GDPR Annotation
Technical Annotation Guide

François Lévy, Adeline Nazarenko

Introduction

Goal of the annotation

The semantic annotation of the GDPR aims to facilitate the exploration of the text for the lawyer and to quickly identify the provisions related to a given subject, related to a particular actor or related to another provision. Actually, advanced search functionalities can be based on the segmentation of regulations into elementary provisions that are semantically typed, on the annotation of the roles that key entities play in these provisions, and on the organization of these provisions into a semantic network.

The objective is to facilitate the analysis and interpretation work of legal experts by allowing them to explore the text. Using queries combining textual patterns and semantic key elements associated to the text thanks to the annotation, one can find a list of relevant textual fragments, each one being placed in its structural and semantic context. Other navigation or exploration devices are also possible.

Annotation campaign and task

An annotation campaign is an annotation work that usually mobilizes several annotators and aims at obtaining a reference annotation (or gold annotation) for a given corpus. The annotators have an annotation task to perform and must conform to the annotation instructions detailed in an annotation guide. They work in parallel and can make different proposals but adjudication steps allow them to converge towards consensual solutions. The reference annotation is the result obtained after annotation and adjudication.

Some annotators are in charge of the adjudication: they confront the various annotators' annotation proposals and try to resolve their disagreements to obtain a new consensus annotation that will be the reference annotation.

In the GDPR annotation campaign, the annotation task itself consists in describing the semantics of the elementary dispositions that are pre-annotated as plain fragments. After reading those dispositions, the annotator has mainly to

1. associate a semantic type to them (e.g. obligation, power, permission),
2. identify the entities that play key roles in them,
3. link the provisions that have semantic relationships,
4. validate the resulting annotation.

In concrete and short terms, the annotators are provided with

- the source pre-annotated text to annotate[1],
- the semantic annotation guide that describe the legal semantic annotation language to be used,
- the present technical annotation guide that explains how to perform the annotation work,

and they output the source text enriched with their semantic annotations.

Annotation example

Figure 1 presents the overall structure of the GDPR file to be annotated and the localisation in that structure of the 2nd paragraph of Article 6. Figure 2 shows a pre-annotated version of that paragraph and Figure 3 the same paragraph once it is annotated.

[1] Additional files are provided in practice (see Starting for details).

