

Univerzita Karlova v Praze, Filozofická fakulta  
Ústav Českého národního korpusu

---



*INTERTEXT EDITOR*  
v1.1  
comprehensive guide

December 2013  
single-sided

Pavel Vondříčka  
Institute of the Czech National Corpus  
Charles University in Prague, Faculty of Arts  
<http://wanthalf.saga.cz/intertext>

---

This software and documentation resulted from the implementation of the project *Czech National Corpus* (LM2011023) funded by the *Ministry of Education, Youth and Sports* within the framework of *Large Research, Development and Innovation Infrastructures*.

# Contents

<b>I</b>	<b>Before you start...</b>	<b>1</b>
<b>1</b>	<b>What is <i>INTERTEXT</i> EDITOR</b>	<b>2</b>
<b>2</b>	<b>Current features of <i>INTERTEXT</i></b>	<b>3</b>
<b>3</b>	<b>Basic terms, concepts and problems of text alignment</b>	<b>5</b>
3.1	Structured text and its basic text units . . . . .	5
3.2	Texts, their versions and their alignment . . . . .	5
3.3	Alignable elements and their parents . . . . .	6
3.4	Segments of aligned units . . . . .	6
<b>4</b>	<b>How does <i>INTERTEXT</i> editor work</b>	<b>9</b>
4.1	Why a local repository ...? . . . . .	9
4.2	Alignment consistency . . . . .	9
4.3	Synchronization with <i>INTERTEXT</i> SERVER . . . . .	10
<b>5</b>	<b>Installation of <i>INTERTEXT</i></b>	<b>12</b>
5.1	MacOS X . . . . .	12
5.2	Linux (64-bit) . . . . .	12
5.3	Windows . . . . .	12
<b>6</b>	<b>Installation and configuration of hunalign</b>	<b>13</b>
<b>7</b>	<b>Things you should probably <i>not</i> do with <i>INTERTEXT</i></b>	<b>14</b>
<b>II</b>	<b>How to do basic things in <i>INTERTEXT</i></b>	<b>16</b>
<b>8</b>	<b>Create or import an alignment</b>	<b>17</b>
8.1	Create new alignment from two text files . . . . .	17
8.2	Import an <i>INTERTEXT</i> compatible alignment from files . . . . .	18
8.3	Import an alignment from an <i>INTERTEXT</i> SERVER . . . . .	19
<b>9</b>	<b>The user interface: or what do you see</b>	<b>21</b>
<b>10</b>	<b>Correcting alignment and text contents</b>	<b>24</b>
10.1	Editing alignment . . . . .	24
10.2	Applying automatic aligner . . . . .	25
10.3	Editing text contents . . . . .	26

---

10.4 Changing text structure . . . . .	27
<b>III Working with <i>INTERTEXT</i> in detail</b>	<b>29</b>
<b>11 Managing alignments locally</b>	<b>30</b>
11.1 Creating new alignment from texts . . . . .	31
11.1.1 Selecting name for the text and its versions . . . . .	31
11.1.2 Selecting text file(s) . . . . .	32
11.1.3 Importing an XML document . . . . .	32
11.1.4 Importing plain text document . . . . .	33
11.1.5 Converting plain text into simple XML document . . . . .	33
11.1.6 Choosing numbering scheme . . . . .	34
11.1.7 Creating automatic alignment . . . . .	35
11.2 Importing alignments from files . . . . .	35
11.3 Exporting alignments into files . . . . .	36
11.4 Exporting texts and alignment in different formats . . . . .	36
11.5 Editing alignment properties . . . . .	37
<b>12 Managing remote alignments (on <i>INTERTEXT SERVER</i>)</b>	<b>39</b>
12.1 Remote alignments manager . . . . .	39
12.1.1 Changing remote alignment properties . . . . .	40
12.1.2 Synchronization with the remote server . . . . .	41
<b>13 Search and replace text contents</b>	<b>44</b>
<b>14 Configuring and extending <i>INTERTEXT</i></b>	<b>46</b>
14.1 Options . . . . .	46
14.2 Configuring <i>INTERTEXT EDITOR</i> interface . . . . .	46
14.3 Configuring automatic aligner . . . . .	49
14.4 Configuring sentence splitter . . . . .	50
14.5 Configuring <i>INTERTEXT SERVER</i> connection . . . . .	52
14.6 Configuring text export formats . . . . .	52
<b>IV Appendix</b>	<b>55</b>
<b>A Regular expressions and special characters</b>	<b>56</b>
<b>B TEI XML alignment file structure</b>	<b>57</b>

## **Part I**

### **Before you start...**

# Chapter 1

## What is *INTERTEXT EDITOR*

*INTERTEXT* is an editor for aligned parallel texts, mainly aimed at creation of translational parallel corpora, potentially also usable with translational (CAT) or educational (CAE) software. It has been developed for the project *InterCorp*<sup>1</sup> to edit and manage alignments of multiple parallel language versions (i.e. translations) of texts at the level of sentences, but it is designed with flexibility in mind and supports custom XML documents and Unicode/UTF-8 encoding.

Please, note that *INTERTEXT* is a post-alignment editor and does *not* contain its own automatic aligner! However, it can utilize with existing automatic aligners like *Hunalign* or *TCA2*.

There are now two different tools available under this name:

1. *INTERTEXT SERVER* is a server based software package for central management and editing of aligned texts. It uses a web-based interface to manage the texts, their different language versions and their mutual alignments. An editor is available to manually correct both the alignment and the contents of the texts. It can also be controlled through scripts for batch manipulation of texts. The software is written in PHP and uses MySQL database as back-end.
2. *INTERTEXT EDITOR* is a desktop application, which can be simply downloaded and run locally on a private computer, without the need for an internet connection. However, it can also be used as a remote editor for alignments managed by an *INTERTEXT SERVER*, usable without a permanent connection (off-line editing). It offers richer and more comfortable editing capabilities and it also offers tools for easy creation of personal collections of aligned texts from raw texts without deeper knowledge of programming or text processing.

This guide only covers the latter *INTERTEXT EDITOR* desktop application, not the *INTERTEXT SERVER* package. Both applications can be obtained from <http://wanthalf.saga.cz/intertext>.

---

<sup>1</sup>Project of parallel corpora between more than 20 languages, managed by Institute of the Czech National Corpus, Charles University in Prague. <http://www.korpus.cz/intercorp/?lang=en>

## Chapter 2

# Current features of *INTERTEXT*

General features of *INTERTEXT*:

- can manage any number of texts and any number of their versions (translations)
- can work with very large texts at once (novels of several hundred pages, tens thousands of sentences)
- can work with (nearly) any valid XML document
- supports Unicode (UTF-8) by default
- allows for arbitrary and independent alignments between any pair of versions of the same text
- uses single-level alignment; every text version can define its own XML elements containing text to be aligned
- cooperates with *hunalign*<sup>1</sup> and *TCA2*<sup>2</sup> automatic aligners
- import and export of alignments in TEI XML format (stand-off alignment based on element IDs)
- free editing of the alignment
- free editing of the text contents
- split or merge existing text units (sentences or paragraphs), incl. automatic renumbering of their IDs
- synchronisation of multiple alignments of one text version (structural changes done to the text while editing one alignment should not break the others)
- manage status of each aligned segment (automatically aligned or manually verified)
- set bookmarks for later revision
- search in text (incl. regular expressions)

Additional features of *INTERTEXT EDITOR*:

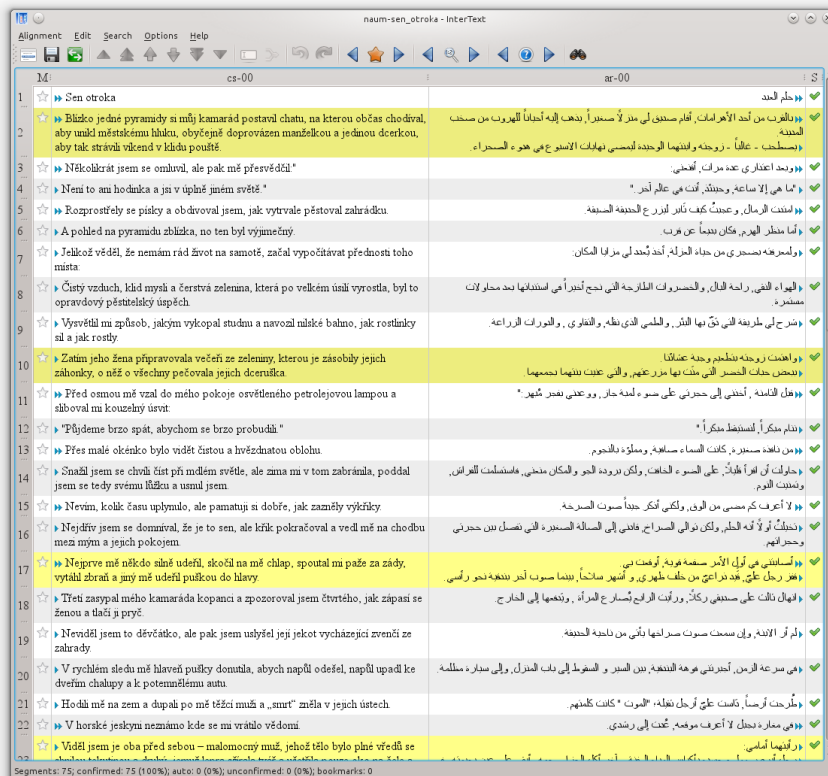
- creating new alignments from plain text files
- import of new-line aligned text files
- integrated sentence-splitter (fully configurable, based on regular expressions)
- (re)align any selected part of the alignment with an automatic aligner at any time
- automatic detection of basic numbering schemes of text unit IDs
- undo/redo capability

---

<sup>1</sup><http://mokk.bme.hu/resources/hunalign/>

<sup>2</sup><http://gandalf.aksis.uib.no/tca2/>

- full search and replace capability (including regular expressions with backreferences, “find all” (highlighting), “replace all”, search for element IDs, bookmarks, empty segments and non-trivial segments)
- user-configurable interface and integrated text-processing tools
- fully configurable export of text contents (pre-defined profiles for new-line aligned texts, ParaConc, TMX)
- runs on Linux, MacOS X, MS Windows





## Chapter 3

# Basic terms, concepts and problems of text alignment

### 3.1 Structured text and its basic text units

A text is usually not just an unstructured linear string of characters, but it consists of several levels of text units of different size, which build a hierarchical structure. The most basic two are paragraphs and sentences. Higher level structures may exist as well (chapters, parts, volumes, collections, etc.) but they are not so crucial for the process of alignment.

