

# OTIO 2D-Annotations Interchange specification

# OTIO 2D-Annotations Interchange specification.

This version: May, 2025 v2

Previous version: [OTIO Annotations - Review-1](#)

Sam Richards

## Introduction

The goal of this format is to facilitate interchange of annotations and notes between different review systems or provide a very simple offline review format for a simple review system.

The user stories are defined in [Review and Annotation User Stories and Requirements](#).

We can break those stories into three categories:

1. 2-D annotations on top of movies or 2-D imagery.
2. 2/D and 3/D annotations on top of models and 3-D environments.
3. Annotations recorded during a live session.

The live session needs to be an extension of

<https://lf-aswf.atlassian.net/wiki/spaces/PRWG/pages/11274625/OTIO-Based+Synchro...>

Similarly model and environment annotations should be handled as a future proposal. For this document we are going to focus on 2-D annotations on top of movies or 2-D imagery, integrating the result into OTIO.


## Why OTIO

The goal is to have an application neutral format for sharing annotations between applications. This format should work both within a facility and between vendors and clients even if they use different production management tools.

OTIO provides a framework for tracking a list of media assets, it is highly extensible so adding additional metadata schemas on-top is fairly straightforward, and some of the more recent developments will work well with the requirements for annotations, such as:

- Spatial coordinate system - <https://github.com/AcademySoftwareFoundation/OpenTimelineIO/pull/1219>
- Color Management - <https://github.com/AcademySoftwareFoundation/OpenTimelineIO/discussions/1805> and <https://github.com/AcademySoftwareFoundation/OpenTimelineIO/discussions/1793>
- Review syncing protocol -

<https://lf-aswf.atlassian.net/wiki/spaces/PRWG/pages/11274625/OTIO-Based+...>

- Strong time framework - <https://opentimelineio.readthedocs.io/en/stable/tutorials/architecture.html#otio-opentime>
- MultiMediaReference -  MultiMediaReference
- OTIOZ files as a way to bundle movie deliverables with notes for review, and for feedback the annotation images.

## Terminology

Annotation – A layer over a 2D or 3D object, typically a drawing, for a single point in time, Typically used to give feedback, but can also be used to identify a point of interest in a piece of reference.

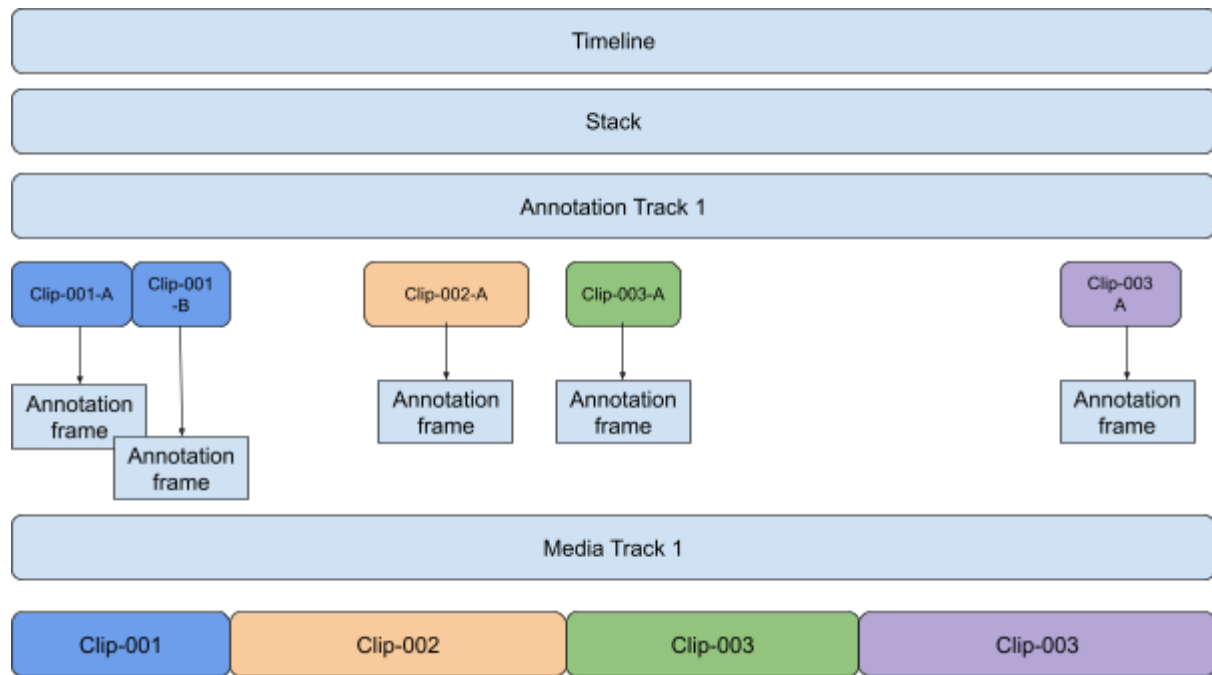
Notes – Are text based notes which can be associated with the overall media being reviewed, but can also be associated with a single frame.

Vendor – a vendor is a company who is creating some but not all of the content.

