Studies in Ethiopian Languages, 12 (2023), 117−139

Research Note

ELAN and IIIF for Language Documentation*

Yona TAKAHASHI
(Tokyo University of Foreign Studies)
takahashi.yona@aa.tufs.ac.jp

Abstract

This article details a straightforward method for converting data from the EAF (ELAN Annotation Format) of the video annotation software ELAN to IIIF Manifest and WebVTT files. The converted IIIF Manifest and WebVTT files can be played back using the Mirador Video Annotation Support Version, which has implemented some additional features necessary for language education. As a result, researchers now have a seamless pathway to leverage their ELAN data through IIIF.

1 Introduction

The theme of this article has two parts.

- What is IIIF, and how can we use it?
- How can we integrate ELAN and IIIF?

That is, we discuss how to create IIIF manifests and the principles of transformation from ELAN data into IIIF manifests.

2 What is IIIF?

One significant drawback of digital archives is that they are individually published in various locations. This situation requires users to remember how to use each site, resulting in significant differences in usability.

Fortunately, the inconvenience of navigating through individually published digital archives is swiftly being alleviated with the widespread adoption and embracement of IIIF (the International Image Interoperability Framework). This framework is revolutionizing the field of digital archives, providing a unified platform for publishing

* This article is a revised version of a presentation at the 'Technical Workshop for Language Documentation' at the Research Institute for Languages and Cultures of Asia and Africa on March 19, 2024, funded by JSPS KAKENHI Grant Number 18KK0009.

and sharing images on the WWW. This standard, spearheaded by renowned institutions such as the British Library, the National Library of France, the Bodleian Library at the University of Oxford, Stanford University, Princeton University, and Yale University, has been prominently featured multiple times in the National Diet Library's Current Awareness Portal, underscoring its credibility and potential.

By standardizing and centralizing the exchange method of images, annotation texts, and other related data among computers in each institution, IIIF empowers users with the freedom to choose from myriad utilization methods, putting them in control of their digital archive experience.

If we publish images and videos in compliance with IIIF, we can choose and use our favorite viewer from several options. Regardless of which server they are published on, we can universally manage them within a single viewer. No matter which digital archive in the world, if we load its image file into our viewer, it will display the information for us.

Example 1:
Manuscrit reconstitué : Châteauroux, Bibliothèque municipale, ms. 5 (*Grandes Chroniques de France*) <https://demos.biblissima.fr/chateauroux/demo/>.

It displays fragments of Western manuscripts scattered and stored in various institutions. Suppose only the functionality of overlaying images like this could have been provided on the web ten years ago. However, IIIF is groundbreaking because it has made it possible to realize this functionality as an open, common standard and many influential institutions worldwide are already adopting that standard.

Example 2:
Ethiopian Language Archive <https://dev.jael.info/documentation/>

We can watch subtitled videos with Mirador (Video Annotation Support Version) <https://dev.jael.info/mirador3va/>, which is an open-source, community-driven software written in JavaScript[1]. Figure 1 shows the specific features of Mirador (Video Annotation Support Version).

1 Mirador (Video Annotation Support Version) is being developed by the Center of Next-Generation Humanities Development at the Graduate School of Humanities and Sociology (The University of Tokyo), the International Institute for Digital Humanities, the Institute for Languages and Cultures of Asia and Africa (Tokyo University of Foreign Studies), and Felix Style Co., Ltd. For further information, see Takahashi, Nagasaki and Homma (2022).