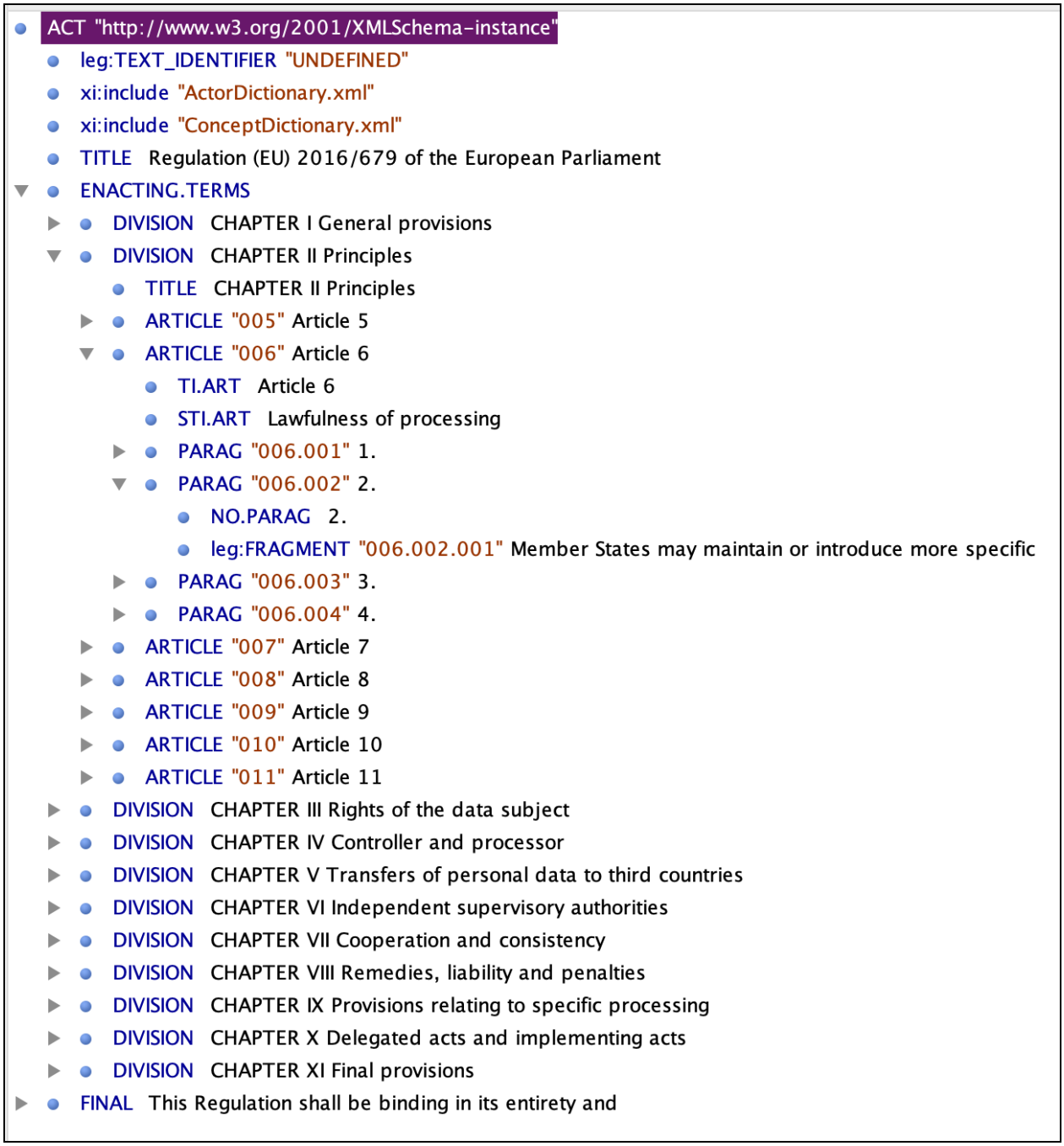


Figure 1. *XML tree of the RGPD file to be annotated*

```
        <PARAG IDENTIFIER="006.002">
          <NO.PARAG>2.</NO.PARAG> <leg:FRAGMENT IDENTIFIER="006.002.001">Member
States may maintain or introduce more specific provisions to adapt the application
of the rules of this Regulation with regard to processing for compliance with
points (c) and (e) of paragraph 1 by determining more precisely specific
requirements for the processing and other measures to ensure lawful and fair
processing including for other specific processing situations as provided for in
Chapter IX.</leg:FRAGMENT> </PARAG>
```

Figure 2. *Pre-annotated version of Article 6, Paragraph 2*

```
          <NO.PARAG>2.</NO.PARAG> <leg:POWER IDENTIFIER="006.002.001"
type="ruling" bearer="le_MS">Member States may maintain or introduce more specific
provisions to adapt the application of the rules of this Regulation with regard to
processing for compliance with points (c) and (e) of paragraph 1 by determining
more precisely specific requirements for the processing and other measures to
ensure lawful and fair processing including for other specific processing
situations as provided for in Chapter IX.</leg:POWER> </PARAG>
```

Figure 3. *Annotated version of Article 6, Paragraph 2*

The comparison of figures 2 and 3 shows what work the annotators have to do. It mainly consists in specifying the semantics of the elementary provisions which are annotated as neutral fragments in figure 3 (`leg:FRAGMENT` XML elements). Article 6, Paragraph 2 contains a single such fragment. The annotator has not changed either its boundaries (the position of the start and end tags) or its identifier (`IDENTIFIER="006.002.001"`) which have been computed during the pre-annotation phase. However, three modifications have been introduced:

- The fragment has been tagged as a power: the name of the element has been changed from `leg:FRAGMENT` to `leg:POWER` both in the start tag and in the end tag,
- The executive type of this power is marked by adding a type attribute (`type="ruling"`),
- The addition of an additional attribute indicates that this power is held by the “Member States”, a legal entity which is identified as `le_MS` in the actor dictionary (`bearer="le_MS"`).

Fortunately, the XML editor facilitates the work of the annotator, so that these changes can be made quickly and in a controlled manner, thus avoiding errors.