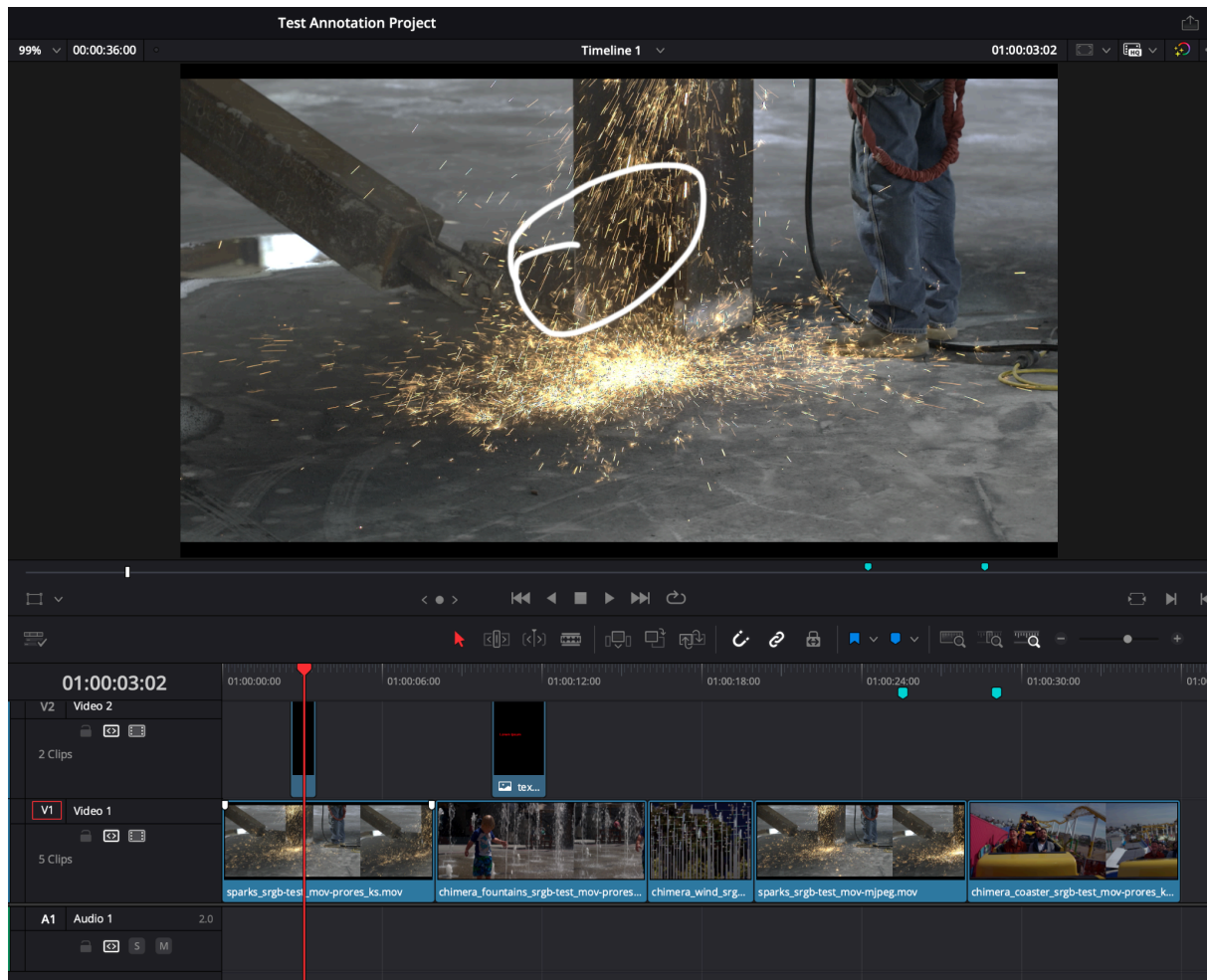
Client – The client for the vendor, all media has its final reviews here. Note, there may be still multiple review levels within client studio, e.g. VFX Supervisor and then Director for final buy-off. Also the client studio may have multiple vendors.

## Track based annotations

Our proposal is to take advantage of the overlaying functionality of tracks, to have each review be a different track. The Annotation track ideally could act as a conventional timeline (for backwards compatibility with legacy systems), where each annotation frame is defined using a Clip.2 schema with a still frame with alpha (using a PNG file).



Additionally, each Annotation-frame node, would also have a representation of the annotation as a vector timeline, using the ANNOTATION\_1.0 timeline defined by the sync messaging - <https://lf-aswf.atlassian.net/wiki/spaces/PRWG/pages/11274625/OTIO-Ba...>