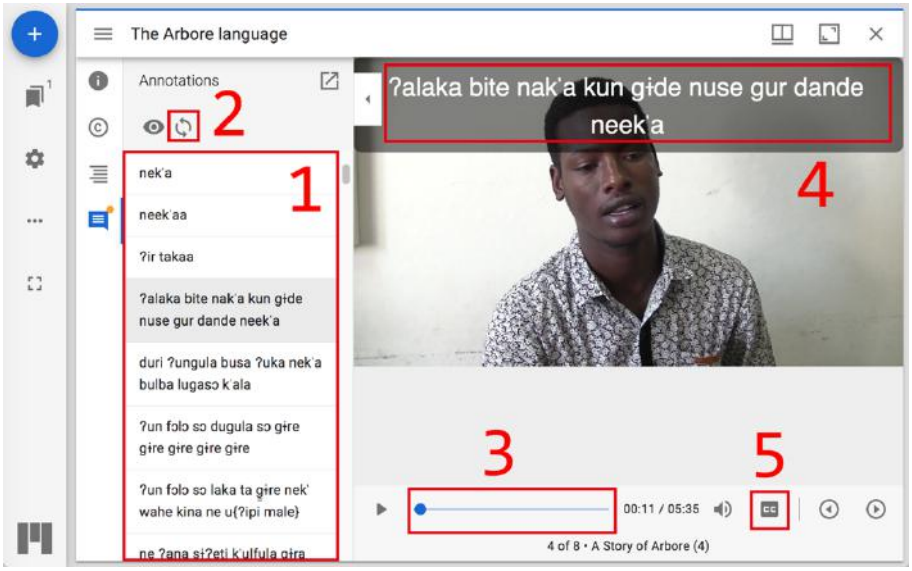


Figure 1: Added features in Mirador (Video Annotation Edition)

1. Automatic Scroll for Annotation Sidebar

Mirador displays annotation texts in a sidebar and automatically selects annotation texts based on the video playback time. If an annotation outside the visible area is selected, the sidebar will automatically scroll to bring that annotation to the center.

2. Toggle for Automatic Scroll

While a helpful feature, automatic scrolling can sometimes hinder our search for specific words in the annotation text list within the sidebar. To give us more control, we have added a button in the header area of the sidebar. This toggle lets us turn automatic scrolling on or off, putting the power in our hands.

3. Bidirectional Link between Sidebar and Seek Bar

With the new bidirectional link feature, navigating between the sidebar's annotation text and the playback position is now a breeze. When we select a playback position from the seek bar, the sidebar automatically scrolls to the corresponding annotation text for that period. This feature saves us time and ensures we get all annotations of our concerns.

4. Subtitles with WebVTT

Mirador internally generates an HTML video element. Modern web browsers can utilize subtitle formats such as WebVTT through the track element. Therefore, the feature to load a WebVTT file in Mirador and display subtitles on the video is now available.

5. Turn on/off subtitles

A button to toggle the on/off state of subtitles using WebVTT is located next to the seek bar. Note that it is currently impossible to toggle individual subtitles, e.g., transcriptions or translations.

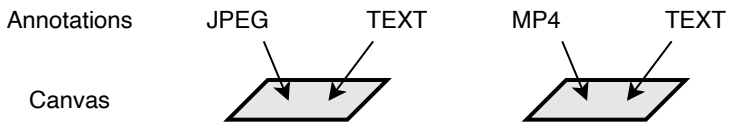
3 Creating IIIF Manifests

3.1 Overview

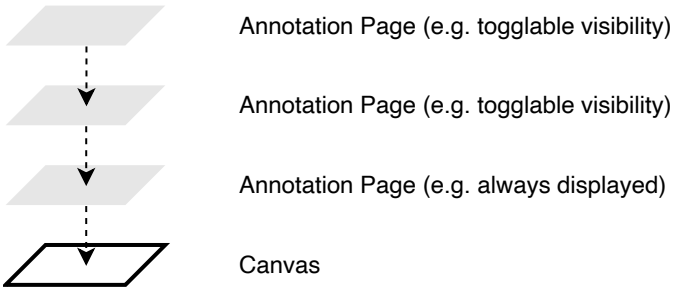
The images, audio, and videos we have on our hands can be transformed into a format compliant with IIIF and publicly available on the WWW. The general flow is as follows:

- Declare a *manifest*.
- Declare each *canvas* included in the *manifest*.
- Declare each *annotation page* contained in each *canvas*.
- Declare each *annotation* contained in each *annotation page*.

A *canvas* represents the display area of an IIIF viewer. In general, one *canvas* is prepared for each material (image, video). Multiple materials can be placed on one *canvas* (e.g., an image and an annotation text, a video and a subtitle text). The materials placed on the *canvas* are called *annotations*.



