

Apache UIMA Cas Editor Documentation

Authors: The Apache UIMA Development Community

Version 2.2.2

Incubation Notice and Disclaimer. Apache UIMA is an effort undergoing incubation at the Apache Software Foundation (ASF). Incubation is required of all newly accepted projects until a further review indicates that the infrastructure, communications, and decision making process have stabilized in a manner consistent with other successful ASF projects. While incubation status is not necessarily a reflection of the completeness or stability of the code, it does indicate that the project has yet to be fully endorsed by the ASF.

License and Disclaimer. The ASF licenses this documentation to you under the Apache License, Version 2.0 (the "License"); you may not use this documentation except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, this documentation and its contents are distributed under the License on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

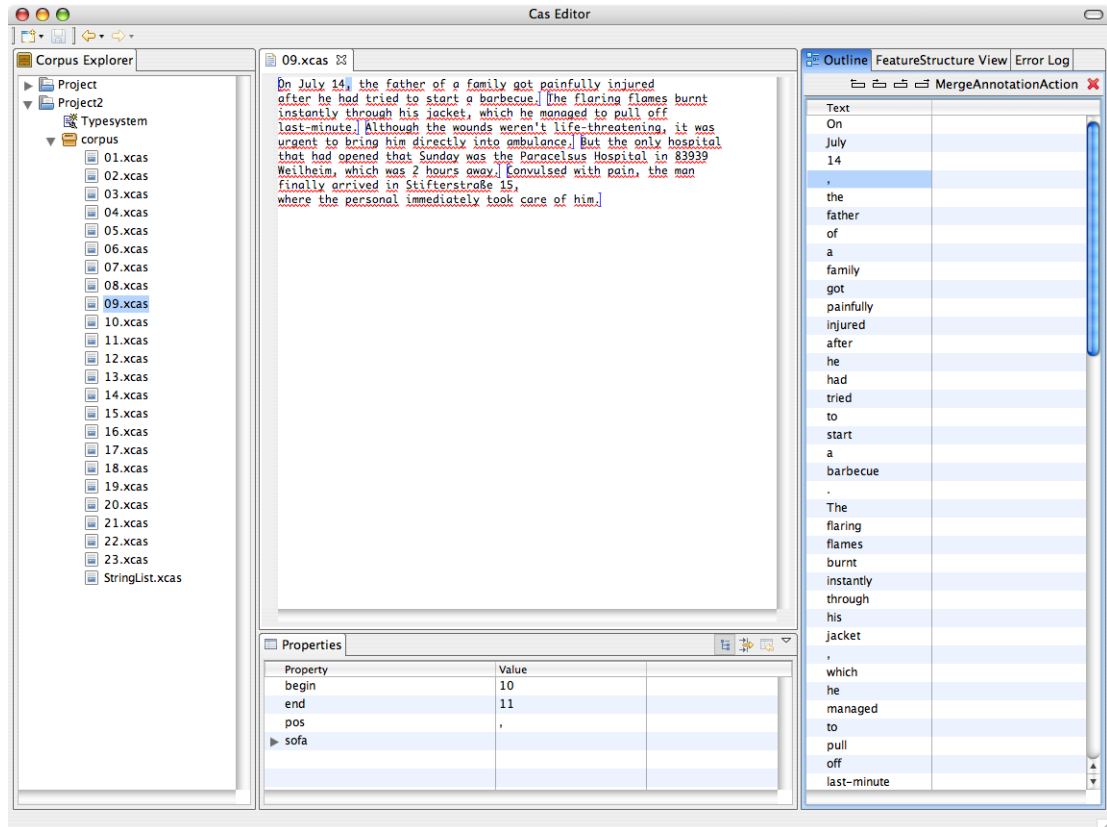
Trademarks. All terms mentioned in the text that are known to be trademarks or service marks have been appropriately capitalized. Use of such terms in this book should not be regarded as affecting the validity of the the trademark or service mark.