## Example Annotation Frame

The annotation is shown within the clip for the specified frame.  
Within the clip, so a clip might look like:

```
{
  "OTIO_SCHEMA": "Clip.2",
  "metadata": {
  },
  "name": "Chimera_DCI4k5994p_HDR_P3PQ_%06d",
  "source_range": {
    "OTIO_SCHEMA": "TimeRange.1",
    "duration": {
      "OTIO_SCHEMA": "RationalTime.1",
      "rate": 60.0,
      "value": 1.0
    },
    "start_time": {
      "OTIO_SCHEMA": "RationalTime.1",
      "rate": 60.0,
      "value": 44200.0
    }
  }
}
```

```

    }
  },
  "effects": [],
  "markers": [],
  "enabled": true,
  "media_references": {
    "DEFAULT_MEDIA": {
      "OTIO_SCHEMA": "ImageSequenceReference.1",
      "metadata": {},
      "name": "",
      "available_range": {
        "OTIO_SCHEMA": "TimeRange.1",
        "duration": {
          "OTIO_SCHEMA": "RationalTime.1",
          "rate": 60.0,
          "value": 1.0
        },
        "start_time": {
          "OTIO_SCHEMA": "RationalTime.1",
          "rate": 60.0,
          "value": 44200.0
        }
      },
      "available_image_bounds": null,
      "target_url_base": "annotation_overlay",
      "name_prefix": "annotation_clip1",
      "name_suffix": ".png",
      "start_frame": 1,
      "frame_step": 1,
      "rate": 60.0,
      "frame_zero_padding": 4,
      "missing_frame_policy": "error"
    }
  },
},
"annotations": {
  "OTIO_SCHEMA": "ANNOTATION_1.0",
  "author": "Sam Richards",
  "canvas_size": [1920, 1080],
  "creation_timestamp": "2025-01-31T16:14:00Z",
  "annotation_note": "This is final",
  "annotation_renderer": "RV-7.1",
  "annotation_commands": [
    {
      "event": "PAINT_START",
      "payload": {
        "source_index": 0,
        "paint": {
          "OTIO_SCHEMA": "Paint.1",
          "points": [],
          "rgba": [ 1.0, 1.0, 0.0, 1.0 ],
          "type": "COLOR",
          "brush": "circle",
          "visible": true,
          "name": "Paint",
          "effect_name": "Paint",
          "layer_range": {
            "OTIO_SCHEMA": "TimeRange.1",

```

```
    "start_time": {
      "OTIO_SCHEMA": "RationalTime.1",
      "value": 0.0,
      "rate": 30.0
    },
    "duration": {
      "OTIO_SCHEMA": "RationalTime.1",
      "value": 1.0,
      "rate": 30.0
    },
    "hold": false,
    "ghost": false,
    "ghost_before": 3,
    "ghost_after": 3
  }
}
}
```

To break it down, each annotation is typically for a single frame for a clip (although could also be for a range of frames or the whole clip), if there is an actual annotation, it should point to a frame, typically a PNG file (TODO Determine if there is a better spec), which needs to have an alpha channel, so that the frames can be overlaid on top of the actual media.

In the *annotations* block you can have the following fields:

- author - Who is making the annotations, possibly based on the username, but this might need to be overwritten for some reviews.
- creation\_timestamp - When was this annotation created using ISO 8601 (e.g. **2017-05-16T10:30:56+01:00**).
- The original\_frame\_number of the clip. Note this is only if start\_frame\_number (see below) is defined.
- Canvas\_size: An array for the width and height of the canvas being drawn on (see below for examples).
- annotation\_note - This is a note associated with this frame (and there might not be an actual image, it might be just the note), we recommend that this at a minimum support [Markdown](#) (See below). NOTE, this note is off-screen, and is in addition to any captioned note (see below).
- clip\_uuid (optional) - A UUID provided by the vendor to help ensure that we are annotating the right clip. This can be any string the vendor likes, provided that its unique within the vendors facility. See [below](#).
- status (optional) - This can be a single string, or an array of strings. If it's an array of strings, this is used to denote the possible values of the status. This would be used in the case of a client updating the status to pass back to the vendor, and the vendor has their own definitions of statuses.
- annotation\_renderer - Which protocol was used to render the annotation. While we should strive to have annotation renderers that look the same most of the time, its possible there may be proprietary renderers where the brush strokes may not be

easily emulated by other renderers. This makes it more likely that you would want to use the pre-rendered overlay.

- `annotation_commands` (optional) - A list of commands that can be used to re-create the annotation using the sync command set.

## Annotation colorspaces

The annotations are assumed to be in sRGB colorspace, however the hope is that the results of the [ASWF color-interop-forum](#) will eventually be merged into OTIO to make it clear what the colorspace of each set of media is, to help with their later combining.

An example of why this is important is that an annotation color could be defined by picking a color off the media, and then painted over part of the screen. If the wrong colorspaces are used, we could easily end up with a different color making the artistic intent wrong.