Prerequisites

Basics of XML

Here is a short introduction to XML containing what annotators need to know for annotating the GDPR. An extended tutorial of XML can be found on the website of the World Wide Web Consortium (W3C).

Extensible Markup Language (XML) is a markup language used for encoding documents in a normalized and interoperable format. It is a textual data format and that is widely used for documents, even if it applies to arbitrary data structures.

XML is a markup language that uses tags to define elements within a document like HTML but it is designed to carry and control data (what is the type of data, how it is organised), whereas HTML is designed to display data (how data looks).

XML documents (xml files)

An XML document is organised as a tree of XML elements. In the GDPR case, the root element is `ACT` (see Figure 1).

As shown in the following example, each element has a name, an opening and a closing tag, possibly some attributes and a content:

- The **name** of the element, in the following example, is `POWER`, prefixed by `leg:` to indicate that it belongs to the legal semantic namespace.
- The **opening and closing tags** correspond to each other. They are respectively of the form `<xxx>` and `</xxx>` and can be grouped as a unique `<xxx/>` tag if the element has an empty content.
- In the example below, the opening tag also contains 3 **attribute-value pairs** in the form `xxx="yyy"`, which specify the semantics of the element and the text segment that it contains.
- The **content** is all that appears between the opening and closing tags. It can be plain text as in the example below ("Member States may… as provided for in Chapter IX") but it can also contain XML elements, in which case it is said to be mixed.

```
<leg:POWER IDENTIFIER="006.002.001" type="ruling"
bearer="le_MS">Member States may maintain...</leg:POWER>
```

Except for the fact that two elements cannot overlap (they are either separate or included in each other), the standard is very flexible.

XML is also designed as a generic formalism: allowing different specific languages to be defined for different applications. Usually, the definition of the language is provided by a distinct file, named a schema (it is not part of the XML document).

XML schemas (xsd files)

An XML file is thus usually associated with one or several XML schemas (`.xsd` files) that specify the vocabulary to be used for elements and attributes, the constraints on the order and embedding of elements as well as on the appearance and the values of attributes.

More precisely, a schema describes:

- which XML elements may be utilized in the XML file,
- what specific attributes each element must or may have and the values authorized for each attribute,
- where (*i.e.* inside the content of what element), in what order and how often an element may appear.

In the GDPR annotation campaign, however, annotators do not need to know about these schema files as the vocabulary to be used for annotation and the rules to be respected are described in the semantic annotation guide (where there is a specific section dedicated to each language element).

Annotators simply have to make sure that the XML and XSD files are stored in the same directory and to verify that their annotations respect the schema by checking the validity of their XML file in Oxygen (see Starting)

Oxygen XML Editor

It is possible to read and modify an XML file using a plain text editor such as notepad (Windows) or TextEdit (OSX), provided that the editor does not add hidden formatting characters. The task is nevertheless much easier using a specialized editor which has many helping features. Oxygen XML Editor (Oxygen for short) automatically reads the schema associated[2] to an XML file, and uses this schema to:

- suggest at writing time which labels and attributes may be used at the current cursor position,
- check on the fly if the XML file is presently conformant to its schema and warn the user of any detected error,
- ease modifications: *e.g.* a change in the label of the opening tag automatically triggers the corresponding change in the closing tag.

Oxygen also provides facilities such as a summary tree view of the text, a quick search tool and folding/unfolding flags which help navigation in the text.

A separate guide helps to install Oxygen. The following Oxygen XML Editor explains how to start working with it.

Starting

Annotation task installation

You need to

[2] The first tag of the XML document establishes the connection with the appropriate schema.

- have the Oxygen XML Editor already installed, which is quite simple following the provided installation guide.
- create your working directory or your personal computer and download the following XML files:
  - `Ligth_GDPR_EN_STi_working.xml` to perform the i-th annotation subtask
  - `ActorDictionary.xml`
  - `ConceptDictionary.xml`

  as well as the following schema files:

  - `GDPR_SemanticSchema.xsd`
  - `LightS_SemanticSchemaWrapper_working.xsd`
  - `LightS&S_MergedSchema_working.xsd`
  - `LightS_SemanticSchemaWrapper.xsd`
  - `LightS&S_MergedSchema.xsd`

  from the relevant Assessment or Sub-Task folder in SPIN Annotations Internships/Annotation Campaign.

