_maximum:

As specified in section 8.6.12, the maximum dimension field only ensures that a canvas is no larger than a specified dimension. Therefore, resulting canvases from different inputs could vary in dimensions if they do not exceed the maximum dimension. The pad to maximum forces resulting canvases to the specified maximum dimension. However it does not scale the image. Instead it adds pillarboxing, or letterboxing if needed.

As an example:

Here is our source, which has a canvas of 4448:3096

The user has chosen to fit the framing decision into a dimension of 3840:2160, and utilized a maximum dimension of 5000:3496. With the pad to maximum field set to false, this would have resulted in a canvas of 4262:2968. Ie; not forcing the canvas to be the maximum dimension.

If the pad to maximum was set to true, this would have resulted in a new canvas that is 5000:3496, with the resulting image both pillarboxed and letterboxed, considering the pad to maximum field does not force any scaling:

If the maximum dimension attribute is blank, the pad to maximum attribute will be disregarded.

Formatting for this attribute will be as follows:

```
"pad_to_maximum": "value",
```

```
As an example:
"pad_to_maximum": "true",
```

Data Type: Boolean

Required Field: Required

Allowed Values: true, false

Default Value: false

7.6.13 `round`:

Users may want to control whether a newly created canvas is allowed to be an odd number. As an example, many users may choose to force any newly created canvas to be an even number of pixels. Different platforms handle rounding in different ways, so defining the rounding “rules” ensures consistency in scaling behavior between platforms. Rounding is one of the key variables that can be defined within a template to ensure consistent results.

The round field is only applicable to the final resulting canvas.dimensions and should not affect the fit source/target dimension.

Whole = to nearest integer
Even = to nearest even-numbered integer

Up = always round up
Down = always round down

Round = follow standard rounding rules (round up for values greater than or equal to +0.5, down for values less than +0.5)

In the case of resulting `canvas.dimensions` defined in a template (if `pad_to_maximum` is `true`) then rounding isn't applicable because that dimension must be filled.

If a resulting canvas needs to have rounding applied, the canvas should never be stretched, squished, or scaled. Instead padding, or cropping should occur.

The formatting for this field will be as follows: `"round": {"even": value, "mode": "value"}`
As an example: `"round": {"even": true, "mode": "up"}`

Show how this will be formatted

Data Type: Enum, Enum

Allowed Values for 1st Enum: Whole, Even, None

Allowed Values for 2nd Enum: Up, Down, Round, None

Required Field: Optional

Default Value: Even, Up

8.0 Appendix

Appendix A: ASC FDL JSON Schema


```
{
  "uuid": "BCD142EB-3BAA-4EA8-ADD8-A46AE8DC4D97",
  "version": {"major": 0, "minor": 1},
  "fdl_creator": "ASC FDL Committee",
  "framing_intents": [{
      "label": "2:1 Framing",
      "id": "FDLSMP01",
      "aspect_ratio": {"width": 2, "height": 1},
      "is_primary": true,
      "protection": 0.1
    }
  ],
  "contexts": [{
    "label": "ArriLF",
    "context_creator": "ASC FDL Committee",
    "canvases": [{
      "label": "Open Gate RAW",
      "id": "ArriLF-20220310",
      "source_canvas_id": "ArriLF-20220310",
      "dimensions": {"width": 4448, "height": 3096},
      "effective_dimensions": {"width": 4448, "height": 3096},
      "effective_anchor_point": {"x": 0, "y": 0},
      "photosite_dimensions": {"width": 4448, "height": 3096},
      "physical_dimensions": {"width": 36.7, "height": 25.54},
      "anamorphic_squeeze": 1.0,
      "framing_decisions": [{
          "id": "ArriLF-20220310-FDLSMP01",
          "framing_intent_id": "FDLSMP01",
          "framing_intent_label": "2:1 Framing",
          "anchor_point": {"x": 222, "y": 547},
          "dimensions": {"width": 4004, "height": 2002},
          "protection_dimensions": {"width": 4448, "height": 2224},
          "protection_anchor_point": {"x": 0, "y": 436}
        }
      ]
    }]
  }]
}
```