## Text Formatting and Tagging.

As mentioned above we recommend using Markdown for text general formatting, but additionally recommend

1. Hashtags (`#tag`):
  - This is a very common convention, borrowed from social media.
  - Example: `This is a note about #project-alpha and #meeting-notes.`
2. Mentions (`@user`):
  - Allow reviewer to flag a particular user.
  - Standard Markdown processors will just render this as text.
  - Example: `Hey @jane-doe, can you review this? #review-request`

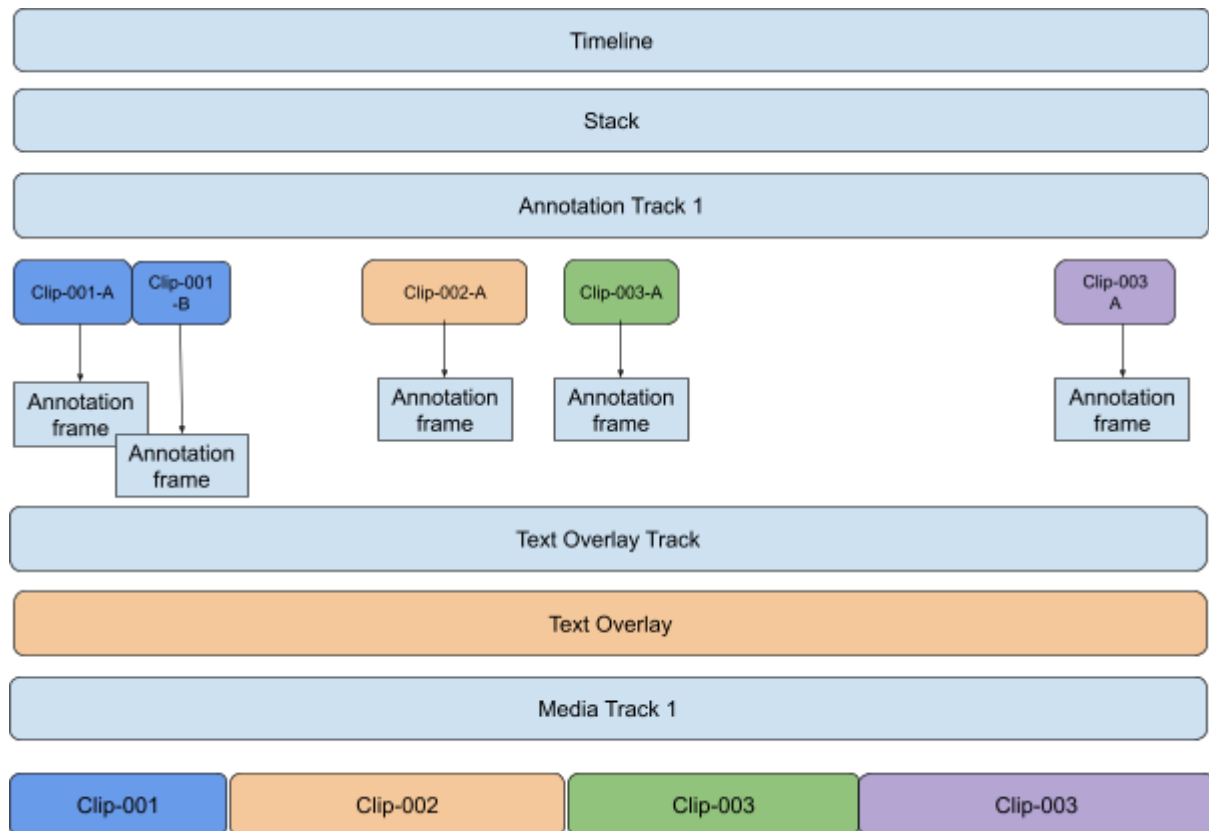
TODO: Should there be other conventions, e.g. to refer to other media?

E.g.: You could have a variable hashtag that refers to underlying media metadata, such as:

- `#var:frame` - the current media frame number.
- `#var:timecode` - the current media timecode.
- `#var:filename` - current media filename.
- `#var:artist` - current media artist name.
- `#var:creationdate` - current media creation date.
- `#var:taskstatus` - show the current task status (which probably comes from production management system, not OTIO file).

This would allow you to create an overlay with metadata and position it as you like that would cover all the media, rather than embedding the overlay in the media. This has the benefit that the overlay track could easily be disabled. See below...





Another example could be tagging media objects...

#asset:spiderman

## Brush Strokes

Note, this is optional, since PNG files can be used, however there are advantages to including the actual brush strokes in potentially a more lightweight transfer file. It does also allow for ingest into other systems (e.g. maya) as a set of brush-strokes which may have additional benefits. Additionally being able to manipulate the original brush-strokes, and read any text allows you to use the raw data in interesting ways (e.g. searching any text, or copy and pasting individual brush-strokes, rather than the whole image).

For each annotation, the canvas-size would be defined. This makes it easy to define text size, since the font pixel size can easily be specified.

Each paint surface needs:

Frame(s)

Window-geometry.

Each brush stroke needs a minimum of:

- stroke\_id - Needed for cases where there are multiple annotations happening at the same time, string based, e.g. [GUID](#).
- point\_index
- Color (RGB)

- Width (float) - Stroke width (Normalized units). TODO - is this scaled for image size?
- point\_value (x, y)
- Brush type: Draw, Erase.
- Pen\_type (Gauss, Circle)
- Pen Opacity: 0-1
- Pen-tilt (tx, ty) (Optional)
- Pen Pressure: (Units?) (Optional) - Not needed for rendering, just reference. Should be converted to brush width, and opacity.