In the phase of manual verification of a pair of aligned texts, the alignment is mostly relevant on one level only<sup>1</sup> and the most practical is the level of sentences.<sup>2</sup>

Therefore, the term *text unit* will be used as a more general term in this guide, while the term *sentence* as its typical example. So, even though both terms have the same denotation in the context of this guide, the difference may still be relevant when speaking about a sentence (or text unit) wrongly split into two text units (not really sentences – that is what is probably wrong with the split).

In XML files, the *text units* are usually delimited by XML elements. Therefore the term *elements* also mostly refers to text units in this guide, however it is understood more technically as a component of the data structure.

Anyway, the level of text units (or elements) to be aligned can include units of different type, e.g. sentences and verses in a book containing a combination of prose and poetry. The only condition is that those structural elements must be exclusive and cannot include each other (in a hierarchical relationship) nor overlap.

### 3.2 Texts, their versions and their alignment

The term *text* is sometimes too general: not just any two texts can be aligned, it must be two *versions* of one and the same text (or document) which are in some way equivalent or corresponding with each other. For parallel translational corpora this means *language*

---

<sup>1</sup>Higher levels (e.g. document) are usually already manually “aligned” by the sole fact that we compare two corresponding versions (e.g. in the form of two text files) of one and the same text.

<sup>2</sup>Considering a written prosaic text. Alignment on lower levels, e.g. phrases or words, is too complex to be manually edited on the scale of very large texts and processed by current corpus engines.

*versions* or *translations*.<sup>3</sup> However, generally the versions to be aligned can also be two different versions (e.g. two different translations) in one and the same language.

Therefore, in the context of this guide (and *INTERTEXT* generally), the term *version* refers generally to one of the multiple parallel versions of one text (or document), which can be aligned with another version of the same text.

### 3.3 Alignable elements and their parents

In the context of an XML document, all text units which should be aligned are called *alignable elements*. XML documents may also contain other types of text elements (metadata), information, which is not part of the text contents and the alignment, and several levels of structural markup irrelevant for the alignment.

The ID attributes of the alignable text units are important for *INTERTEXT*, because they are used to identify the text units grouped into segments in a separate (stand-off) alignment file, and they must be changed (renumbered) whenever a structural change is made to the text and the number of units changes – i.e. when two units are merged or one unit is split into two.

If the IDs are numbered in two levels (e.g. when the ID of a sentence includes first the number of the paragraph and then the number of the sentence within this paragraph), *INTERTEXT* must also take care of the parent elements (i.e. paragraphs, or generally any higher level text units, a kind of “containers”) of the alignable elements (see chapter 11.1.6 for more details).<sup>4</sup> *INTERTEXT* is also capable of merging or splitting the parent elements.

If you only start with plain text, there are no metadata and all the text is considered alignable and is split into alignable text units, which will be automatically numbered by *INTERTEXT*. The number will become their ID.

### 3.4 Segments of aligned units

One *segment* is a cluster of alignable text units from two aligned texts: a pair of groups of units from the two text versions that fully correspond to each other. It is supposed that a segment is ideally a complete semantic and pragmatic unit: everything expressed by the units in the first version is either expressed by the units from the second version, or it has been “lost in the translation”, or the other way around.

The most common and most trivial type of segments contains only one text unit on each side: one sentence is usually translated by one sentence. Such segments are also marked as 1:1-segments and may thus also be called *trivial* segments. However, no alignment would be necessary if all the texts corresponded to each other so directly. Besides of the question “what is one sentence”, which cannot be treated here, quite often one sentence in one language corresponds to two or more sentences in another language. Then one segment may contain one sentence from one text version, but two or more sentences from the other version. Such *non-trivial* segments (1:2, 1:3, etc.)

<sup>3</sup>But the term “translations” excludes the original, which is not what we desire either. The original is also one of the parallel versions. Actually the original may not be present at all: translations may be aligned with each other directly.

<sup>4</sup>The terms *parent element* or *container* and the term *paragraph* will be used here in the same way as the mentioned basic terms *text unit* (or *text element*) and *sentence*. For a *verse* as a basic text unit, its parent element may be a *stanza* and not a *paragraph*.

require most attention in the process of automatic alignment and especially its manual verification.

The following example illustrates the most common situations:

<b>[en-s1]</b> Then there was a great scraping of chairs and the next moment, Harry found himself shaking hands with everyone in the Leaky Cauldron.	<b>[it-s1]</b> Ci fu un grande tramestio di sedie, e subito dopo Harry si trovò a stringere la mano di tutti i presenti.
<b>[en-s2]</b> "Doris Crockford, Mr. Potter, can't believe I'm meeting you at last."	<b>[it-s2]</b> 'Sono Doris Crockford, Mr Potter. <b>[it-s3]</b> Non riesco a crederci! <b>[it-s4]</b> Finalmente la conosco!'
<b>[en-s3]</b> "So proud, Mr. Potter, I'm just so proud."	<b>[it-s5]</b> 'Sono così orgoglioso, Mr Potter, veramente orgoglioso'.
<b>[en-s4]</b> "Always wanted to shake your hand – I'm all of a flutter."	<b>[it-s6]</b> 'Ho sempre desiderato stringerle la mano... Sono così agitato!'

In the table above, each row corresponds to one segment and each cell to a cluster of sentences from one text version. Most of the segments include only single sentences in the trivial ratio 1:1. Only the second row shows a segment where a single English sentence has three corresponding sentences in the Italian translation. The ratio is thus 1:3.<sup>5</sup>

Sometimes, translations may omit a whole sentence (or even several) completely. In such cases, segments with a ratio like 1:0 are convenient.

More complex segments are those where each side includes more than one sentence. For example, when complex sentences in one language continuously overlap in meaning with differently segmented complex sentences in the other language, more complicated clusters may arise, such as 2:3, 3:5, etc. The ratio may as well be equal (2:2, 3:3, etc.) – for example if one text uses one long sentence and one short sentence to express what the other language expresses by one short and one long sentence subsequently.

The most tricky situations (and fortunately the rarest ones) are those where one sentence seemingly corresponds to another one in the second language, but maybe just one detail (e.g. a single word's meaning) is seemingly missing. A closer look may reveal, that the word or its meaning is projected into the following sentence in the other language, so that the actual segment should contain both pairs of sentences (2:2) and not stay as two seemingly independent trivial 1:1 segments. This is a case which is extremely difficult for automatic aligners and which is easily overlooked even in the process of quick manual verification, unless the editor compares both texts really carefully. An extreme case of such configuration may appear where the meaning is projected beyond the lexical level: e.g. when a modal adverb in one sentence is actually "translated" into a modified grammatical category of the main verb in another sentence. From both a translational and linguistic point of view, such sentences should all become members of one single segment, even though the sentences would otherwise seemingly easily correspond to each other in the trivial 1:1 ratio. Otherwise the user of the final

<sup>5</sup>The example also illustrates the problem of the definition of a "sentence": Some of the more complex sentences in both languages could actually be split into smaller independent sentences according to different more or less sensible linguistic arguments, especially in the last segment. What delimits a sentence is a difficult question even for European languages. If it is a period, exclamation and question mark, why not a dash or an ellipsis sign as well, especially when followed by a capital letter.

corpus would be misled by the impression that the translator just completely omitted the word and its meaning.

## Chapter 4

# How does *INTERTEXT* editor work

### 4.1 Why a local repository ...?

*INTERTEXT EDITOR* stores all its files (documents and alignments) in a *local repository*, its own subdirectory created somewhere<sup>1</sup> in the user's directory. It takes care of consistency across several alignments of one text version as long as all the alignments are in the repository. Therefore it does not operate directly on just any text file you point at. Documents and alignments must be first imported into the repository from local files, from a remote *INTERTEXT SERVER*, or they must be created from some plain-text or XML document files.

When two language versions of the same text are pre-aligned, the list of their alignable elements (usually sentences) is displayed in the form of a table, such as shown above. Each column represents one text version and each row represents one segment of aligned elements and the sentences can be moved around as necessary. Their order cannot be changed, however: even though cross-align would be no technical problem for *INTERTEXT*, it is still a complication not worth implementation in most other current corpus tools – especially corpus search engines, where it could significantly degrade performance.

### 4.2 Alignment consistency

If there are several alignments of one single text version and you try to merge together two text units, which are aligned to two different sentences in some other language version (i.e. they are part of two different segments in some other alignment), *INTERTEXT* will report a conflict and you are forced to verify the problem and fix it manually before proceeding. In such situation it is obvious, that one of the two decisions you have made is wrong: either the decision to merge the two text units into one, or the decision to align them to two separate sentences in another alignment. When trying to merge two text units into one in a text version synchronized with an *INTERTEXT SERVER*, the server

---

<sup>1</sup>In Linux, the path is typically `~/.local/share/data/InterText/`; in MacOS X it can be found in `~/Library/ApplicationSupport/InterText/`; in MS Windows it is `C:\DocumentsandSettings\  
<username>\LocalSettings\ApplicationData\InterText\`

is checked for possible conflicts with other alignments as well, if there is currently an internet connection available. If there is a conflict, the operation will be denied as well. If there is no connection to the server, you will be warned that the acceptability of the change by the server cannot be verified and that the change may be denied later by the server.