  The `README` file of the Resources folder explains what those files are but you do not need to worry about that: Oxygen will use them under the hood.

Overview of the GDPR in XML format

`Light_GDPR_EN_STi_working.xml` contains the text of the GDPR enriched with a light version of the structural annotation. Its structure is the following :

```
1. <ACT …. xsi:schemaLocation="... Working_LightS&amp;amp;S_MergedSchema.xsd" … >
2.    <leg:TEXT_IDENTIFIER IDENTIFIER="UNDEFINED"></leg:TEXT_IDENTIFIER>
3.    <xi:include href="ActorDictionary.xml" …"></xi:include>
4.    <xi:include href="ConceptDictionary.xml" …></xi:include>
5.    <TITLE>...</TITLE>
6.    <ENACTING.TERMS> …. </ENACTING.TERMS>
7.    <FINAL> … </FINAL>
8. </ACT>
```

The first tag is `<ACT …>`. The attribute `xsi:schemaLocation` associates the `LightS&S_MergedSchema_working.xsd` schema file to the XML document[3]. Lines 2-4 create the `UNDEFINED` reference and include the dictionaries. This is followed by the title of the act, the enacting terms on which annotation focuses and the final section. Line 8 corresponds to the closing `ACT` tag, the whole document constituting the content of the `ACT` element.

Lawyers will notice that visas and recitals are omitted: annotators are not expected to annotate them.

[3] `&` is the xml special notation to write the ampersand character.

Inside the enacting terms, `<DIVISION>` tags embed both chapters and sections. Inside these are articles (in `<ARTICLE>` tags) which may directly contain text or be further subdivided in numbered paragraphs (`<PARAG>` tags) which contain text. Note that the text may in turn contain a lower level paragraph (`P`) and lists (`LIST`) which enumerate items (`ITEM`). Numerous examples figure in the semantic annotation guide.