Pen move / up:

- stroke\_id, string based, e.g. [GUID](#).
- point\_index
- point\_value (x, y)
- pen\_tilt (tx, ty) (Optional)
- pen\_pressure (Optional)

## Text Captions

- text\_id (unique ID, e.g. [GUID](#))
- Color (RGBA)
- Spacing
- Size (pixel height).
- Scale
- Background color (defaults to 0, 0, 0)
- Background opacity (defaults to 0)
- Font - We may want to limit what fonts can be used to fonts that are known to be cross platform, such as Arial/Helvetica, Times new roman, Verdana (Need to check)? We should support UTF-8 and I18N character encoding. TODO - figure out how this is supported by json.
- Text - Note, text should allow markdown formatting, see below.
- Origin of the text box (top left)
- Width: - Box width is defined, but height is governed by the amount of text.
- Alignment - Left, center, right.

TODO - Arrows, boxes, circles? - Are these important?

## Overall review metadata

Additionally there would be track metadata that would denote information about each review.

Example information could be:

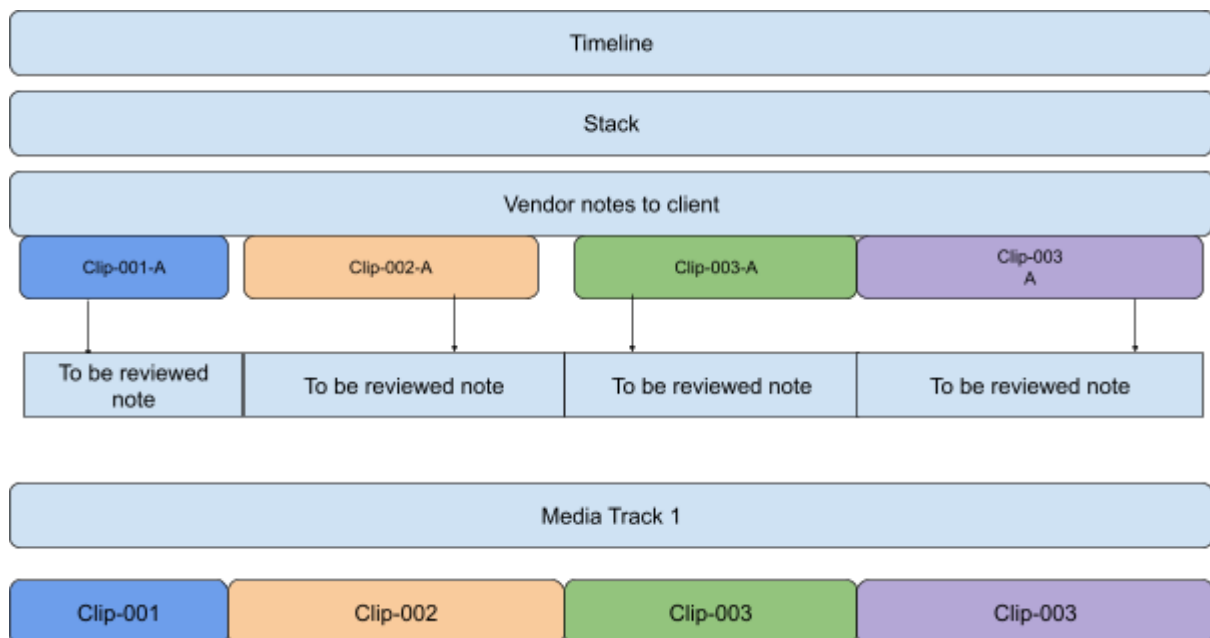
- Review title
- Participants
- Review start time
- Location(s)
- Review description
- Vendor name

Note: an annotation is not required, you can just have the text note, which would make it somewhat similar to a marker.