Appendix B:
Example ASC FDL JSON File With 2 Framing Decisions


```
{
  "uuid": "DEAF6B84-6B8C-46DB-8CE3-DA5DAB8C9817",
  "version": {"major": 0, "minor": 1},
  "content_creator": "Jane Doe",
  "framing_intents": [{
      "label": "Hero 1.78",
      "id": "29A901F1",
      "aspect_ratio": {"width": 16, "height": 9},
      "is_primary": true,
      "protection": 0.05
    },
    {
      "label": "Hero 2:1",
      "id": "0302684B",
      "aspect_ratio": {"width": 2, "height": 1},
      "is_primary": false,
      "protection": 0.05
    }
  ],
  "contexts": [{
    "label": "ArriLF",
    "content_creator": "Arri LF",
    "canvases": [{
      "label": "Open Gate ARRIRAW",
      "id": "ArriLF-20210902",
      "source_canvas_id": "ArriLF-20210902",
      "dimensions": {"width": 4448, "height": 3096},
      "effective_dimensions": {"width": 4448, "height": 3096},
      "effective_anchor_point": {"x": 0, "y": 0},
      "photosite_dimensions": {"width": 4448, "height": 3096},
      "physical_dimensions": {"width": 36.7, "height": 25.54},
      "anamorphic_squeeze": 1.0,
      "framing_decisions": [{
          "id": "ArriLF-20210902-29A901F1",
```

```
          "framing_intent_id": "29A901F1",
          "framing_intent_label": "Hero 1.78",
          "anchor_point": {"x": 111, "y": 360},
          "dimensions": {"width": 4226, "height": 2376},
          "protection_dimensions": {"width": 4448, "height": 2508},
          "protection_anchor_point": {"x": 0, "y": 294}
        },
        {
          "id": "ArriLF-20210902-0302684B",
          "framing_intent_id": "0302684B",
          "framing_intent_label": "Hero 2:1",
          "anchor_point": {"x": 112, "y": 492},
          "dimensions": {"width": 4224, "height": 2112},
          "protection_dimensions": {"width": 4448, "height": 2224},
          "protection_anchor_point": {"x": 0, "y": 436}
        }
      ]
    }]
  }]
}
```


Appendix C:

Example ASC FDL JSON File With only the Canvas Template


```
{
  "uuid": "3E9F94EF-A910-470D-8EC4-B14E551AC6AB",
  "version": {"major": 0, "minor": 1},
  "content_creator": "The Camera",
  "framing_intents": [{
      "label": "2.39 Framing",
      "id": "FDLSMP05",
      "aspect_ratio": {"width": 2048, "height": 858},
      "is_primary": true,
      "protection": 0.1
    }
  ],
  "canvas_templates": [{
      "label": "VFX Pull",
      "id": "VX220310",
      "target_dimension": {"width": 4096, "height": 1716},
      "target_anamorphic_squeeze": 1.0,
      "fit_source": "framing_decision.dimensions",
      "fit_method": "width",
      "alignment_method_vertical": "center",
```

```
      "alignment_method_horizontal": "center",
      "alignment_offset": {"x": 0, "y": 0},
      "preserve_from_source_canvas": "canvas.dimensions",
      "round": {"even": true, "mode": "up"}
  },
  {
      "label": "Editorial Dailies",
      "id": "ED220310",
      "target_dimension": {"width": 1920, "height": 1080},
      "target_anamorphic_squeeze": 1.0,
      "fit_source": "framing_decision.dimensions",
      "fit_method": "width",
      "alignment_method_vertical": "center",
      "alignment_method_horizontal": "center",
      "alignment_offset": {"x": 0, "y": 0},
      "preserve_from_source_canvas": "framing_decision.dimensions",
      "round": {"even": true, "mode": "up"}
  }]
}
```