### 4.3 Synchronization with *INTERTEXT SERVER*

It is difficult enough to keep consistency across a single repository of texts and alignments, especially when accessed by several users. When texts are distributed across several isolated private computers and often even out of connection with the server, some limitations must arise to be able to keep things under control at all.

The first limitation is the need to use the official synchronization process either to download an alignment (with the appropriate text versions) from an *INTERTEXT SERVER* to the local repository of an *INTERTEXT EDITOR*, or to upload a local alignment by the same way to the server.<sup>2</sup> It is not possible to synchronize text and alignments just by manually copying or otherwise transferring the files between *INTERTEXT SERVER* and *INTERTEXT EDITOR*, nor by simply stating that some alignment existing on both places is identical and should be synchronized. In such cases the alignment and texts must be simply deleted on one side and synchronized from the other side, because *INTERTEXT* would have otherwise no idea how to solve possible conflicts and differences.

The second limitation concerns the fact, that only the text contents and text structure can be synchronized in both directions. *INTERTEXT* cannot synchronize changes to the alignment itself made on both sides, nor would it actually have much meaning. The current state of alignment of two texts is always just uploaded from *INTERTEXT EDITOR* to *INTERTEXT SERVER*, where it replaces the previous one.

That means that after the first step of synchronization (either download from a server or upload to the server), the alignment is locked on the server for further modifications, and it stays locked until explicitly released by the user of the *INTERTEXT EDITOR*.<sup>3</sup> The text versions taking part in the alignment can still be modified both on the server<sup>4</sup> and in *INTERTEXT EDITOR*, and the system will synchronize the changes – it will warn the user of the *INTERTEXT EDITOR* that changes have been made to the text on the server, and the local user must either accept those changes locally as well, or refuse them (which reverts them, by effect). After dealing with any changes on the server, the local user is finally allowed to synchronize again, i.e. to submit his own changes back to the server.

When the local user finishes his work on some alignment, he can explicitly release the alignment, which opens it for changes on the server again. However, since then synchronization is not possible any more, it is meaningful to delete the alignment from the local *INTERTEXT EDITOR* repository. To continue with the synchronization, the alignment must be completely downloaded from the server again, to obtain it with all changes that may have been done to it on the server in between.<sup>5</sup>

That means that it is not possible to freely work on one and the same alignment both on the server and in *INTERTEXT EDITOR*, or to work on it in several different installations

<sup>2</sup>If allowed in the server configuration, which is *not* the default.

<sup>3</sup>Administrators may still change the alignment, but all the changes would be completely ignored and replaced by the newly uploaded alignment from the *INTERTEXT EDITOR* in the nearest process of synchronization.

<sup>4</sup>For example if it takes part in another alignment with some other text version, which is *not* going to be locked.

<sup>5</sup>Another possibility is to delete it on the server and upload again the version available in *INTERTEXT EDITOR*, if permitted by the server configuration.

of *INTERTEXT EDITOR* at the same time. The work must at every moment be done on one particular place only and all participating sides must be aware of the fact who has currently the full control of the alignment. The current place of work can be changed, but only via the central server. The work must be explicitly released on one place to be continued on another one. It cannot be just “open to any change” at several places at once.

Read more on management of remote alignments and the process of synchronization in chapter 12.

## Chapter 5

# Installation of *INTERTEXT*

Installation of *INTERTEXT EDITOR* is very easy. However, in most cases you will probably want to install an automatic aligner as well. *INTERTEXT EDITOR* can directly integrate only the *hunalign* automatic aligner, but it can also import stand-off alignments created by TCA2, which uses (as one of its outputs) the same TEI XML format.

### 5.1 MacOS X

1. Download and install the newest Qt 4.8 framework libraries from the *Qt* project site <http://qt-project.org/downloads>. (Do not use Qt version 5 or above!)
2. Download the latest *INTERTEXT EDITOR .dmg* package from <http://wanthalf.saga.cz/intertext/>, open it and copy the *INTERTEXT* application into your Application folder (or wherever else you want to have it).
3. (Optional) Download, compile and install *hunalign* – see below.

### 5.2 Linux (64-bit)

1. Install Qt 4.8 libraries with your package manager.
2. Download the latest *INTERTEXT EDITOR* binary for Linux from <http://wanthalf.saga.cz/intertext/>, and copy it into your favourite location.
3. (Optional) Download, compile and install *hunalign* – see below.

### 5.3 Windows

1. Download the latest *INTERTEXT EDITOR .exe* binary for Windows from <http://wanthalf.saga.cz/intertext/>, and copy it into your favourite location.
2. (Optional) Download and install *hunalign* – see below.



## Chapter 6

# Installation and configuration of hunalign

1. Visit <http://mokk.bme.hu/resources/hunalign/> to download the current version of *hunalign*. For MS Windows, you can download pre-compiled binaries, on other systems you have to compile them according to the instructions provided on the homepage.
2. Copy the resulting *hunalign* (or *hunalign.exe*) binary to your favourite location.
3. Copy the file “null.dic” from the directory “data” in your hunalign package into the same location as the binary and rename it to “empty.dic” (or alternatively just create an empty file with this name in the destination).
4. Start *INTERTEXT EDITOR*, open the settings (menu **OPTIONS > SETTINGS**) and switch to the **ALIGNER** tab. If the profile for the aligner “hunalign” is not selected from the list, select it. For the “Execute” parameter, select the *hunalign(.exe)* executable from the location you have copied it to. You should be done.

See also section 14.3 for details on aligner configuration.

## Chapter 7

# Things you should probably *not* do with *INTERTEXT*

Even though *INTERTEXT* is generally written with robustness in mind, the primary goal is its flexibility and configurability, which relies on the user, and therefore there are currently still some dangers you should be well aware of.

- **Running multiple instances of *INTERTEXT*** at the same computer (as one and the same user) may be risky. You should definitely not edit two related alignments (i.e. alignments of different versions of the same text) at once. All alignments of a text are interdependent and they are loaded in the background, when you open an alignment, so that they may be modified when some structural change is made to the text. Other problems may arise from synchronization with a remote *INTERTEXT SERVER*. If you open several instances of *INTERTEXT EDITOR* and make changes in both, one may overwrite and damage the work done in the other one. To prevent starting several instances of *INTERTEXT EDITOR* by accident, it creates a temporary lock-file in its repository (since version 1.1) while it is running, so that you will get a warning if you start a new instance of *INTERTEXT* when another one is still running.
- **Automatically accepting all changes from a server** is an option of the dialog asking for confirmation of changes (updates) received from an *INTERTEXT SERVER* when synchronizing a text. It is highly improbable, but still possible in theory, that *INTERTEXT* could make a mistake in matching a received update of a block of text units with the local text units to be replaced. If you want to be absolutely sure that your text will not be damaged, please verify that each change will exactly replace the appropriate part of text and nothing more or less. This is especially important when receiving complex structural changes to the text, or when significant parts of the text units were removed or added. See chapter 12.1.2 for more details.
- **Importing texts with wrong settings** may result in unusable alignments that must be deleted. A typical example is importing a text segmented into sentences and sending the text through a sentence-splitter again. This may result into a text with duplicate sentence mark-up, which will fail to open. Take care of whether you are importing a plain text or an XML document, and whether the text has already been segmented into sentences or not, and how the sentences are marked or delimited.



### *Warning*

A common warning concerning all software also applies here: *INTERTEXT EDITOR* is a **new, but already relatively complex piece of software** made for very different situations and tasks, and it has *not* been developed by a truly professional team of programmers. It has not been heavily tested under all circumstances and bugs may appear as it is exposed to various situations, types of texts, complex editing and especially synchronization with *INTERTEXT SERVER*. Such bugs may damage your data without notice. Please, backup your data regularly and inform the author if you notice some undesired or suspicious behaviour.

## **Part II**

# **How to do basic things in *INTERTEXT***

## Chapter 8

# Create or import an alignment

When you run *INTERTEXT EDITOR* for the first time, you probably do not really see very much. First you need to create or obtain some alignment to work on. Please, skip to the appropriate section depending on how you want to start. You can:

- Create an alignment from two text files
- Import an *INTERTEXT* compatible alignment from files you have obtained somewhere else (e.g. from TCA2 aligner).
- Import an alignment from your project's *INTERTEXT SERVER* (if you already are running one).

### 8.1 Create new alignment from two text files

*INTERTEXT EDITOR* can import and process any plain text files or XML documents as source of text. If you have your texts in MS Word or Open/LibreOffice, you do not have plain text files nor any usable XML documents.<sup>1</sup> You need to export your texts into plain text files in the unicode UTF-8 encoding or in some usable HTML/XML format<sup>2</sup> (in case you want to keep some formatting features like emphasized text). However, this subject cannot be treated in detail here and you may need to consult you local IT guru, if you do not succeed with pure experiments.

Once you have two parallel text files or XML documents in UTF-8 encoding, you can create a new alignment, e.g. through the menu *ALIGNMENT > NEW* (or from the *Repository manager*). First, you will be asked to enter an arbitrary name (identifier) for the text (use e.g. the name of the book or story, do not worry about file names here!). Both texts will be treated under this common name, even though they are probably in different languages. You also have to enter the names for the two versions you are going to align. A good practice is to use the names or (even better) just the two-letter *codes* of the two languages as the names of the versions.

Next you have to select the file with the first text version. In the “filter” box, select whether you are going to open an XML file or just any plain text file and then select the file. (If you open XML, skip 3 paragraphs below.) Confirm, that the file is a plain

---

<sup>1</sup>You may have heard that those editors also store documents in XML nowadays, but unless you are a computer wizard, forget about this possibility.