However, instead of directly placing materials (*annotations*) on the *canvas*, layers called *annotation pages* are created first and then overlaid on it.



Each *manifest*, *canvas*, *annotation page*, and *annotation* are all described as JSON objects based on JSON-LD.

3.2 Step by Step

3.2.1 Manifest

The *manifest* is a JSON object encompassing all the settings required to load into an IIIF viewer.

- It must have a '`type`' property with the value '`Manifest`'.
- It must have an '`id`' property, with the value being the URI of the *manifest* itself.
  - It can also have a resource file name like <`https://example.org/Archive/Record1/manifest.jsonld`>. It is sufficient as long as a unique identifier that does not overlap with others (the URI string itself has no semantics).
- It must have a '`label`' property with the value being the title of the entire data the *manifest* indicates. Since the property supports multiple languages, it must be written in the format `{ "language": [ "title" ] }`.
- It must have '`items`' property with the value of the declaration of each *canvas*.

```
{
  "type": "Manifest",
  "id": "https://example.org/Archive/Record1/manifest",
  "label": { "en": [ "Record 1" ], "ja": [ "記録 1" ] },

  "items": [
     // The declaration of each canvas is placed here
   ]
}
```

In practice, we often have a *manifest* as a single separate file. If creating a single separate file named `manifest.json`, it would be as follows:

- The single separate file must declare the version of IIIF with the '`@context`' at the beginning.

```
// manifest.json

{
  "@context": "http://iiif.io/api/presentation/3/context.json",

  "type": "Manifest",
  "id": "https://example.org/Archive/Record1/manifest.json",
  "label": { "en": [ "Record 1" ], "ja": [ "記録 1" ] },
```

```
  "items": [
     // The declaration of each canvas is placed here
  ]
}
```

3.2.2 Canvas

Canvas is a JSON object that represents the display area of an IIIF viewer. The following are the necessary elements, with items and annotations being essential.

- It must have a '`type`' property, and its value must be '`Canvas`'.
- It must have an '`id`' property, and its value is the URI of the *canvas* itself.
- Attach a '`label`' property to indicate the title.
  - The '`label`' of the *manifest* represents the overall title, while the '`label`' of the *canvas* represents the title of individual data.
- In '`items`' property, include declarations of *annotation pages* that should always be displayed.
- In '`annotations`' property, include declarations of *annotation pages* whose visibility can be toggled.

```
{
  "type": "Canvas",
  "id": "https://example.org/Archive/Research2020/canvas/1",
  "label": { "en": [ "Monologue 1" ], "ja": [ "モノローグ 1" ] },

  "items": [
    // Declarations of annotation pages
    // that should always be displayed are placed here
  ],

  "annotations": [
    // Declarations of annotation pages
    // whose visibility can be toggled are placed here
  ]
}
```

Additionally, the following declarations can be made as needed.

- We can specify the *canvas* size using '`width`' and '`height`' properties (in pixels). If specified, both '`width`' and '`height`' must be written.
- We can specify the duration in seconds using the '`duration`' property.

```
{
  "type": "Canvas",
  "id": "https://example.org/Archive/Research2020/canvas/1",
  "label": { "en": [ "Monologue 1" ], "ja": [ "モノローグ 1" ] },

  "width"   : "640",
  "height"  : "480",
  "duration": "120",

  "items": [
    // Declarations of annotations pages
    // that should always be displayed are placed here
  ],

  "annotations": [
    // Declarations of annotations pages
    // whose visibility can be toggled are placed here
  ]
}
```