Table of Contents

1. Introduction	1
2. Projects	3
2.1. Project structure	3
2.2. Add a type system	4
2.3. Add corpus folder	5
3. Annotation editor	7
3.1. Editor	7
3.2. Outline view	8
3.3. Edit Views	8
3.4. FeatureStructure View	9
4. Cas processor integration	11

Chapter 1. Introduction

The CAS Editor is an annotation tool which supports manual and automatic annotation (via running UIMA annotators) of CAS files. Currently only text-based CAS are supported. The CAS Editor can visualize and edit all feature structures. Feature Structures which are annotations can additionally be viewed and edited directly on text.



Chapter 2. Projects

The CAS Editor organizes all artifacts in one or more projects. It is not possible to open artifacts which are located outside of a project.

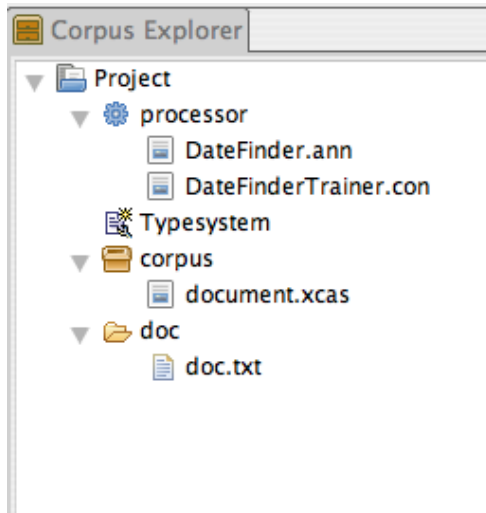
2.1. Project structure

A project includes these elements:

- **Type system** The type system must be present for opening a CAS file or running a CAS processor.
- **Corpus folder** A corpus folder is a collection of CAS files in the project. A project can have multiple corpus folders.
- **CAS file** The CAS itself. It must be located in a corpus folder and must end with ".xmi" or ".xcas" to be recognized as a CAS file.
- **CAS Processor folder** A processor folder contains Analysis Engine and CAS Consumer Descriptors. The CAS processor folder is also put on the data path for the processors when they are run. A project can have multiple processor folders.
- **Analysis Engine Descriptor** Configuration for an Analysis Engine which can be used to annotate CAS files in a corpus folder. To be recognized as Analysis Engine Descriptor the file must end with ".ann" and must be placed in a processor folder.
- **Consumer Descriptor** Configuration for a Consumer which can be fed with the CAS files in a corpus. To be recognized as Consumer Descriptor the file must end with ".con" and must be placed in a processor folder.

These elements are shown differently than normal files and folders in the corpus explorer view. In addition to the listed elements a project can also contain files and folders e.g. for documentation. If one of these special elements contains an error, a marker which describes the problem is added to the file.

The corpus explorer with a project looks like this:

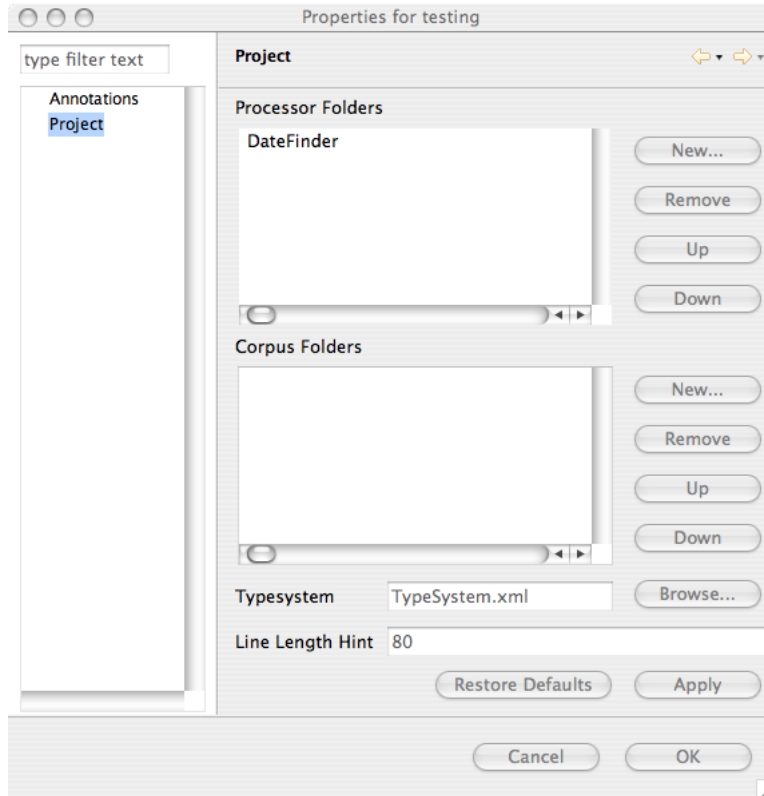


2.2. Add a type system

It's strongly recommended to first add a valid type system to the project; other functions are only available if the type system is present. Use copy and paste to import an existing type system. Editing of the type system is currently not supported inside the CAS Editor.

Note: If an external text editor is used to modify the type system, close all editors inside the CAS Editor and run the refresh action.

After the type system file is added, you need to make the CAS Editor aware of its existence. To do this open the Properties dialog for the project and then select the type system as shown here:



Now the new type system element can be seen in the project tree of the corpus explorer.

2.3. Add corpus folder

To add a corpus folder first create a new folder. Then open the Properties dialog and add the folder to the list of corpus folders. It then appears as a corpus folder in the corpus explorer.

The corpus explorer automatically hides all non-CAS files in the corpus folder. The CAS files are organized in a flat hierarchy; sub folders which contain CAS files are not shown.

Chapter 3. Annotation editor

The annotation editor shows the text with annotations and provides different views to show aspects of the CAS.

3.1. Editor

The editor has an associated, changable CAS Type. This type is called the editor "mode". By default the editor only shows annotation of this type. Actions and views are sensitive to this mode. To change the mode for the editor, use the "Mode" menu in the editor context menu.

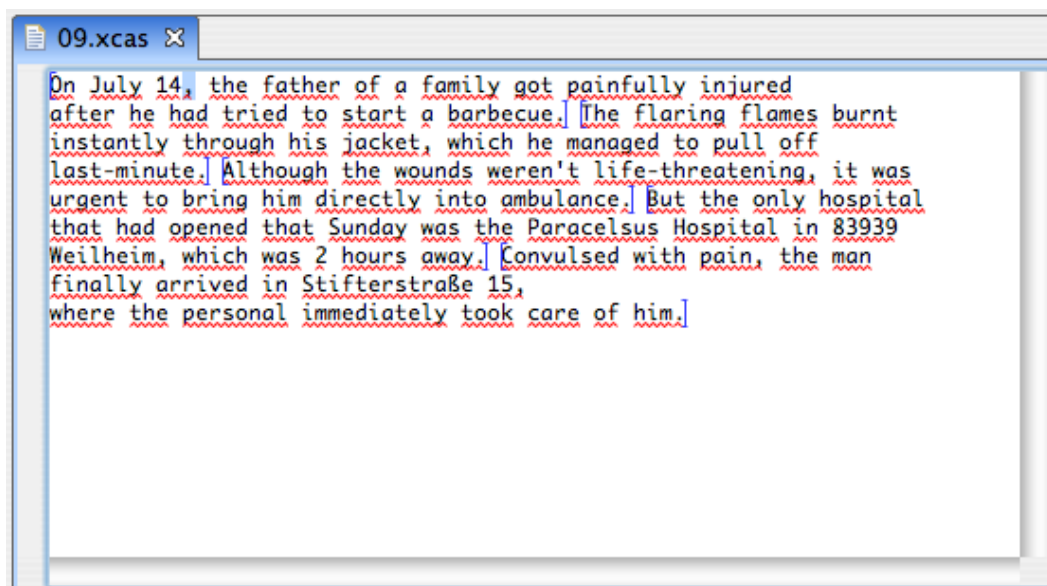
The editor can also show annotations of other Types. To do this, use the "Show" menu in the context menu. The annotation renderer and rendering layer can be changed in the Properties dialog. After the change all editors should be re-opened.