Appendix D:
Example ASC FDL JSON File With Effective Canvas

```
{
  "uuid": "5EDD03DC-4EFF-42BB-8085-DDECC3036982",
  "version": {"major": 0, "minor": 1},
  "content_creator": "ASC FDL Committee",
  "framing_intents": [{
      "label": "2:1 Framing",
      "id": "FDLSMP04",
      "aspect_ratio": {"width": 2, "height": 1},
      "is_primary": true,
      "protection": 0.05
    }
  ],
  "contexts": [{
    "label": "ArriLFV",
    "content_creator": "ASC FDL Committee",
    "canvases": [{
```

```
      "label": "Open Gate Vignette",
      "id": "ArriLFV-20220311",
      "source_canvas_id": "ArriLFV-20220311",
      "dimensions": {"width": 4448, "height": 3096},
      "effective_dimensions": {"width": 4004, "height": 2786},
      "effective_anchor_point": {"x": 222, "y": 155},
      "photosite_dimensions": {"width": 4448, "height": 3096},
      "physical_dimensions": {"width": 36.7, "height": 25.54},
      "anamorphic_squeeze": 1.0,
      "framing_decisions": [{
          "id": "ArriLFV-20220311-FDLSMP04",
          "framing_intent_id": "FDLSMP04",
          "framing_intent_label": "2:1 Framing",
          "anchor_point": {"x": 100, "y": 442},
          "dimensions": {"width": 3804, "height": 1902},
          "protection_dimensions": {"width": 4004, "height": 2002},
          "protection_anchor_point": {"x": 222, "y": 547}
        }
      ]
    }]
  }]
}
```


Appendix E:
ASC FDL Represented in an ALE

A user may choose to communicate which ASC FDL should be applied to which shot through a metadata exchange file such as an ALE. There are two columns that should be provided in an ALE in order for any application to know which FDL to apply to any given shot:

- fdl-uuid
- fdl-framing-decision-id

The uuid field will be able to tell any application down stream which ASC FDL file to use for that particular shot. Whereas the framing decision id will tell the application which specific frame within the FDL to use.

Appendix F:
ASC FDL Data Placed within the header of a file:

A user may choose to communicate which ASC FDL should be applied to which shot by adding the necessary FDL info to the header of a rendered file. There are two key pieces of data that should be shared in order for any application to know which FDL to apply to any given shot:

- fdl-uuid

- fdl-framing-decision-id

The uuid field will be able to tell any application down stream which ASC FDL file to use for that particular shot. Whereas the framing decision id, will tell the application which specific frame within the FDL to use.

Appendix G:
ASC FDL Represented in an EDL

A user may choose to communicate which ASC FDL should be applied to which shot by adding the necessary FDL info to the comments section of an EDL. There are two key pieces of data that should be shared in order for any application to know which FDL to apply to any given shot:

- fdl-uuid
- fdl-framing-decision-id

The uuid field will be able to tell any application down stream which ASC FDL file to use for that particular shot. Whereas the framing decision id will tell the application which specific frame within the FDL to use.

The formatting of these values should be as follows:

* FDL-UUID: value
* FDL-FRAMING-DECISION-ID: value

Appendix H: Character sets allowed for ASC FDL labels

Any human editable attribute within the ASC FDL shall only utilize the numbers 0-9, letters a-z and the following special characters: “-”, “_”, “+”, “.”. Spaces shall not be permitted within the name of an ASC FDL. Here is an example set of characters that shall not be permitted:

*@ # $ % ^ & * ( ) ` ; : < > ? . , [ ] { } / \ ' " | ~*

TRADEMARK
ASC FDL™ is a trademark of American Society of Cinematographers.

DISCLAIMER
American Society of Cinematographers is not liable for any damages, including direct or consequential, from the use of ASC FDL specification outlined in this document.

LICENSING
This specification is made available under MIT License stated here - https://github.com/git/git-scm.com/blob/main/MIT-LICENSE.txt

NOTICE
For any further explanation of the contents of this document, or in case of any perceived inconsistency or ambiguity of interpretation, please contact American Society of Cinematographers at: ascfdl.feedback@gmail.com