To separate the *canvas* into an independent file (e.g., creating a file named `canvas1.json`), '`@context`' must be specified at the beginning.

```
{
  "type": "Manifest",
  "id": "https://example.org/Archive/Record1/manifest",
  "label": {"en": ["Record 1" ],"ja": ["記録 1" ] },

  "items": [
    // Declarations of each canvas are placed here
    {
      "type": "Canvas",
      "id": "https://example.org/Archive/Research2020/canvas1.json"
    }
  ]
}
```

```
// canvas1.json

{
  "@context": "http://iiif.io/api/presentation/3/context.json",

  "type": "Canvas",
  "id": "https://example.org/Archive/Research2020/canvas1.json",
  "label": {"en": ["Monologue 1" ],"ja": ["モノローグ 1" ] },

  "width"   : "640",
  "height"  : "480",
  "duration": "120",

  "items": [
    // Declarations of annotations pages
    // that should always be displayed are placed here
  ],

  "annotations": [
    // Declarations of annotations pages
    // whose visibility can be toggled are placed here
  ]
}
```

3.2.3 Annotation Page

The material (*annotation*) is not directly placed on the *canvas* but is first accumulated in the *annotation page* layer.

- It must have a ‘type’ property, and the value is ‘AnnotationPage’.
- It must have an ‘id’ property, and the value is the URI of the *annotation page* itself.
- In the ‘items’ property, list the declaration of each *annotation*.

```
{
  "type": "AnnotationPage",
  "id": "https://example.org/Archive/Research2020/page/1",

  "items": [
    // Here, the declaration of each annotation is placed
  ]
}
```

For example, we can simultaneously load the transcription subtitles of a video (`page1.json`) and the translated subtitles (`page2.json`) as separate *annotation pages*. The '`@context`' must be specified at the beginning of each independent file.

```
{
  "type": "Canvas",
  "id": "https://example.org/Archive/Research2020/canvas/1",
  "label": { "en": [ "Monologue 1" ], "ja": [ "モノローグ 1" ] },

  "items": [
    // The declaration of the annotation pages
    // that need to be permanently displayed is placed here
  ],

  "annotations": [
    // The declaration of the annotation pages
    // for switching display/unveiling is placed here
    {
      "type": "AnnotationPage",
      "id": "https://example.org/Archive/Research2020/page1.json"
    },
    {
      "type": "AnnotationPage",
      "id": "https://example.org/Archive/Research2020/page2.json"
    }
  ]
}
```

```
// Subtitle Text / page1.json

{
  "@context": "http://iiif.io/api/presentation/3/context.json",

  "type": "AnnotationPage",
  "id": "https://example.org/Archive/Research2020/page1.json",

  "items": [
    // Here, the declaration of each annotation is placed
  ]
}
```

```
// Translation Text / page2.json

{
  "@context": "http://iiif.io/api/presentation/3/context.json",

  "type": "AnnotationPage",
  "id": "https://example.org/Archive/Research2020/page2.json",

  "items": [
    // Here, the declaration of each annotation is placed
  ]
}
```

3.2.4 Annotation

The JSON object for the material (*annotation*) becomes like the following:

- It must have a '`type`' property, and the value is '`Annotation`'.
- It must have an '`id`' property, and the value is the URI of the *annotation* itself.
- '`motivation`' is specified with one of the following values:
  - If it should always be displayed on the *canvas*, use '`painting`'. *Annotations* with value must be included in the '`items`' property of the *canvas* (i.e., *annotations* that should always be displayed).
  - If it needs to be switched between display and hiding, use '`supplementing`'. *Annotations* with this value must be included in the '`annotations`' property of the *canvas* (i.e., *annotations* for switching display/hiding).
  - Other values defined by the Web Annotation Recommendation, such as '`commenting`', can also be used. The available values depend on the IIIF viewer.
- In '`body`' property, the content of the *annotation* is described.
- In '`target`' property, the same value of the '`id`' property of the *canvas* where we want to display this *annotation*.

```
{
  "type": "Annotation",
  "id": "https://example.org/Archive/Research2020/annotation/1",

  "motivation": "painting",
  "body": {
    // The entity of the annotation
  },
```

```
  "target": "https://example.org/Archive/Research2020/canvas/1"
}
```