The editor automatically selects annotations of the editor mode Type that are near the cursor. This selection is then synchronized or displayed in other views.

To create an annotation manually using the editor, mark a piece of text and then press the enter key. This creates an annotation of the type of the editor mode, having bounds corresponding to the selection.

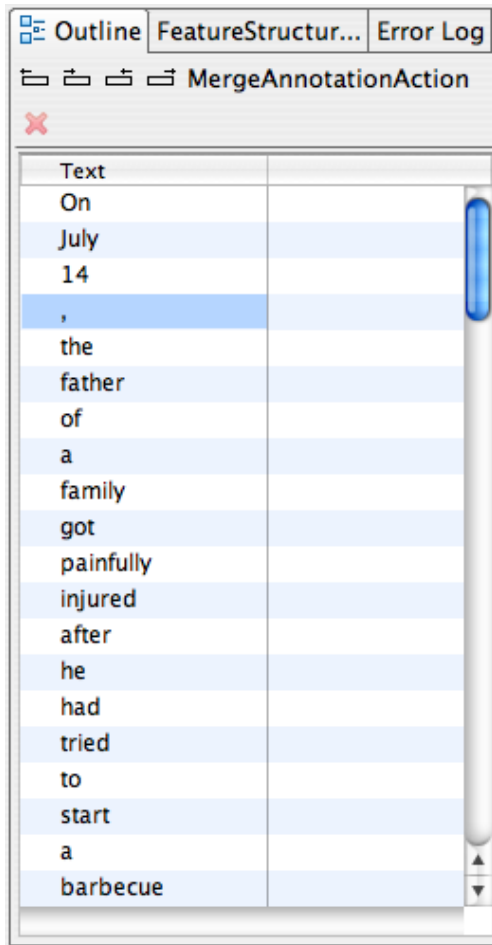
It is also possible to choose the annotation type; press shift + enter for this. Then a dialog asks for the annotation type to create.

To delete an annotation select it and press the delete key. Only annotations of the editor mode can be selected.