<sup>2</sup>The HTML export in most word processors (unfortunately including the latest LibreOffice versions) usually produces a lot of rubbish mark-up in the text, which makes it nearly unusable for practical text-processing.

## 8.2. IMPORT AN *INTERTEXT* COMPATIBLE ALIGNMENT FROM FILES

text file (and not something else). In the next dialog you will have to say how your paragraphs are separated in the text file, whether it is already segmented into sentences, and confirm how the resulting XML document will look like. If you have no cue about XML, just keep the defaults. However, you need to know something about your source text. Usually, paragraphs are separated by a single Enter (a “line break”), which is the default setting. But sometimes the paragraphs may be separated by blank lines (or “empty lines” – i.e. by two Enters) – in that case you would select the appropriate option. You may also select the encoding of the file, if it is different from UTF-8: the contents will automatically be converted, so you should *not* change the “UTF-8” string in the XML header field.

If your text file is not already segmented into sentences (e.g. one line = one sentence) by some tool or manually, you probably want to switch to the integrated automatic sentence splitter, which will usually be better than nothing as a starting point. You do not need to change any other settings (unless you know why you should).

In the next dialog, you have to choose how your sentences will be numbered. The default is “two level numbering” which means, that each sentence will first bear the number of the paragraph it belongs to and then the number of sentence within that paragraph. “Single level numbering” means that sentences will just be numbered continuously through the whole document by a single number.

Now, you have to repeat the same steps for the other text version to be aligned to: select text file, its type and set the appropriate parameters.

If you have selected an XML document and not a plain text file, you will be greeted by another dialog window, where you have to enter whether the text is already segmented into the alignable text units (usually sentences) and what are the names of all the XML elements containing the alignable text units (all others will be considered metadata and ignored) or whether you want to apply the integrated sentence splitter to create the sentences – in the latter case you have to enter the names of elements (containers) containing the text to be split into sentences (which usually means paragraphs). Take care: segmenting a text which already has been segmented into sentences (or contains XML elements with the same name of another reason) may result into unusable alignment which will fail to open.

When both texts are set up, a fake 1:1 alignment of the texts is immediately created and opened in the window, and at the same moment you are automatically offered to run a real automatic aligner (see chapter 10.2). If you have some aligner configured, you can just select the right aligner and profile and run it. By default it will (re)align the texts from the beginning to the end.

For more details on importing text files see chapter 11.1.

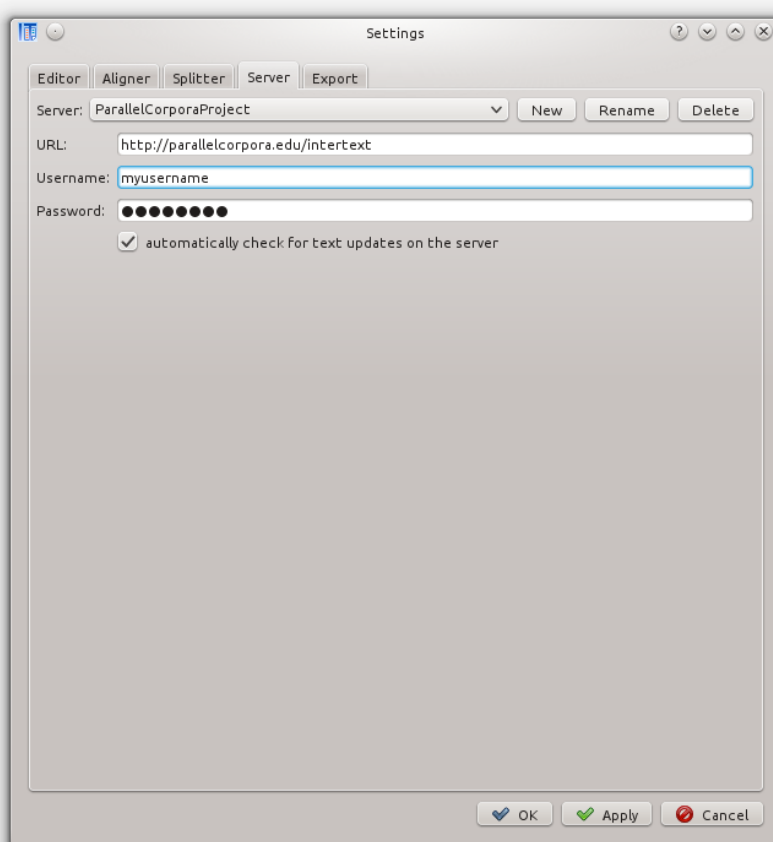
## 8.2 Import an *INTERTEXT* compatible alignment from files

You can import an TEI XML alignment created by *INTERTEXT* or a compatible tool (e.g. TCA2). You need to have the two XML documents with the two text versions and a separate TEI XML file with a stand-off alignment (see appendix B for details). Then you can for example select IMPORT from the ALIGNMENT menu and you will be asked to select the XML file with the stand-off alignment. This file should also contain the names of the two aligned XML documents (which should be accessible in the same directory as the alignment file), so that *INTERTEXT* can also automatically load the documents and

add them to its repository.

### 8.3 Import an alignment from an *INTERTEXT SERVER*

If you want to import an alignment from some *INTERTEXT SERVER*, you first need to configure the connection to your server. Open the menu **OPTIONS** and select **SETTINGS**. The settings dialog will pop-up. Select tab **SERVER**. Click the **New** button and enter some arbitrary name for your server into the dialog that pops-up now.<sup>3</sup>



The new server profile will be created and shown with empty values. You need to enter the URL you use to log-in to your *INTERTEXT SERVER* (just the basic directory, no particular PHP script – e.g. <http://ourserver.org/intertext>) and the login and password you use on the server. The option **AUTOMATICALLY CHECK FOR TEXT UPDATES ON THE SERVER** means that every time you open an alignment or document (which originates from this server), *INTERTEXT* will try to connect to the server and check whether

<sup>3</sup>You can use the name of your project, corpus or just name of the server or its location. The name will then appear in the sub-menu **ALIGNMENT > REMOTE SERVER**.

any changes have been done to the text in between, so that you always work with an up-to-date text. (See also chapter 14.5.)

After closing the settings dialog, you can find your server as a new item in the sub-menu **ALIGNMENT > REMOTE SERVER**. Selecting it will open a server dialog, which will show you all alignments accessible to you on the server. Using the **SYNC** button, you can download an alignment from the server into your local repository to work on it. The alignment will be locked for changes on the server until you release it with the **RELEASE** button. During that time, the texts can be any time synchronized between the server and your repository and their alignment can be uploaded from your *INTERTEXT* installation to the server (see chapter 4.3 for details on synchronization principles and limitations).

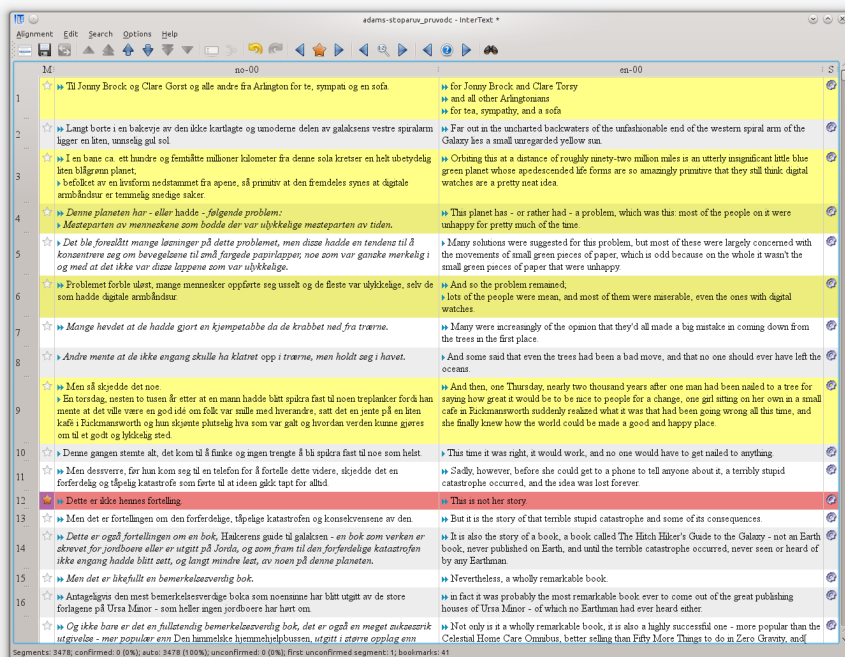
You can configure as many server connections as you want to. However, you cannot share alignments between servers. One alignment or document version can only be in synchronization with one server. It is not possible to have several documents or alignments with the same name in your repository. If your repository already contains an alignment with the same name as an alignment on the server, and it is not result of synchronization with this particular server, any synchronization of this alignment or its documents will be blocked.



## Chapter 9

# The user interface: or what do you see

If you already have some alignment in your local repository, you can open it from the *Repository manager* (open it e.g. from the menu **ALIGNMENTS**). The *repository manager* is a small dialog showing a list of all alignments in your local repository. You can select one and open it using the button **OPEN** or just by double-clicking the item. (For more details on the *repository manager* see chapter 11.)



The alignment will open in the main window in the form of a table similar to the scheme shown in the introductory chapter 3.4. The two main columns contain the two aligned texts side-by-side. Column headers show the names of the versions. Each row

in the table represents one segment.<sup>1</sup> It has a number (on the left side) showing the position (or segment number) in the alignment. Even rows are slightly darker than odd rows for easier orientation (depending on the settings). Each segment contains zero, one or several text units (usually sentences) – each starting with a blue triangle ▶ or a double blue triangle ▶▶. The double blue triangle marks units (e.g. sentences) which take first place within their parent container (e.g. paragraph) – so, it usually marks where new paragraphs start. Segments grouping a different amount of text units than 1:1 are by default highlighted with a yellow background (can be changed in the settings or just turned off in the menu OPTIONS).