If we make the *annotations* as independent files (e.g., creating `annotation1.json`, `annotation2.json`), the '`@context`' must be specified at the beginning.

```
{
  "type": "Canvas",
  "id": "https://example.org/Archive/Research2020/canvas/1",
  "label": { "en": ["Monologue 1" ], "ja": ["モノローグ 1" ] },

  "items": [
    // The declaration of the annotation pages
    // that need to be permanently displayed is placed here
    {
      "type": "AnnotationPage",
      "id": "https://example.org/Archive/Research2020/page/1",
      "items": [
        // Here, the declaration of each annotation is placed
        {
          "type": "Annotation",
          "id": "https://example.org/Archive/Research2020/annotation1.
json"
        }
      ]
    }
  ],

  "annotations": [
    // The declaration of the annotation pages
    // for switching display/unveiling is placed here
    {
      "type": "AnnotationPage",
      "id": "https://example.org/Archive/Research2020/page/2",
      "items": [
        // Here, the declaration of each annotation is placed
        {
          "type": "Annotation",
          "id": "https://example.org/Archive/Research2020/annotation2.
json"
        }
      ]
    }
  ]
}
```

```
// annotation1.json

{
  "@context": "http://iiif.io/api/presentation/3/context.json",

  "type": "Annotation",
  "id": "https://example.org/Archive/Research2020/annotation1.json",

  "motivation": "painting",
  "body": {
    // The content of the annotation
  },
  "target": "https://example.org/Archive/Research2020/canvas/1"
}
```

```
// annotation2.json

{
  "@context": "http://iiif.io/api/presentation/3/context.json",

  "type": "Annotation",
  "id": "https://example.org/Archive/Research2020/annotation2.json",

  "motivation": "supplementing",
  "body": {
    // The content of the annotation
  },
  "target": "https://example.org/Archive/Research2020/canvas/1"
}
```

3.2.5 Annotation Content

As mentioned above, in IIIF, the materials placed on a *canvas* are called *annotations*. The *annotation* content can take various forms, such as images, videos, audio, annotation text, caption text, etc.

The *annotation* content is described within the '`body`' section. Whether it is an external file or not, it must have a URI as an '`id`' property.

When referencing an image:

- We can load the image file using the URI provided in the '`id`' property, or (if the server supports it) you can use the Image API.
- The '`type`' must be 'Image'.

```
{
  "type": "Annotation",
  "id": "https://example.org/Archive/Research2020/annotation/1",

  "motivation": "painting",
  "body": {
    "id": "https://example.org/Archive/Research2020/content/image1.jpg",
    "type": "Image",
    "format": "image/jpeg",
    "width": "640",
    "height": "480"
  },
  "target": "https://example.org/Archive/Research2020/canvas/1"
}
```

When we are using a server that complies with IIIF Image API:

```
{
  "type": "Annotation",
  "id": "https://example.org/Archive/Research2020/annotation/1",

  "motivation": "painting",
  "body": {
    "id": "https://example.org/Archive/Research2020/content/image1/full/
640,480/0/default.jpg",
    "type": "Image"
  },
  "target": "https://example.org/Archive/Research2020/canvas/1"
}
```

When referencing a video or audio:

- Load the video or audio file using the URI provided in the '`id`' property.
- For videos, the 'type' property must have a value '`Video`', and for audio, the value must be '`Sound`'.

```
{
  "type": "Annotation",
  "id": "https://example.org/Archive/Research2020/annotation/1",

  "motivation": "painting",
  "body": {
    "id": "https://example.org/Archive/Research2020/content/video1.mp4",
    "type": "Video",
    "format": "audio"
  },
  "target": "https://example.org/Archive/Research2020/canvas/1"
}
```

When referencing caption texts for a video (WebVTT):

- Load the caption text file using the URI provided in the '`id`' property.
- The '`type`' property must have a value '`Text`'.

```
{
  "type": "Annotation",
  "id": "https://example.org/Archive/Research2020/annotation/1",
  "motivation": "painting",
  "body": {
    "id": "https://example.org/Archive/Research2020/content/tracktext1.
vtt",
    "type": "Text",
    "format": "text/vtt"
  },
  "target": "https://example.org/Archive/Research2020/canvas/1"
}
```

You can also embed annotation text directly into the body for images, videos, audio, etc.