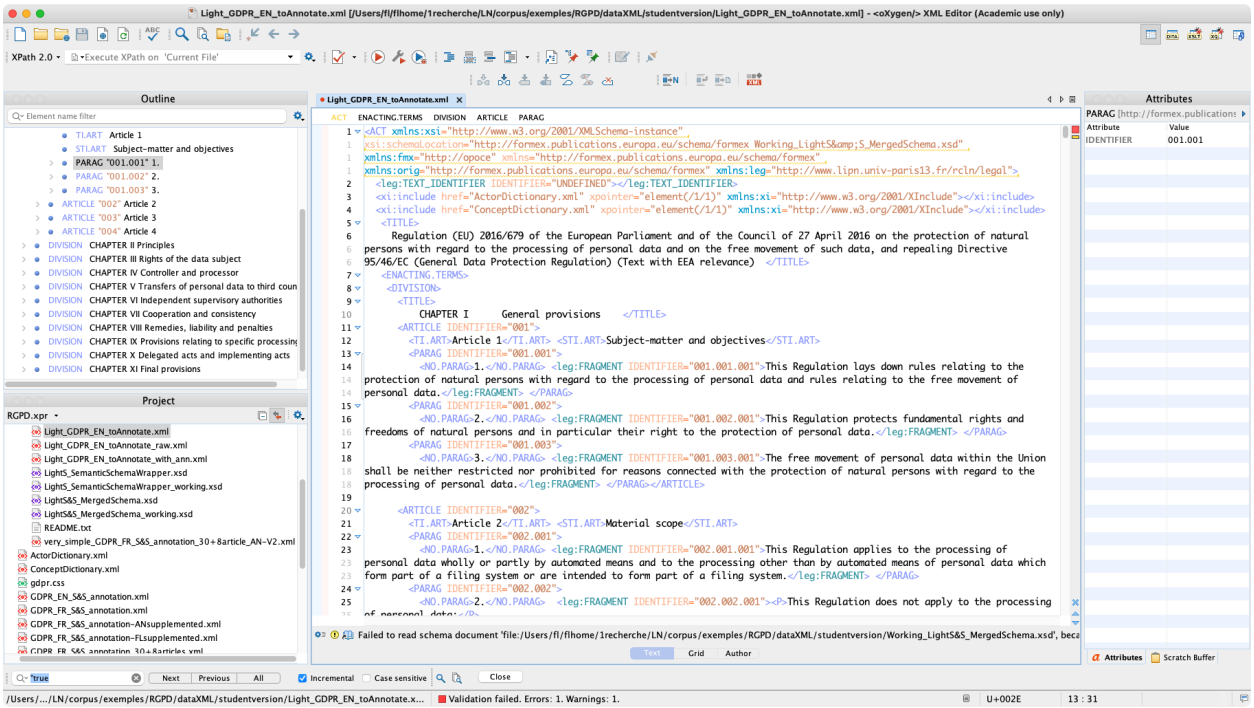
Getting started with Oxygen

Once the XML and XSD files are copied in your working directory, launch Oxygen XML Editor.

Before starting to work, you must set up your Oxygen application. To do this,

- Open the preferences window via “Oxygen XML Editor” menu (on OS X) or in the Window of Help menu (for Windows),
- Open the "Editor / Edit Modes / Text" sub-window from the tree menu on the left panel,
- Activate the following options: “Highlight current line”, “Show line numbers”, “Line wrap”, “Enable folding” and “Highlight matching tag” ; deactivate the other options as shown on the following window shot.
- Open the "Editor / Document Validation” subwindow.
- Activate the “enable automatic validation” option.
- Click OK to close the preferences window.

Open `Light_GDPR_EN_STi_working.xml` (according to the current step). What you see should look like the following picture :



The XML file is edited in the middle panel. The upper left panel presents the outline of the document in the form of an XML tree. Note that you can unfold an item by clicking its left '>' sign, and select it in the editor by clicking the item in the outline.

You can also fold or unfold the content of an annotation in the editor, by clicking the small triangle on the right of its line number. Try it for Article 2 (line 20).

Last, at the bottom of the left column is a small text field used for quick search. Go there and type "053" (including the quote characters). You directly find Article 53 (because no other `IDENTIFIER` has the value 053). Advanced search functionalities are also accessible in the Find menu.

In the preference window, you can also tune your visual environment. Choose `Editor/Syntax Highlight` and select XML in the right upper pane. Click the > sign. Now you can change colors of tags, attributes and text if you wish. In the editing window, an icon in the upper icon bar of the window allows users to switch between different contrasts of tags and text.

First example

Take as example Fragment `045.009.001`, the annotation of which is analyzed in the annotation guide (Annotation of fragments/subordinate fragments/COMPLEMENT/Examples(procedure).

First, find the fragment identifier using the quick search tool. Its tag should be `<leg:COMPLEMENT>`. So select the `FRAGMENT` in the opening tag, and type `COMP` (which replaces the selection). Then type control-SPACE (press the control key and type the spacebar): this asks for suggestions and, as there is only one, completes the answer. Note first that the closing tag is also changed, and second a light bulb appears on the left. This light bulb means that the syntax is not correct. There is also an error message down the panel.

Leaving the pointer on the bulb or the error message displays a pop-up with explanations (be patient). You immediately know that `type` and `rel` attributes are lacking. Put the cursor between the identifier and the tag-closing ">" and type a space. A pop-up menu presents the names of the various allowed attributes , among which you can select `type`. A set of values is immediately proposed : select `procedure`.

Now type in one more space and choose `rel` in the pop-up menu. No other suggestion appears and you have to type `045.003 045.005` inside the quotes The error messages now disappear. For a last check, type a space just before the tag-closing ">". The remaining options are `except`, `has_list_header`, `is_list_header`, none of which is relevant. So delete the space and the annotation is done.

Before moving to a different fragment, click on the validation "check-in" button on the top menu and check that your XML file conforms to the schemas. A green "successful validation" message should appear at the bottom of the main window.

Organization of the annotation campaign

An annotation campaign involves **several types of actors** and is organised in an alternati**on of annotation subtasks and adjudication steps,** according to a schedule set up by the annotation managers. See Figure 1.