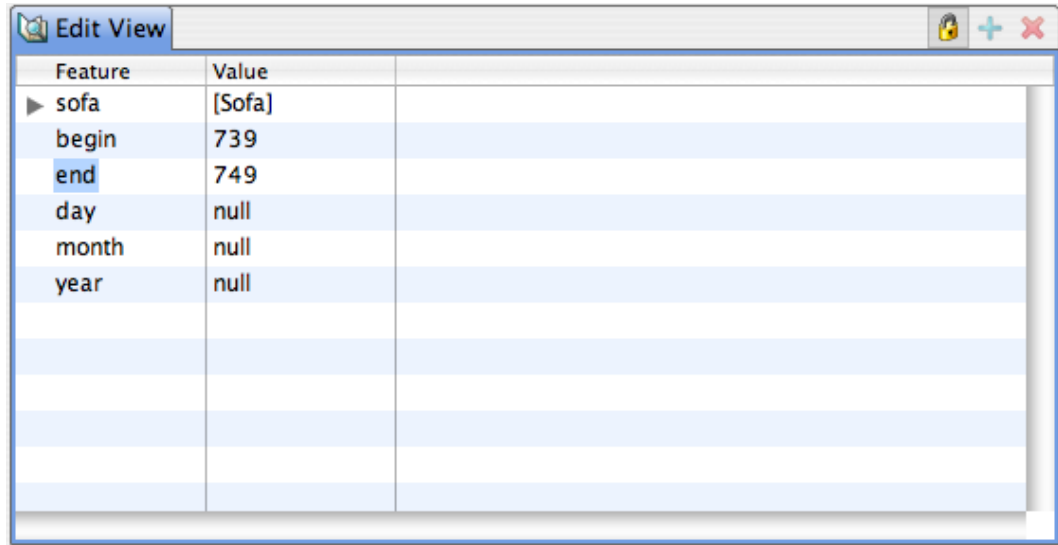
3.2. Outline view

The outline lists the annotations of the current editor mode. There are actions to increase or decrease the bounds of the selected annotation. There is also an action to merge all selected annotations.



3.3. Edit Views

The Edit Views show details about the currently selected annotations or feature structures. It is possible to change primitive values in this view. Referenced feature structures can be created and deleted including arrays. To link a feature structures with other feature structures it can be pinned to the edit view. This means that it does not change if the selection changes.



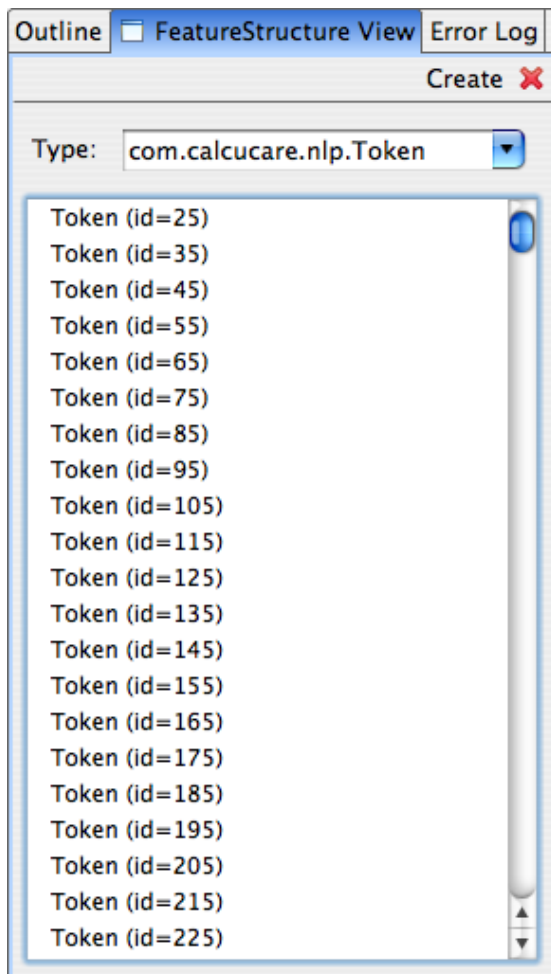
The screenshot shows a window titled "Edit View" with a table of feature structures. The table has two columns: "Feature" and "Value". The first row is expanded, showing a tree structure for the feature "sofa". The "sofa" feature has a value of "[Sofa]". The "begin" feature has a value of "739". The "end" feature has a value of "749". The "day", "month", and "year" features all have a value of "null".

Feature	Value
▶ sofa	[Sofa]
begin	739
end	749
day	null
month	null
year	null

3.4. FeatureStructure View

The FeatureStructure View lists all feature structures of a specified type. The type is selected in the type combobox.

Its possible to create and delete feature structures of every type.



Chapter 4. Cas processor integration

An Analysis Engine can be run against either a whole corpus or just a few CAS files. To do this select a corpus or some CAS files and then choose in the context menu the correct Analysis Engine. The filename of the Analysis Engine must end with ".ann" otherwise it is not recognized as an Analysis Engine.

The CAS Consumer can be fed with the CAS files loaded from a corpus. To do this select a corpus and then select the consumer in the context menu. To add a CAS Consumer Descriptor paste a file into the processor folder. The filename must end with ".con"; otherwise it is not recognized as consumer.