- Provide a suitable URI as the '`id`' property.
- Set the '`type`' as '`TextualBody`'.
- Write the text in the '`value`' property. If desired, we can use HTML by specifying '`text/html`' in the '`format`' property.
- We can also specify the language using the '`language`' property.
- After the URI of the canvas referenced by '`target`',
  - We can append it with '`#xywh=0,0,20,30`' to specify that it is an annotation for the range from the top left of the canvas as the origin, with coordinates of (0, 0) to a width of 20 pixels and height of 30 pixels.
  - By adding '`#t=20,25`' you can specify that it is an annotation for the range from 20 to 25 seconds of the playback time.
  - We can also combine both by appending '`#xywh=0,0,20,30&t=20,25`'.

```
{
  "type": "Annotation",
  "id": "https://example.org/Archive/Research2020/annotation/1",

  "motivation": "commenting",
  "body": {
    "id": "https://example.org/Archive/Research2020/content/text1",
    "type": "TextualBody",
    "format": "text/html",
    "language": "en",
    "value": "I love <strong>this</strong> portion of the movie!"
  },
  "target": "https://example.org/Archive/Research2020/canvas/1#xywh=0,0,
20,30&t=20,25"
}
```

4 Creating WebVTT files

The structure of a WebVTT file is simple.

- Begin by writing '`WebVTT`' at the beginning, followed by an empty line (more precisely, one or more whitespaces).
- Write the start and end times to display the captions, connected by '`-->`'.
- Write the caption text below the time. Multiple captions can be written.

```
WEBVTT

00:00:06.697 --> 00:00:09.131
neek'aa
<i>The lion</i>
```

As the example above shows, we can embed HTML-like element tags in the caption text. The available elements are `c` (class and voice), `v` (voice), `i` (italic), `b` (bold), `u` (underline), and `ruby` and `rt` (ruby characters).

The above example can also be written as separate lines. Additionally, we can specify the display position of each caption using keywords such as line and align, and assign Identifiers to each line. Here is an example:

```
WebVTT

transcription1
00:00:06.697 --> 00:00:09.131 line:0 align:center
neek'aa

translation1
00:00:06.697 --> 00:00:09.131 line:55% align:center
<i>The lion</i>
```

Here are some examples of the main keywords:

Keywords	Meaning
`line:0` or `line:0%`	top
`line:-1` or `line:100%`	bottom
`position:0%`	left
`position:100%`	right
`size:100%`	full width
`size:50%`	half width
`align:start`	align text to the left
`align:center`	align text to the center
`align:end`	align text to the right

5 Transformation from ELAN Data to IIIF Manifest

5.1 ELAN as an Education Tool

As one of the teaching materials for mother tongue education, we are progressing with creating "captioned videos" that add subtitles such as audio description, translation, and glossary to videos. Specifically, we are working on annotating the video with annotation text for audio description and translation using ELAN (EUDICO Linguistic Annotator). As shown in Figure 2, ELAN has the characteristic of being able to annotate multiple layers of text while visually observing the waveform data of the audio. It can simultaneously overlay audio description, translation, etc., and be done for multiple speakers. Therefore, it can accommodate both monologues and dialogues.