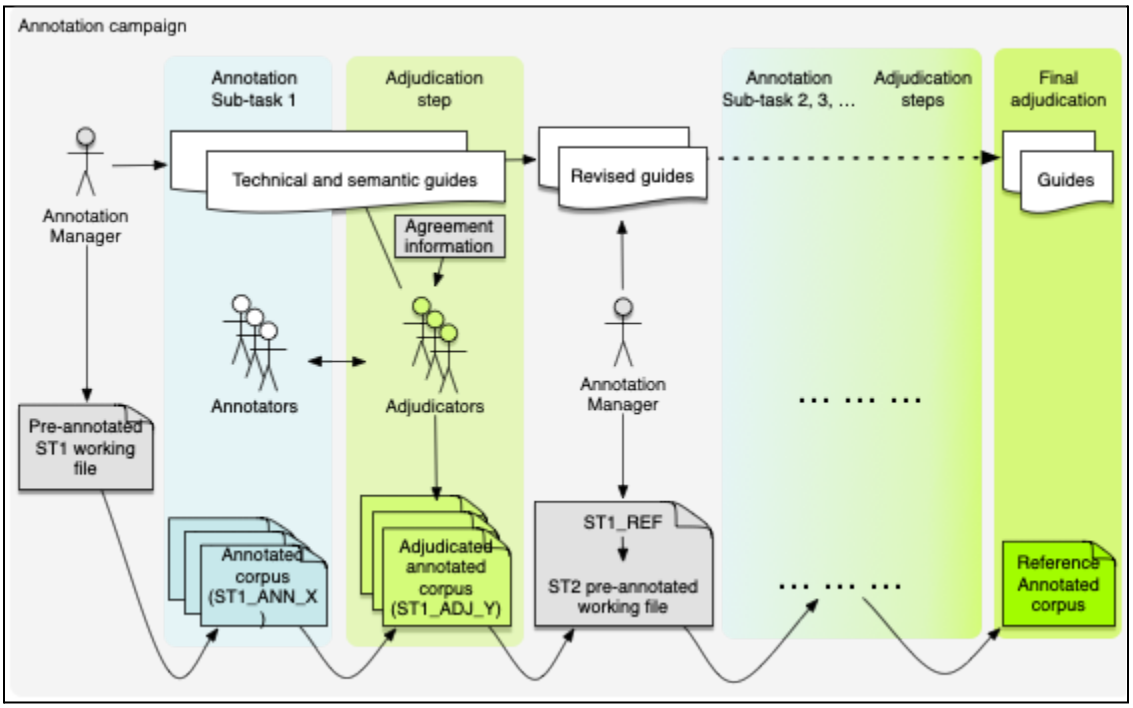


Figure 1. *Organisation of an annotation campaign*

Actors

The **actors** of an annotation campaign are :

- the annotation managers who organize and supervise the campaign,
- the annotators who annotate the source text and
- the adjudicators who propose a reference annotation from the potentially different proposals made by the different annotators.

The annotation subtasks are performed in parallel and independently by the different annotators while the adjudication work is done collectively by the annotators and the adjudicators, under the responsibility of the latter.

File and folder management

Many files are managed during the annotation campaign. They are organized in different folders corresponding to the different annotation sub-tasks, with an additional folder for the assessment.

Each of the sub-task folders contains 4 sub-folder (for the i-th sub-task, all the files have the `STi` suffix) :

- The starting folder, which contains the annotators' working files (XM) and the schemas:
- The Annotators' output folder where the annotators upload the result of their annotation work (files with the `ANN` and the annotator's Surname suffixes).
- The Adjudicators' output folder, where the adjudicators upload the result of their adjudication (files with the `ADJ` and the adjudicator's Surname suffixes).
- The sub-task output folder contains the reference files (1 annotated XML file and two dictionaries with the `REF` suffix).

Annotation and adjudication processes

For each **annotation sub-task** (*STi*):

- The managers prepare the annotation work in the form of an XML file (`Light_GDPR_EN_STi_working.xml`) where the passages to be annotated are pre-annotated: the provisions are broken down into elementary provisions that are tagged with a neutral `leg:FRAGMENT` element and simply associated with an identifier (`IDENTIFIER`).
- Each annotator proceeds to the annotation by semantically characterizing the elementary provisions identified as `leg:FRAGMENT` elements in the source file (`Light_GDPR_EN_STi_working.xml`).
- Once the annotation work is completed, each annotator checks the validity of his annotated file, saves it under the name (`Light_GDPR_EN_STi_Surname.xml`) and uploads it on `Sub-Task i/1.2 Outputs of the Annotators`.