Hovering the mouse cursor above any segment shows the list of IDs of all elements (units) in the segment in a tool-tip.

The leftmost narrow column contains bookmarks. A white (empty) star ☆ is a symbol for no bookmark. Clicking it toggles the bookmark ★ on and off again. By default, bookmarked segments are highlighted with a red background (can be changed in the settings or just turned off in the menu OPTIONS).

The rightmost narrow column shows the status of each segment: ✓ for manually verified (confirmed) segments; ⚙ for segments created by an automatic aligner; ⚠ for segments created in another way (e.g. by a fake 1:1 alignment), with unknown status or otherwise marked for further verification. Clicking the icon can toggle the status between “confirmed” and “unknown”.






Hovering the mouse cursor above a status icon shows its meaning in a tool-tip.

Clicking any of the text cells makes it selected for any further operation. It is possible to use cursor keys to move around in the table, as well as page-up and page-down keys (the exact behaviour of paging is configurable in the settings).













The status bar at the bottom of the window usually shows how many positions (segments) there are in the alignment altogether and how many of them have one of the possible states set (both as an absolute number and a percentage of the alignment size): number of confirmed (manually verified) segments, number of automatically aligned segments, number of segments with unknown status (see paragraph above), the number of the first non-confirmed segment in the text (i.e. with another status than “confirmed”, if there is any).

The status bar also shows context help when browsing the menu or state of a process running in background (e.g. synchronization).

At the top of the window, between the menu and the alignment table, there is a toolbar with buttons. The buttons trigger the most frequently used actions also accessible from the main menu or from the context menu (or from the *floating controls* - see below). Hovering the mouse cursor above a button shows its function in a tool-tip and a context help in the status bar at the bottom of the window. The buttons and their function is as follows:

-  opens the *repository manager*
-  saves the open alignment in its current state to the repository
-  starts synchronization with the server (if the currently open alignment is in synchronization with any *INTERTEXT SERVER*)
-  shifts the first text unit (sentence) from the currently selected segment up into the previous segment (row); rest of the text remains on the same place
-  moves the the whole text (in the same text version) one position (row) up – i.e. the selected segment’s text units are all shifted up to the previous segment (row) and all following segments are moved up by one position (row) up

<sup>1</sup> See chapter 3.4 on explanation of what is a segment.

-  moves both texts up by one position (row) – i.e. merges the whole current segment (row) with the previous one
-  moves both texts down by one position – i.e. inserts an empty segment before the current one (new row in the table)
-  moves the whole text (in the same text version) one position (row) down
-  pops the last text unit (sentence) from the currently selected segment down into the next segment (row); rest of the text remains on the same place
-  opens the currently selected text unit for editing; if a segment with several text units is selected, it turns into a menu and a particular single text unit can be chosen for editing<sup>2</sup> (the editing has to be triggered again on the selected unit)
-  triggers merge of the selected text unit with the preceding unit; the two text units have to be grouped in one single segment; if a whole segment with multiple text units is currently selected, a menu is opened and the particular text unit must be chosen (trigger merge again when it is selected); if the first text unit in a segment is selected, it will merge with the *following* text unit (since there is no preceding unit in that segment); if a segment is selected containing exactly two text units, they will be merged directly; a merge operation requires additional confirmation and for alignments synchronized with an *INTERTEXT SERVER*, the server is asked if the merge is not conflicting with some other alignment (if connection to the server is currently available)
-  undo the last operation (all text and alignment editing can be undone, including automatic re-alignment; synchronization or text updates received from a server cannot be undone)
-  redo the last operation undone
-  the arrows skip to the closest preceding or following bookmarked segment; clicking the bookmark symbol itself skips always to the *first* bookmark from the start of the alignment
-  the arrows skip to the closest preceding or following non-1:1 segment; clicking the symbol itself skips always to the *first* non-1:1 segment from the start of the alignment
-  the arrows skip to the closest preceding or following unconfirmed segment; clicking the symbol itself skips always to the *first* unconfirmed segment from the start of the alignment
-  opens the search bar (see chapter 13 for details)

When moving the mouse cursor above the table rows, the *floating controls* appear in the middle of the table above each segment by default. They contain the most important of the buttons needed to change the alignment. The controls can be switched to appear only on mouse-click or they can be turned off (hidden) completely in the *OPTIONS* menu.

<sup>2</sup>Or another action: merging, inserting or removing a paragraph break, etc.

## Chapter 10

# Correcting alignment and text contents

### 10.1 Editing alignment

By default, *INTERTEXT* expects you to read (and check) the alignment continuously from the beginning to the end. Once you make a correction to the alignment (not to the text contents!) at some position, it automatically marks all the preceding segments (positions) as confirmed – it is supposed you have already checked them and found no problems there. If you want to work differently and turn off this feature, you have to uncheck the `AUTO UPDATE STATUS` mode in the `OPTIONS` menu.

You can select the segment you want to change by a mouse click, or you can just move the cursor around with the cursor keys (arrows). At any position, you can either use the toolbar buttons, the `EDIT` menu, keyboard shortcuts (shown in the `EDIT` menu), context menu (triggered by right mouse-click) or the *floating controls* to trigger an action at the currently selected position (context menu and floating controls concern always the segment under the mouse cursor, not the selected one, of course).

- If you want to move the whole text column one position down, you can trigger the `MOVE TEXT DOWN` action in the menu, or using the ▼ button, or just by pressing `Enter`.
- If you want to move the whole text column one position up (i.e. to merge all the text units in the current segment with the preceding one and move the rest of the text upwards), trigger the `MOVE TEXT UP` action, or click the ▲ button, or just press the `Backspace` key
- If you just want to move a single sentence (the first one in the segment) to the previous segment, trigger the `SHIFT FIRST UP` action, or click the corresponding ▲ button, or press `Ctrl/cmd + Arrow-Up` keys at the same time.
- If you just want to move a single sentence (the last one in the segment, if there are several ones) to the following segment, trigger the `POP LAST DOWN` action, or use the corresponding ▼ button, or just press the `Ctrl/cmd + Arrow-Down` keys

In addition, there are two more shortcut actions available:

- If you need to insert a new empty segment (position or row in the table) on both sides (for example in order to split contents of one segment into two separate

segments), you can either “move down” each version one by one, or you can trigger the action INSERT SEGMENT (MOVE BOTH DOWN) in the menu, or by clicking the ▼ button, or just by pressing the Ctrl/cmd + Enter keys.

- If you need to merge the contents of two segments (rows) into one single segment, you can either move both versions up separately, or you can as well trigger the MERGE SEGMENTS (MOVE BOTH UP) action, click the ▲ button, or just press the Ctrl/cmd + Backspace keys



### Warning

Please, note the difference between “merging” text units (sentences) from different segments into one **segment**, and *merging* two text units into one single **unit** (as described in the following subsection)! The first operation only changes *grouping* of the sentences into segments within the **alignment**, not the sentence *boundaries* within the **text** itself.

If you need to move some text contents really far away (for example because of a missing chapter in a translation) and you want to avoid repeating the “move down / up” actions too many times, you can also use a special action called simply MOVE TEXT in the menu. Then you may enter the number of the position in the alignment you want to move the current segment’s contents to. Of course, the whole following text of the selected text version will be moved along.

You can toggle bookmarks either by clicking the bookmark symbol in the leftmost column of the table, or by triggering the action TOGGLE BOOKMARK in the EDIT menu, or just by pressing the M key.

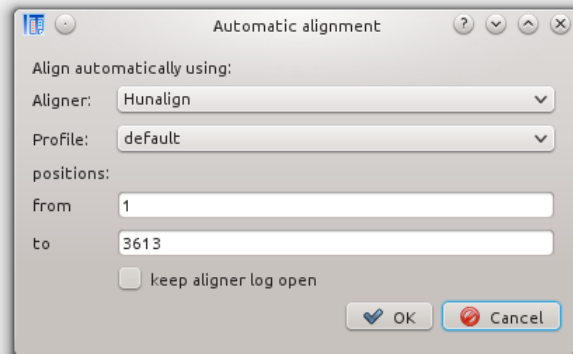
You can also manually toggle status of the single segments (positions) between “confirmed” and “unconfirmed” either by clicking the status symbol in the last column, or by triggering the action TOGGLE STATUS in the menu, or by pressing the S key. You cannot manually set the status to “automatically aligned”, though.

## 10.2 Applying automatic aligner

Automatic aligner is usually being run when a new alignment is created (see chapter 11.1). But sometimes you may need to run the automatic aligner again just on a part of your alignment in order to save yourself a huge amount of manual corrections. This occurs frequently if one text version is missing a whole chapter or more. In such case you only need to align the start and end of a piece of text and run the automatic aligner on that piece only. Or, if there is a large gap in the middle of one text version, you can only align the start of the block where the two texts correspond again (e.g. using the MOVE TEXT operation, see 10.1) and let the automatic aligner re-align the whole rest of the alignment.

The automatic aligner (if installed and properly configured – see chapters 6 and 14.3 for details on configuration) can be run from the menu EDIT > AUTO ALIGNER. A dialog window asks for more parameters for the alignment process.

You can select a configured automatic aligner and a particular profile (e.g. language specific configuration) to be used for the alignment. You can also determine the range of current positions (segments) to be re-aligned. By default the range is set from the currently selected position (or the first position if nothing is selected) to the last position in the alignment. After re-alignment, the position numbers will most probably change, of course.




The option `KEEP ALIGNER LOG OPEN` can be used to prevent the window showing the console output from the automatic aligner to close automatically when the process is finished. This feature can be used to debug the alignment process.