## Vendor notes

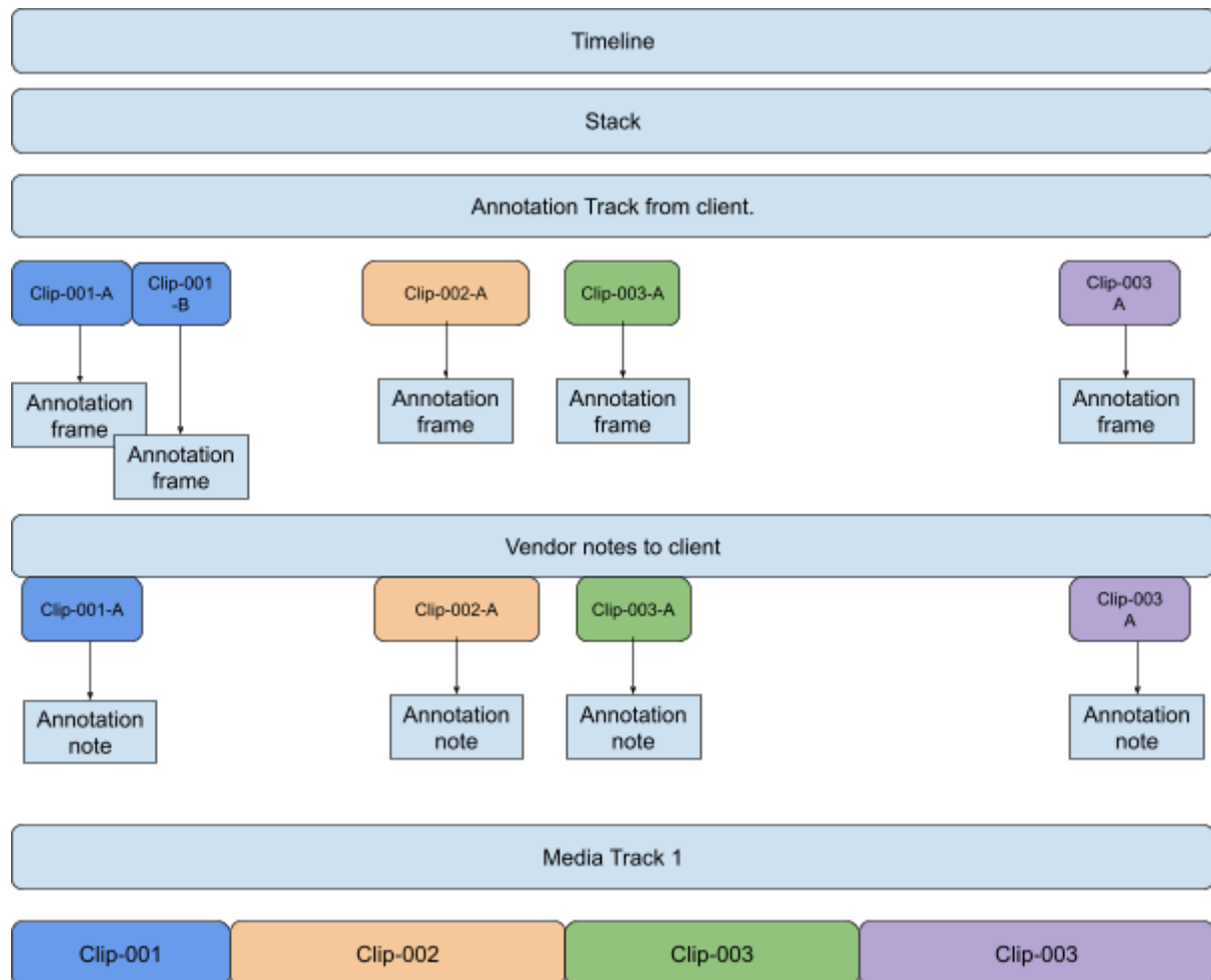
Beyond just the clipname, there is metadata that the vendor needs to send to the client.

In this situation the “to be reviewed note” would typically be everything other than the actual annotations (although there might be cases where annotations are handy too). The note would be information for the reviewer on what they are reviewing it for, e.g. WIP, For Final, or for approval of an element (e.g. Approval of water sim). The note would span the whole clip, unless a specific frame needs to be highlighted by the vendor as a message to the client.



So in the above example, the vendor would send along with the media to be reviewed a OTIO clip that looks like the above.

Once the client reviewer has completed their work, they would get back an OTIO file, which would have the same OTIO file, but with an additional track with the feedback from the vendor.



## Clip Metadata

We would also expect a number of metadata fields to be added to each clip:

- **start\_frame\_number** (optional) - The start frame number is 0 for compressed movies unless somebody is assuming timecode has been mapped to frame number, explicitly defining the start-frame is a useful way to correctly identify offsets within the clip. If this is undefined the timecode should be converted to a frame number.
- **vendor\_name** (optional) - The name of the vendor
- **Task** (optional) - The task that is being reviewed.
- **artist\_name** (optional) - the name of the artist who submitted the media.
- **Vendor-id** (optional) - A unique ID generated by the vendor (see below).
- **client-id** (optional) - A unique ID generated by the client(see below).
- **Clip\_uuid** (optional) - A unique ID for this clip, to be used by the annotation note, to ensure that we are annotating the right clip. This is generated by the vendor and should be carried through the review pipeline. If its undefined it should never be added, since its not clear if the clip\_uuid has simply been dropped by some process.

## vendor-id/client-id

It's common for vendors to have a way to identify an item for review, e.g. bar-4356 where "bar" is the show code, and 4356 is an index into a review table for the "bar" show. This ID is human readable, and relatively short compared to a UUID, so is ideal for referencing compared to a file-string, and it also allows the indirection of proxy media for review, since they all would share the same vendor-id.

The problem comes when sharing the media with a client, who ingests it into their own system, and have their own tracking ID. So we need a clear way to have both a vendor and client id.

We propose that the vendor-id be added and if it exists, it is not overwritten. The client-id should be the review-id of the local system, i.e. when ingested into a clients system, you would overwrite this ID.

## Media Multi References

OTIO has a framework for allowing multiple pieces of media to be referred to within a single clip. A typical use case could be the original full rez frames, along with proxy versions.

This when combined with the media reference plugin framework

<https://opentimelineio.readthedocs.io/en/latest/tutorials/write-a-media-linker.html> could be used to have the client, vendor and "DEFAULT\_MEDIA" path. We recommend that the media that is travelling with the OTIO file (e.g. in an OTIOZ) would be the "DEFAULT\_MEDIA" entry, and have "active\_media\_reference" set to "DEFAULT\_MEDIA". Then each vendor/client would have alternate clips based on their facility name.