At each **adjudication step**:

- The managers compare the files of the different annotators, compute an agreement score and provide the adjudicators with the list of fragments that are not annotated in the same way by the different annotators (points of disagreement).
- Adjudicators review the annotations provided by the annotators, especially (but not exclusively) the points of disagreement.
- In a group meeting, the adjudicators discuss with the annotators how to resolve the disagreements and produce a consensus annotation (`Light_GDPR_EN_STi_ADJ_Surname.xml` file). If there are multiple adjudicators, each one produces its own adjudicated file.
- The managers check that the different adjudicators' files are identical, correct any remaining discrepancies and/or disagreements, provide the reference annotation

(`Light_GDPR_EN_STi_REF.xml`) and possibly update the semantic and technical annotation guides following the recommendations resulting from the adjudication work.

The `Light_GDPR_EN_STi_REF.xml` file is used to create a new `Light_GDPR_EN_STi+1_working.xml` file where managers identify additional passages to annotate if a new annotation sub-task (*STi+1*) is planned. Otherwise, it is the final reference annotated file (`Light_GDPR_EN_REF.xml`).

As the campaign progresses, annotators should in principle converge more easily showing increasing agreement scores with a reduction in the number of points of disagreement. The guides stabilize in parallel.

Annotation work

The annotation is done iteratively, as the text is read, considering the pre-annotated neutral fragments (elements tagged as `leg:FRAGMENT` in the working file) one by one and specifying their semantic values. As long as the annotation task is in progress, the correctness of the XML file should be checked with the working schema (LightS&S_MergedSchema_working.xsd). At the end of an annotation task, the file should no longer contain any neutral fragments but only fully characterised semantic fragments and the file should be validated with the standard XML schema (LightS&S_MergedSchema.xsd).

Characterizing a fragment

The boundaries and the identifier of the fragment tagged as `leg:FRAGMENT` should not be modified (placement of the start and end tags and `IDENTIFIER="..."` attributes) but the annotator has to associate a semantic type and add the related attributes to those fragments.

The annotator may also add a comment to the adjudicators, mark-up a relevant sub-fragment and enrich the dictionaries with new entities.

Finally, it is recommended to do an intermediate validation of the file each time a new fragment is annotated.

Choosing a type fragment

Just replace `FRAGMENT` by the type you choose (type `Control-space` to get propositions)

Adding attributes

In the opening tag, place the cursor at the end, just before “>”, and type in a space. Choose in the popup menu.

Adding a comment for further discussion

Anywhere inside the fragment, type the opening "<". In the popup menu, select leg:COMMENT, so Oxygen writes `<leg:COMMENT></leg:COMMENT>` and the cursor is in between the tags. Just type your comment. If it is long or repeated in several places, you can also type "see note nnn" and gather numbered notes in a separate text file.

Validating the new fragment

If the option "automatic validation" is not selected, click the "validate" checkbox in the upper icon bar.

Handling specific cases

- Marking up a subfragment : see Adding a subfragment to a fragment
- Referring to a new entity : see Adding an entity in the dictionary

Adding a subfragment to a fragment

Subfragments are not pre-marked, because they need to be first detected and (contrarily to sentences), only the annotator's semantic understanding can decide that some part of a sentence is a subfragment - presently an exception. Once that is decided, when there is no list in the fragment, the technical operation is easy :

- First case (the more frequent): marking one subfragment in a fragment
    - Select exactly the subfragment
    - Click the icon "frame with tags" in the upper icon bar, or use the shortkey Command-D
    - Choose the label `leg:EXCEPT`

That is the basic operation, which marks one except. Nevertheless, when the fragment embeds a list, the operation becomes slightly more elaborate. Two cases arise:

- Second case: the whole list is covered by two fragments, the first with the attribute `is_list_header="true"`, the second with `has_list_header="xxx.xxx.xxx"`, and the exceptional sub-fragment meets both. In that case, mark one `EXCEPT` in each fragment, following the procedure of the first case. Complete the former with the attribute `is_except_list_header="true"`, and the latter with `is_except_list_items="true"`. Note that the second `EXCEPT` generally scopes over its whole fragment.