For details on configuring an automatic aligner and its profiles see chapter 14.3.

## 10.3 Editing text contents

*INTERTEXT* is not meant as a replacement for a text editor and definitely not an XML editor. Its main purpose is to align two texts by corresponding text units (typically sentences). However, during this process one frequently discovers typos or other mistakes in the source text and wants to quickly fix them.

Text contents can therefore be easily edited in *INTERTEXT* by triggering the `EDIT ELEMENT` action in the `EDIT` menu, in the context menu (right mouse-click), or by clicking the  symbol in the toolbar, or by pressing the `E` key when the desired text unit is selected, or just by double-clicking the segment. To select a single text unit within a segment with multiple units, you have to use *Edit* twice: first to open the segment contents menu to select the desired text unit (sentence), and then finally to open that unit for editing. If there is only one text unit in the current segment, it will be opened for editing instantly on the first *Edit* action.

When you finish editing the contents of a text unit, you can save the new contents by pressing the `F2` functional key, or you can anytime just escape the editing mode by pressing the `Esc` key to discard the changes made and return the contents to the previous state. If you just leave the editor, *INTERTEXT EDITOR* will ask you whether you want to save or discard the changed contents. By checking the option “Remember my choice and do not ask again.” you can set the default behaviour and avoid being asked again. This choice may always be changed in the `SETTINGS` later, though (see chapter 14.2).

Advanced search and replace functionality is also available and will be described in chapter 13.





### Warning

Please, remember, that editing is *not* meant to be used to move text contents between different text units. While moving single words, phrases or clauses between the units linguistically defined as “sentences” is an effective method to “cheat” and make all segments look like 1:1, this is not only terribly incorrect and destroying the whole linguistic mark-up, but it is actually also much more laborious thing to do. In addition, it will completely break things up if there are several alignments of one and the same text version: all the other alignments will be broken, because they rely on the fact, that element contents do not change.

## 10.4 Changing text structure

It is also possible to change the basic borders of linguistic units in the structure of the texts. Because *INTERTEXT* only cares about the elements containing alignable text units and their immediate parent elements (containers, i.e. usually the paragraphs), only the following four basic operations are available:

- Elements (text units) can be split into several new elements of the same type in the following way: open the defective element for editing, point the mouse cursor above the place where you want to split the unit, and from the context menu (right mouse-click) select **SPLIT**. Alternatively, you may also manually insert empty line(s) (two **Enters**) into the text on the place(s) where you want to split the element. After saving the changed contents (by pressing **F2**), the element will be split into as many new elements as desired, according to the empty lines in the new contents.
- Elements can be merged by using the **MERGE (BOTH) ELEMENTS** action from the **EDIT** menu, from the context menu, or by clicking the  symbol in the toolbar, or by pressing the **Alt + Backspace** keys simultaneously. Merge is only possible when both elements are currently grouped in one common segment. If there are more than two elements in the current segment, a submenu will be opened by triggering the action (same as for the *Edit* action) and you have to select the element to be merged with its *preceding* element and trigger the **MERGE** action one more time. Applying the action to the first element in a segment will have exactly the same effect as applying it to the second one: the first two elements will be merged.
- Parent element (e.g. paragraph) breaks can be changed too: it means you can split or merge paragraphs as well. By selecting a sentence which opens a new paragraph (i.e. it starts with the  symbol), you can use the action **DELETE PARAGRAPH BREAK** in the **EDIT** menu, in the context menu, or press **Ctrl/cmd + B** to delete the paragraph break and merge the current paragraph with the previous one. If the elements are part of containers (parent elements) of different type or on different levels in the XML structure, the merger will be denied.<sup>1</sup> To select an element within a segment with several elements, use the *Edit* action to open the submenu for selection of the single elements and then trigger this action on the selected element only. In case the action is triggered on the whole segment containing several elements, the action always affects only the first one.

<sup>1</sup>For example. when using different types of TEI containers for a prose “paragraph” containing sentences and a “stanza” containing verses, it will not be possible to merge a “stanza” with a “paragraph”.

- By selecting a sentence which is just somewhere in the middle of a paragraph (i.e. it starts with the ▶ symbol), you can insert a new paragraph break just before it by triggering the INSERT PARAGRAPH BREAK action in the EDIT menu, the context menu, or by pressing Ctrl/cmd + B. The current paragraph will then be split into two separate ones, the second one starting with the currently selected sentence. If the elements are enclosed in some other type of structural container (parent element) than a “paragraph” (e.g. “verses” in a “stanza”), the new container will be of the same type as the old one, of course.

Merging two elements can be a dangerous action for consistency of other alignments of the same text (see chapter 4.2 for explanation), and therefore it will be refused if there is some other alignment of the text, where the two elements are part of two different segments. If the text is synchronized with an *INTERTEXT SERVER* and an internet connection to the server is available, the server will also be asked for acceptability of such a merger.<sup>2</sup> If the merger is not approved by the server, the server may completely deny any further synchronization of the text.



#### Warning

Please, note that merging or splitting elements (sentences) is not “another way” of aligning the text contents. It is meant to correct a linguistically incorrect segmentation of the text structure. If a long sentence corresponds to two separate shorter sentences in the other text, you are probably expected to group it into one segment with the two target sentences, and not to split it according to the target text. If something is defined as one “sentence”, it does not suddenly become two sentences just because it is being translated by two sentences in another language. Of course, the definition of a “sentence” (or another linguistic unit used as alignable element) may be more or less arbitrary and flexible for different projects or purposes.

<sup>2</sup>It cannot be expected that all existing alignments of the two texts are also available (downloaded) in the local repository and fully synchronized.




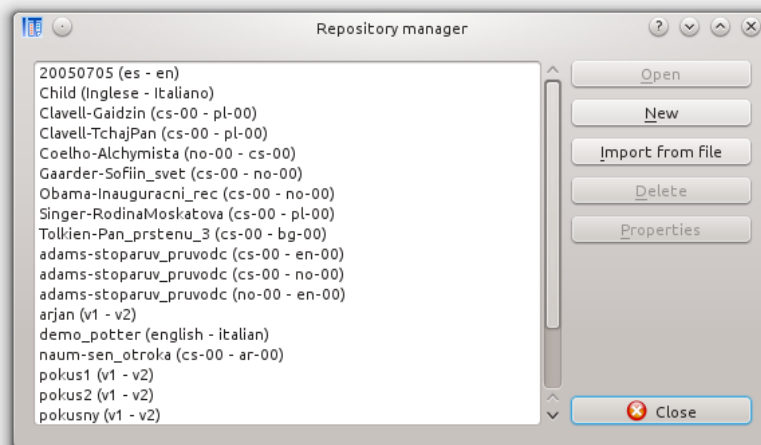
## **Part III**

# **Working with *INTERTEXT* in detail**

## Chapter 11

# Managing alignments locally

*INTERTEXT* keeps all local alignments in its own repository to be able to keep their consistency. All operations on the alignments can be accessed from the **ALIGNMENT** menu and/or from the *Repository manager*, which can be opened by the appropriate command in the **ALIGNMENT** menu, or by clicking the  symbol in the toolbar, or by pressing the **Ctrl/cmd + M** keys.



A simplified version of the manager with the single purpose to select an alignment to be opened for editing can be shown by the **OPEN FROM REPOSITORY** command in the menu.

*Repository manager* is a dialog window with a list of all alignments in the local repository and buttons for the most common operations: *open* an alignment for editing, create a *new* alignment, *import* alignment from files, *delete* an alignment and edit *properties* of an alignment. A few additional actions are available through the **ALIGNMENT** menu for the currently open alignment: *save* current state of editing to repository, *close* currently open alignment, *export* alignment to files, *export texts* and/or alignment in

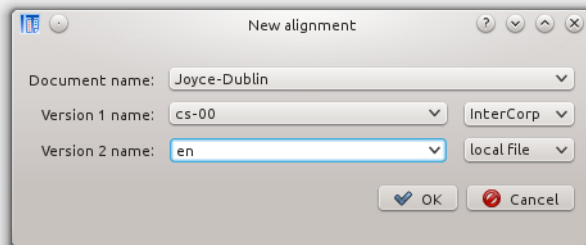
custom format, *synchronize* alignment with a server.

## 11.1 Creating new alignment from texts

Basic instructions for creating new alignments have been presented in chapter 8.1. The details on importing parameters will be described here.

### 11.1.1 Selecting name for the text and its versions

The first dialog window presented requires filling in the common name of the document and the two names for the versions to be aligned. The names can either be entered manually or chosen from a list of already existing names in the repository.



Select boxes on the right hand side of the version names can be used to switch the source from a local file import to a remote server, in order to select a text version to be imported from a pre-configured *INTERTEXT SERVER*. If a server is selected, the text name and version name editor fields change into select boxes as well and only text and version names available at the server can be selected. Arbitrary names cannot be chosen of obvious reasons: the text (document) and version names are considered general and thus unique identifiers across all projects you may access.

The source can be selected for both versions separately, so that texts from different sources can be aligned with each other:

- one version may be imported from a server and aligned with another version imported from a local text file
- both versions may be imported from the same server to create a new alignment between versions not yet aligned together on the server
- alignment can also be created from two text versions originating from two different *INTERTEXT* servers, but only if there are any documents with a common name (identifier) on the two different servers – the identity and uniqueness of text names as identifiers is automatically considered a necessary requirement

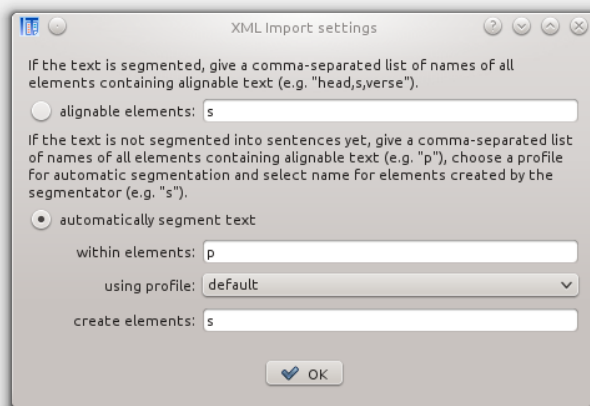
If you select a name of a text version already present in the repository, you will be forced to use that particular text or choose a different name to import a new text file. You cannot import a new text from either file nor a server using a name of some text version already existing in the repository.