## Import Export Adapters

The fact that the actual clips are on a separate track does make it slightly harder to match the clips up. So an API should be developed to allow easy importing and exporting to the neutral OTIO format.

This would be designed for easy ingest into existing applications, would be using C++ for easy usage in non-python friendly applications.

## Pros/Cons

There are several benefits of this system:

- It could allow a reviewer to be able to load a OTIOZ file into a reviewing platform that supports the annotation workflow, and do the review offline. Save the OTIO file back

out with their notes, and then send the smaller OTIO file back to a client. If the overlay images are required, it would still need some form of the OTIOZ file (perhaps the actual media track wouldn't be embedded in the OTIOZ file).

- It would allow Editorial to be able to load the annotations into any editor that supports OTIO files (TODO creates an OTIO file to illustrate this).

There are a couple of concerns:

- This does not natively support non time based workflows, such as model reviews, or for media such as PDF's. PDF's are the more straightforward, since we can treat each page as a separate image.
- We are relying on the annotation format to support 3d annotations, which is not currently defined.
- PDF media in particular might benefit from more direct markers associated with text (i.e. highlight text, and then have a comment thread).

## Future Extensions

There are a number of areas that a future iteration of the annotation spec could be extended to support, we list some of the ideas, and what technology needs to be in place to support it.

1. [Framing decision lists](#) - for now, we are assuming all media sets are the same aspect ratio. FDL's could help enormously in how to combine media that does not have the same aspect ratios, or based on the project needs to be combined in a particular way.
2. Track/clip colorspace - integrating the work from the color-interop-forum into OTIO, which would be a huge benefit when picking annotation colors off the screen.
3. Taking advantage of advanced brush stroke renderers such as <https://disneyanimation.com/technology/meander-1/>
4. Support for 3d annotations, e.g. for model reviews.

## Additional Tasks

- Create example media. A reference OTIOZ file with embedded PNG files should be possible without any coding, as a test.
- We are relying on the sync protocol for the annotations, but that is not well defined.
- Build reference plugin to xstudio and rv.
- Build reference plugin for Flow production management.
- A sample annotation renderer will need to be written, this would allow for similar results to be rendered across multiple applications. Good example of issues this addresses include how brush-strokes with opacity that intersect are rendered.

# Annotations V2 additions

## Annotations V2 additions

This has mostly merged into the main doc, see the main tab...

## Coordinate system

The OTIO spatial coordinate system -

<https://github.com/AcademySoftwareFoundation/OpenTimelineIO/blob/main/docs/tutorials/spatial-coordinates.md> - does not specify how things drawn inside work. Its not normalized.

Is there a place for FDL -

[https://github.com/ascmitc/fdl/blob/main/ASCFDL\\_UserGuide\\_v1.0.pdf](https://github.com/ascmitc/fdl/blob/main/ASCFDL_UserGuide_v1.0.pdf) Seems useful if it is defined, perhaps we include it, if its known in the OTIO file?

This is solving a related problem to the spatial coordinate systems, in how different resolution pictures relate to each other. (Particularly if they are crops).

Can we solve this by simply stating what the resolution is of the annotation area, and reference a FDL if it exists? All coordinates should be floating point, rather than absolute pixels, to ensure that brush strokes (particularly for low resolution media) are still smooth.

Benefits of absolute coordinate system:

- Does make it easier to define text size, and exact brush sizes, particularly for tilt.
- Assuming we always supply "canvas size" its easy to normalize coordinates if necessary.

## Annotation Scaling

When you zoom into an annotated image, do you want to see the brush strokes and text scale, or should they stay the same size?

- Having text size preserved can be good, but sometimes if its a note, you just want to be able to read it, and know where the note is associated to, so having the text scale up if you zoom in is not necessarily helpful.
- Similarly with some annotations, seeing a sharp edge, can be useful.

## Colorspace

There is progress defining standard colorspace names (see <https://www.aswf.io/news/introducing-the-aswf-color-interop-forum/>) but this still may be a little way off. For the initial release, we should assume all annotations whether its a PNG overlay, or vector graphics, we should assume that they are created in sRGB but composited in linear space.

Colorspace can be an issue when an image is sampled to use for a brush, and the annotation is based on that sampled brush stroke.



# Brush Strokes

Image space should be normalized.

Decision - Normalized for image width only? If its normalize for x, and y, does that affect how scaling works for brush-strokes (particularly for tilt?).

Each paint surface needs:

Frame(s)

Window-geometry.

Each brush stroke needs a minimum of:

- Stroke-ID - Needed for cases where there are multiple annotations happening at the same time.
- point\_index
- Color (RGB)
- Width (float) - Stroke width (Normalized units). TODO - is this scaled for image size?
- point\_value (x, y)
- Brush type: Draw, Erase.
- Pen\_type (Gauss, Circle)
- Pen Opacity: 0-1
- Pen-tilt (tx, ty) (Optional)
- Pen Pressure: (Units?) (Optional) - Not needed for rendering, just reference. Should be converted to brush width, and opacity.