- Third case: the whole list is covered by one single fragment, which contains the header inside a `P` structural element, and then the `LIST` element. If you detect an exceptional sub-fragment which meets both the `P` and the `LIST`, you must first split the fragment :

- Copy the attributes of the opening tag
- Delete the fragment tags (use the icon “delete tags” in the upper icon bar)
- Select the content of the P structural element (without the `P` tags) and click the “surround with tags” icon
- Choose the same label as was in the initial fragment
- Paste the copied attributes
- Select from after `</P>` to the end of the list (after `</LIST>` and click the “frame with tags” icon
- Choose the same label as was in the initial fragment
- Paste the copied attributes. Add 1 to the identifier
- Add the `is_list_header=”true”` and `has_list_header=`[identifier of the header] at their respective place

You can now continue following the procedure of the first case.

Adding an entity in the dictionary

You need to add an entity in the dictionary when, during the annotation of a fragment, you know which actor fills a role but this actor does not appear in the dictionary. Then :

- open the ActorDictionary.xml file and add a new line after the *persons* or the *legal entities* bloc, according to the kind of actor you have identified (legal entities are in charge of the creation or the enforcement of law).
- just type `<` and choose the relevant type of entity in the popup menu. Type a space and accept the `id` attribute name proposed. Then a popup presents the list of existing id’s ; **choose none of them** and create a new one, following the pattern of the block (`p_xxx` or `le_xxx`).
- move between `>` and `<`, for instance by keying the right arrow twice. Type a `<` and accept `LABEL`. Type a space, choose `lang` in the popup, and `EN` in the next popup. Then another space yields `value`, and last you need to type a name or an expression characterizing the entity.
- move after the ‟ character and type `/`. This will yield a shorter form of the element (because its content is empty). You can add other `LABEL` elements if there are several expressions characterizing the element.
- do not forget to save the dictionary !!

Validating the annotations (at the end of a sub-task)

In the preferences window, the “Editor/Validation of the document” item of the left panel displays some choices, among which “Activate automatic validation”. This checkbox is generally ticked, so that error messages appear on the fly. You can, if you prefer, turn this choice off. Validation will then have to be manually triggered. Just click the “Validate” icon in the upper icon bar. And when you get a red light, use the error messages to have an idea of the problem.

Adjudication work

Input files

Each adjudicator collects

- The annotations proposed by the different annotators (the various `_STi_Firstname.xml` files, where *i* is the number of the sub-task adjudication step),
- The agreement information delivered by the annotation managers:
    - The agreement score,
    - The list of annotations on which two or more annotators disagree.

Discussion

Based on their own analysis of the regulation and the recommendations of the annotation guide, the adjudicators discuss with the annotors to understand their points of disagreement and propose a solution.

Each adjudicator creates his or her adjudication file (`_STi_ADJ_Firstname.xml`) from any of the annotators' files and corrects it based on the solutions that emerge from the discussion with the annotators. In the end, all annotators and adjudicators should agree on the proposed annotation. Comments left by annotators must also be resolved.

In the meantime, the adjudicators record the main points of discussion, explain the proposed solutions and may propose new or revised annotation instructions/recommendations for the updating the annotation guides by the managers.

When all the issues and annotators' comments are resolved, the adjudicators must check the validation of their adjudication files against the XML schema (`LightS&S_MergedSchema.xsd`).

Results

Each adjudicator issues

- A validated adjudication file (`Light_GDPR_EN_STi_ADJ_Surname.xml`),
- A report of the disagreement discussed and the proposed solutions, with possible suggestions to improve the guide and reduce discrepancies between annotators.

The managers

- Check that the files delivered by the different adjudicators are identical and resolve the remaining issues if necessary,

- Update the annotation technical and semantic guides
- Discuss the adjudicators' suggestions and update the annotation technical and semantic accordingly,
- Deliver a new stabilized version of the annotated file (`Light_GDPR_EN_STi_REF.xml`),
- Deliver a new working file if a new annotation sub-task is planned (`Light_GDPR_EN_STi+1_working.xml`).