### 11.1.2 Selecting text file(s)

If you did not select a remote server as a source, you will have to select a file to import for one (or both) text versions. You can either select an XML document or just any plain text file (switch filter to *All files*).<sup>1</sup>

### 11.1.3 Importing an XML document

If you select an XML document, you will be greeted by a new dialog asking for the relevant XML elements containing the text units to be aligned.



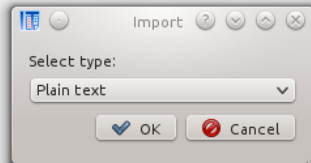
If the XML document is already segmented into alignable text units, you can directly enter a list (comma separated) with names of all elements containing alignable text units. All other elements will be ignored. Any sub-elements within these text elements will be treated as text mark-up. In case of simple prose text, it will be probably segmented into “s” elements containing single sentences. If you use rich TEI mark-up, you may wish too align several different text units (elements), e.g. *s*, *head*, *verse*, etc.

If your text is not segmented into the desired text units yet, you may use the integrated configurable splitter. By default it is configured to split text into sentences based on simple regular expressions (which may need adaptation for your particular language). But it can be configured to split text on arbitrary places, which can be defined by a set of regular expressions and a list of exceptions (see chapter 14.4 for details). To use the splitter, switch to the **AUTOMATICALLY SEGMENT TEXT**. You have to enter the list of names of all current XML elements containing blocks of text to be segmented (e.g. elements “p” containing paragraphs), you have to select a pre-configured profile of the splitter (several independent sets of splitting rules may be defined for different languages or purposes), and you have to choose a name for the newly created XML sub-elements delimiting the desired text units (“s” for sentences by default).

<sup>1</sup>Plain text can also be a XML fragment or text containing some XML/HTML-style mark-up. *INTERTEXT* can also import so-called *newline-aligned texts* – see chapter 11.1.4.

### 11.1.4 Importing plain text document

When selecting a plain text file, you will have to select whether it is really just some “plain text” file or a text from a newline-aligned pair of texts.



Newline aligned text pairs are plain text files, where each line corresponds to a single segment in the alignment and may thus include several sentences, just one sentence or none at all. It is produced by some automatic aligners, such as *hunalign* or *TCA2*. A pair of lines with the same number thus contains a segment (group) of text units corresponding to each other in the two corresponding text files.

### 11.1.5 Converting plain text into simple XML document

The next dialog will ask for details on how to convert the plain text file into an XML document. First of all, you should select the encoding of your text if it is different from UTF-8. The text will automatically be converted into UTF-8 and then processed in the following steps.

The text will be enclosed by an XML header and footer and its body will be parsed depending on the fact whether it is only segmented into text blocks (like paragraphs) or whether it is already segmented into the single alignable text units (e.g. sentences).

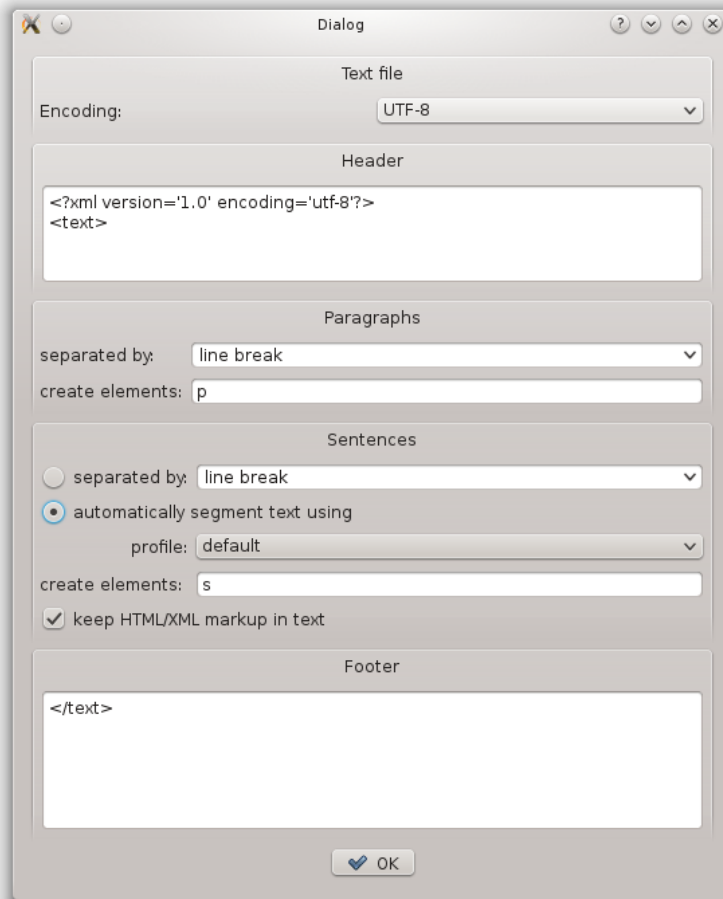
In this dialog, you can modify the XML header<sup>2</sup> and footer to be pre-pended and appended to the body of the text. You can also define the names for XML elements containing paragraphs and sentences, which will be created,<sup>3</sup> and select how these text units are delimited in the current source text file.

For current delimitation of paragraphs and sentences only two choices are available at the moment: either plain *line-breaks* or *empty lines* (i.e. two line-breaks following each other immediately). In common cases, you will only have paragraphs separated by line breaks and the sentences will not be segmented (separated) yet. In that case you may switch to the option `AUTOMATICALLY SEGMENT TEXT` and select a pre-configured profile for the integrated rule based sentence splitter (see chapter 14.4 for details). The splitter will delimit sentences for you and enclose them into elements with the name configured in the next field.

In case your text file contains some XML/HTML-style mark-up (emphasis by italics, bold type or underline, superscript or subscript, etc.), you should check the option `KEEP HTML/XML MARKUP IN TEXT`. Otherwise the elements from the source text will be converted into entities and handled as text characters and not as mark-up elements.

<sup>2</sup>Do not change the encoding in the XML header! Whatever you have selected as the encoding of the text file, it will be converted into UTF-8, so that the resulting XML will *always* be an UTF-8 encoded document.

<sup>3</sup>Of course these text units do not necessarily need to be paragraphs and sentences, but just any arbitrary alignable text units you want to distinguish and their corresponding parent blocks or containers.



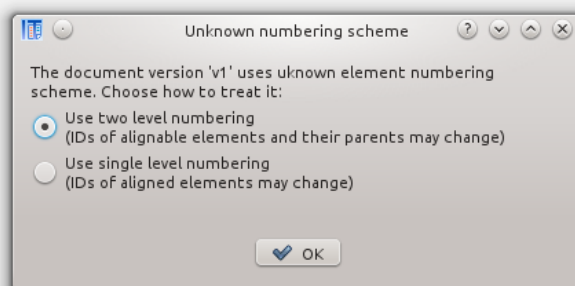
### 11.1.6 Choosing numbering scheme

Because *INTERTEXT* is able to change the structure of XML files (split or merge text elements / units), it must also be able to generate unique IDs of all the elements concerned.<sup>4</sup> When importing XML documents, *INTERTEXT* tries to determine any existing numbering scheme used in the document. If no numbering scheme is detected or its structure is too complex for *INTERTEXT* to understand,<sup>5</sup> you will be asked which scheme you want to use. *INTERTEXT* is able to create two types of generated IDs:

1. single level numbering, where only the alignable elements are numbered and the ID is thus a simple number from a single sequence of numbers (e.g. sentences 1, 2, 3, 4, etc.) showing the order of the unit in the whole text, independent of its place in the document's structure
2. two level numbering, where the parent elements contain a simple number from a single sequence of numbers (e.g. paragraph 1, 2, 3, 4, etc.) and the alignable

<sup>4</sup>See chapter 3.3

<sup>5</sup>See chapter 11.5 for closer description of acceptable numbering structures and how the scheme can be changed.



text-units (e.g. sentences) within contain both the number of the container (parent element / text block), a separator and a number showing the order of the unit within the current container (e.g. sentences in the paragraph 3 will be numbered as 3:1, 3:2, 3:3, etc.)

When importing new texts, you have to decide which numbering scheme will be used. If you choose the two level numbering, the default separator will be “:” and there will be no additional prefix.

### 11.1.7 Creating automatic alignment

When both parallel text versions are imported, a simple 1:1 pseudo-alignment is created in the window. Because such “alignment” is laborious to correct manually, it is more practical to align it with an ordinary automatic aligner first. This action is normally available in the **EDIT** menu, but at this moment it will be offered to you automatically. See chapter 10.2 on more details on running automatic aligners.

## 11.2 Importing alignments from files

Importing ready-made alignments from files is different from creating new alignments from text documents, because it expects you to have all three files – both XML document versions and their TEI XML stand-off alignment – already available in the correct format (e.g. as one of the formats generated by TCA2 or created by another *INTERTEXT* application).<sup>6</sup> You are only asked to select the file with the stand-off alignment and the text files are automatically imported according to their filenames presented in the alignment file. It is expected that they are available in the same directory as the alignment file.

See also the following chapter for more information.

<sup>6</sup>See appendix B for details on the alignment file format.