Pen move / up:

- stroke\_id
- point\_index
- point\_value (x, y)
- pen\_tilt (tx, ty) (Optional)
- pen\_pressure (Optional)

## Text Captions

- Text ID
- Color (RGB)
- Spacing
- Size (pixel height).
- Scale
- Font - We may want to limit what fonts can be used to fonts that are known to be cross platform, such as Arial/Helvetica, Times new roman, Verdana (Need to check)?

We should support UTF-8 and I18N character encoding. TODO - figure out how this is supported by json.

- Text - Note, text should allow markdown formatting, see below.
- Origin of the text box

TODO - Arrows, boxes, circles?

## Text Formatting and Tagging.

As mentioned above we recommend using Markdown for text general formatting, but additionally recommend

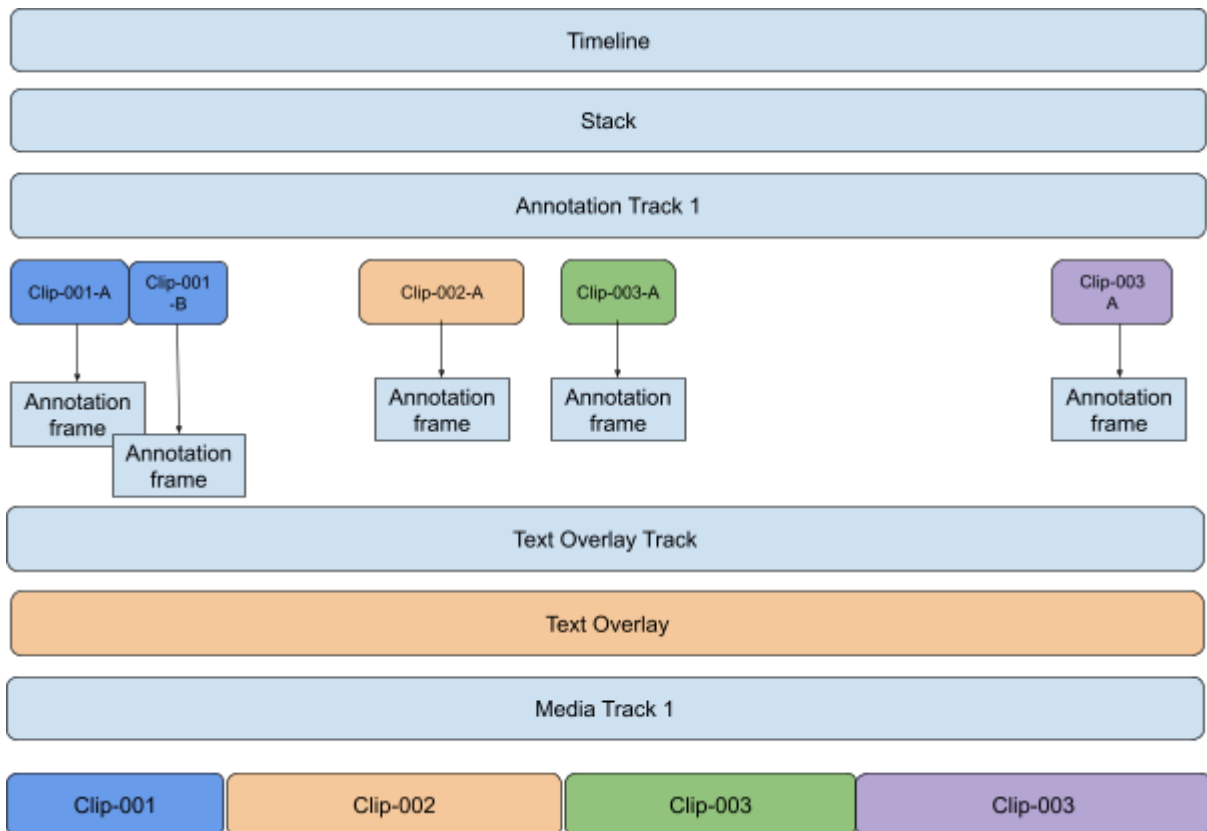
1. Hashtags (`#tag`):
  - This is a very common convention, borrowed from social media.
  - Example: `This is a note about #project-alpha and #meeting-notes.`
2. Mentions (`@user`):
  - Allow reviewer to flag a particular user.
  - Standard Markdown processors will just render this as text.
  - Example: `Hey @jane-doe, can you review this? #review-request`

TODO: Should there be other conventions, e.g. to refer to other media?

E.g.: You could have a variable hashtag that refers to underlying media metadata, such as:

- `#var:frame` - the current media frame number.
- `#var:timecode` - the current media timecode.
- `#var:filename` - current media filename.
- `#var:artist` - current media artist name.
- `#var:creationdate` - current media creation date.
- `#var:taskstatus` - show the current task status (which probably comes from production management system, not OTIO file).

This would allow you to create an overlay with metadata and position it as you like that would cover all the media, rather than embedding the overlay in the media. This has the benefit that the overlay track could easily be disabled. See below...



Another example could be tagging media objects...

#asset:spiderman

Tab 3

Adaptor nodes? How to map from RVPaint to

Using custom schemas - not portable?

HOW to use the tool.