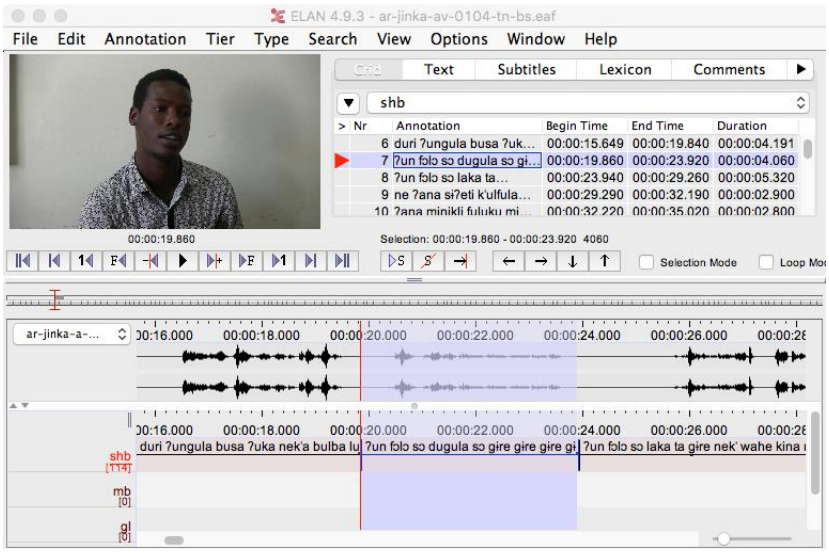


Figure 2: ELAN (Entering transcription)

However, ELAN does not have a function to create "captioned videos" itself. ELAN creates annotation texts and can convert and output them in various formats. For example, it can be converted to formats for various tools such as Toolbox (Field Linguistics Toolbox), FLEx (Fieldworks Language Explorer), Praat, and subtitle formats such as SRT (SubRip Text).

SRT format is often available on video streaming websites these days. Therefore, it is also possible to consider "captioned video" materials based on video streaming sites as a platform. However, instructional videos should have an interface where captions (annotation texts) and videos are linked bidirectionally, not just playing "captioned videos." For example, it would be preferable to be able to repeat playing scenes where specific phrases are used.

5.2 EAF

ELAN's annotation data is stored in an XML format called EAF (ELAN Annotation Format) The specifications of EAF are publicly available, and the structure can be summarized as follows.

```
<ANNOTATION_DOCUMENT ...>
  <HEADER TIME_UNITS="milliseconds" ...>
    ...
  </HEADER>
  <TIME_ORDER>
    <TIME_SLOT TIME_SLOT_ID="ts5"
               TIME_VALUE="6697" />
    <TIME_SLOT TIME_SLOT_ID="ts6"
               TIME_VALUE="9131" />
    ...
  </TIME_ORDER>
  <TIER TIER_ID="shb" ...>
    <ANNOTATION>
      <ALIGNABLE_ANNOTATION
          TIME_SLOT_REF1="ts5"
          TIME_SLOT_REF2="ts6">
        <ANNOTATION_VALUE>neek'aa</ANNOTATION_VALUE>
      </ALIGNABLE_ANNOTATION>
    </ANNOTATION>
    ...
  </TIER>
  ...
  <TIER TIER_ID="FT" ...>
    <ANNOTATION>
      <ALIGNABLE_ANNOTATION
          TIME_SLOT_REF1="ts5"
          TIME_SLOT_REF2="ts6">
        <ANNOTATION_VALUE>The lion</ANNOTATION_VALUE>
      </ALIGNABLE_ANNOTATION>
    </ANNOTATION>
    ...
  </TIER>
  ...
</ANNOTATION_DOCUMENT>
```

What should be noted is:

- Instead of directly embedding the time interval into the annotation text, the definition of time points and the text are separated.

- `TIME_SLOT` elements within the `TIME_ORDER` element define time points and always have an ID.
- The annotation text is placed within `ALIGNABLE_ANNOTATION` elements. The `@TIME_SLOT_REF1` represents the start time, and `@TIME_SLOT_REF2` represents the end time, referring to their respective time points. In the example, the annotation text '`neek'aa`' indicates that it belongs to the time interval from 6,697 milliseconds to 9,131 milliseconds.
- `TIER` elements group the annotation text.

Therefore, the annotation text should be converted to IIIF's Annotation as follows. Note that the time on *canvas* must be written in seconds.

```
{
  "type"       : "Annotation",

  "motivation" : "commenting",
  "body"       : {
    "type"     : "TextualBody",
    "value"    : "nee'kaa"
  },
  "target"     : https://example.org/Archive/Research2020/canvas/1#t=
6.697,9.139
}
```

```
{ "type"       : "Annotation",

  "motivation" : "commenting",
  "body"       : {
    "type"     : "TextualBody",
    "value"    : "The lion"
  },
  "target"     : https://example.org/Archive/Research2020/canvas/1#t=
6.697,9.139
}
```

In addition, it is also possible to obtain subtitle text in WebVTT format. We must remember the conversion from milliseconds to the format `hh:mm:ss.sss`.

```
WebVTT

00:00:06.697 --> 00:00:09.131
neek'aa

00:00:06.697 --> 00:00:09.131
The lion
```

5.3 Supplementation of missing information in EAF

Since the EAF file does not include the following information, it is necessary to supplement it as appropriate.

Most of the information that needs to be supplemented consists of the URI, an identifier for machines, and the label, an identifier for humans. URIs can be constructed systematically.

Missing	Manifest URI
Example	`https://example.org/Archive/Research2020/manifest`
Location to be supplemented	Manifest/id

Missing	Manifest name (Canvas collection name)
Example	`"Stories recorded in 2020"@en`
Location to be supplemented	Manifest/label

Missing	URI of each canvas
Example	`https://example.org/Archive/Research2020/canvas/1`
Location to be supplemented	Canvas/id

Missing	Name of each canvas
Example	`"Story 1: Lion and Ape"@en`
Location to be supplemented	Canvas/label

Missing	URI of the annotation pages for loading videos and subtitles for each canvas
Example	`https://example.org/Archive/Research2020/page/1`
Location to be supplemented	AnnotationPage/id

Missing	URI of the annotations (videos, subtitles)
Example	`https://example.org/Archive/Research2020/annotation/1`
Location to be supplemented	Annotation/id

Missing	Actual URI of the images, videos and subtitles
Example	`https://example.org/Archive/Research2020/content/video1.mp4` `https://example.org/Archive/Research2020/content/tracktext1.vtt`
Location to be supplemented	Annotation/body/id

Some of the information that needs to be supplemented includes the size of the images and videos and their playback duration.

Missing	Size of each canvas
Example	Width 640px, height 480px
Location to be supplemented	Canvas/width

Missing	Duration of the video
Example	682 seconds
Location to be supplemented	Canvas/duration Annotation/target

6 Conclusion

With the above procedures, publishing existing images, audio, and videos in IIIF-compliant format becomes possible. In particular, annotation-embedded videos created with ELAN can be subtitled in the IIIF Viewer (Mirador). What was introduced this time was a theoretical matter, but there will be a demand to develop tools to automate a sequence of tasks in the future.

Reference

Takahashi, Yona, Kiyonori Nagasaki and Jun Homma (2022) "Improvements to the IIIF viewer 'Mirador Video Annotation Support Version': Toward Integration with ELAN, a Video Annotation Tool," *Information Processing Society of Japan Research Report*, Vol. 2022-CH-129, No. 3, 1–7.

Resources

Appleby, Michael., Tom Crane, Robert Sanderson, Jon Stroop and Simeon Warner (eds.) (2020) *IIIF Image API 3.0*. <https://iiif.io/api/image/3.0/> [Accessed: 2022-04-28]

Appleby, Michael., Tom Crane, Robert Sanderson, Jon Stroop and Simeon Warner (eds.) (2020) *IIIF Presentation API 3.0*. <https://iiif.io/api/presentation/3.0/> [Accessed: 2022-04-28]

Kellogg, Gregg, Pierre-Antoine Champin and Dave Longley (eds.) (2020) JSON-LD 1.1, W3C Recommendation. <https://www.w3.org/TR/json-ld11/> [Accessed: 2022-04-28]

Max Planck Institute for Psycholinguistics. *The Language Archive: ELAN*, <https://archive.mpi.nl/tla/elan> [Accessed: 2022-04-28]

Max Planck Institute for Psycholinguistics (2017) *The Language Archive: ELAN Annotation Format: Schema version 3.0*. <https://www.mpi.nl/tools/elan/EAF_Annotation_Format_3.0_and_ELAN.pdf> [Accessed: 2022-04-28]

Pfeiffer, Silvia (ed.) (2019) *WebVTT: The Web Video Text Tracks Format*, W3C Candidate Recommendation. <https://www.w3.org/TR/2019/CR-webvtt1-20190404/> [Accessed: 2022-04-28]