## 11.3 Exporting alignments into files

A pair of texts and their alignment can be exported in the form of three files, such as those which can be directly imported. The two text versions are exported as two independent XML files and their alignment is exported as a separate, third TEI XML alignment file.<sup>7</sup>

You will be presented with three different file saving dialogs offering you to save the files with the default filenames. The alignment expects you to keep the filenames. Otherwise you need to fix the alignment manually.

## 11.4 Exporting texts and alignment in different formats

The submenu `EXPORT TEXTS AS` offers different pre-configured formats to export the texts and/or their alignment to. By default, there are the following formats prepared:

- **NEWLINE ALIGNED (ONE FILE)** – exports both text versions into a single plain text file containing the alignment in the form of a tab-separated table: each line represents a single segment and contains: all text units from the first text version, a TAB character as separator, and all text units from the second text version belonging to the same segment. This format is a common output of hunalign as well.
- **NEWLINE ALIGNED (SEPARATE FILES)** – exports both text versions as two separate plain text files, where each line corresponds to a single segment and may contain one, several or no sentence, so that text units placed on lines with corresponding line numbers are members of the same segment (i.e. correspond to each other). This is the format which may also be imported as indicated in chapter 11.1.4.
- **PARACONC TEXT (UTF-8)** – exports both text versions as (seemingly) ordinary XML files, but with their contents partitioned by additional “seg” tags into the segments numbered sequentially by the position numbers like row numbers in *INTERTEXT*. These “seg” elements with corresponding numbers link the aligned contents of the two text files. However, ParaConc does not care about validity of XML structure and so is this export not a valid XML file either.<sup>8</sup>
- **TMX (STRIPPED MARKUP)** – exports a simple TMX file<sup>9</sup> stripped of any mark-up and all segments where one or the other text version is missing any contents (translation). This file can be imported by different CAT tools<sup>10</sup>, e.g. to be used in translation memories. You will be asked to enter valid values for language identifiers used in the resulting files (by default filled by the names of the exported versions).

All text export formats are limited by the fact, that they only can export raw text contents, including any custom text unit and container unit delimiters. The rest of the XML structure and any possible metadata will be ignored. Text content mark-up can be preserved, of course.

<sup>7</sup>Its structure is described in chapter B

<sup>8</sup>The “seg” elements completely ignore the rest of the XML structure and just overlap randomly with the structure of the contents.

<sup>9</sup>*Translation Memory eXchange* is an open XML standard for the exchange of translation memory data created by computer-aided translation and localization tools. (Wikipedia: [http://en.wikipedia.org/wiki/Translation\\_Memory\\_eXchange](http://en.wikipedia.org/wiki/Translation_Memory_eXchange))

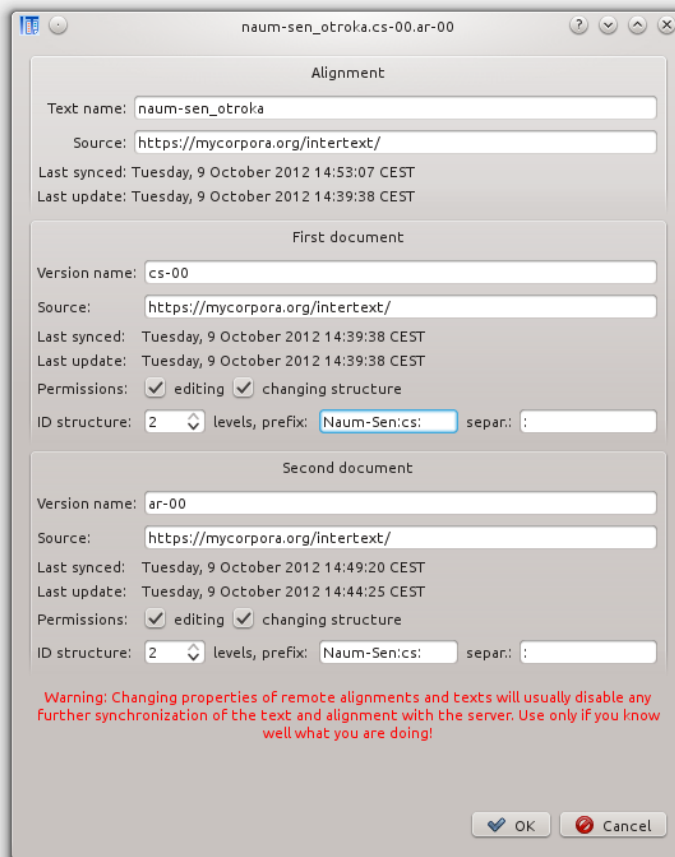
<sup>10</sup>Computer Aided Translation



The formats may be modified<sup>11</sup> or extended according to any needs (with the limitation mentioned above) and new formats may be defined, because they are just different individual configurations for a flexible text contents exporting method. See chapter 14.6 for details on configuration of text export.

## 11.5 Editing alignment properties

*INTERTEXT* keeps some metadata connected to each alignment. These are especially crucial for synchronization with a remote *INTERTEXT SERVER*. When a text is synchronized with a server, it is also not recommended to change these settings unless you know well what you are doing. Changing settings for a synced alignment or some of its text versions may result in irrecoverable inability to synchronize with the server.



The dialog shows properties connected to an alignment. The name of the text and the names of the versions may be changed here. The field `SOURCE` is crucial for text

<sup>11</sup>For example, you may desire to change the default ParaConc export encoding to some legacy 8-bit encoding, because ParaConc does not fully support Unicode.

versions or whole alignments synchronized with a remote server, because they contain the URL of the server. Otherwise the contents of this field is arbitrary and irrelevant for *INTERTEXT*.

If only some text version contains an URL in the source field, it is only this version which is being synchronized. If the whole alignment is synchronized, the URL will be filled as the *source* of the alignment as well. The URL must correspond to some server configuration in the settings. If you change URL of your *INTERTEXT SERVER*, you will not only have to change your *INTERTEXT EDITOR* settings, but also manually fix the URL of all synchronized alignments and text versions!

The alignment and each text version also contain information about the last successful synchronization and the last local change. Editing a text only affects the time of change of the particular text version. Changing structure of a text version affects both the text version and the time of last change of the alignment. A change in the alignment only affects time of last change of the alignment itself.

Permissions can also be set for editing and changing structure of each text version separately. However, if you override permissions of texts imported from a server, you will be able to change the text locally, but you will not be able to submit the changes to the server, because it will deny them. Synchronization will then be impossible unless the administrator or your supervisor allows you to make such changes on the server as well.

You can also change the parameters of the numbering scheme for each document. The “prefix” is a general string pre-pended to each ID. The separator is only relevant for two level IDs (see chapter 11.1.6 for more details). The combination of a general prefix and either a single level numbering or a two level numbering with a defined separator are the only numbering principles understandable for *INTERTEXT EDITOR*. These are also the numbering schemes it is able to automatically detect on import of external alignments.

## Chapter 12

# Managing remote alignments (on *INTERTEXT SERVER*)





*INTERTEXT EDITOR* may also be used as an off-line editor for alignments available on one or several remote *INTERTEXT* servers. Because it has additional features,<sup>1</sup> it may also be used for more complex operations not available by the *INTERTEXT SERVER*. However, it is also missing some features of the *INTERTEXT SERVER*.<sup>2</sup>

The principles and limitations of the synchronization between an *INTERTEXT EDITOR* and an *INTERTEXT SERVER* have already been extensively described in chapter 4.3 and it is important to understand them well before using this functionality. The configuration of a server connection and the basics of synchronization have also been described in chapter 8.3.

### 12.1 Remote alignments manager

The menu **ALIGNMENT** contains a submenu **REMOTE SERVER** presenting a list of all configured remote *INTERTEXT* servers. Choosing one of them opens a remote alignment manager for that particular server.

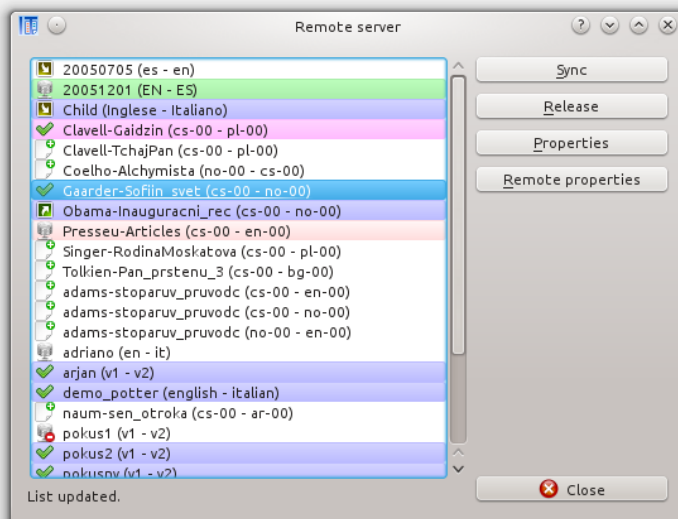
The dialog will download a list of all alignments accessible to the configured user on the remote *INTERTEXT SERVER*, mixed with a list of local alignments available in the repository. Different icons and colours indicate the current state of each alignment:

-  marks alignments only existing in the local repository
-  marks alignments only available at the remote server
-  marks alignments that cannot be synchronized because of a conflict (either the alignment or some of its text versions is available under the same name both on the server and in the local repository, but they are not marked for synchronization; this also applies to previously synchronized alignments which have been *released* but not removed locally)
-  marks synchronized alignments which need to be updated, because some of the text versions has been modified on the server

---

<sup>1</sup>Such as *search and replace* capability or ability to automatically re-align just an arbitrary part of an alignment.

<sup>2</sup>E.g. the history of changes.



- marks synchronized alignments containing local changes which have not yet been submitted to the server
- marks fully synchronized alignments (their state on the server is the same as in the local repository)

The background colour marks the remote state of the alignment:

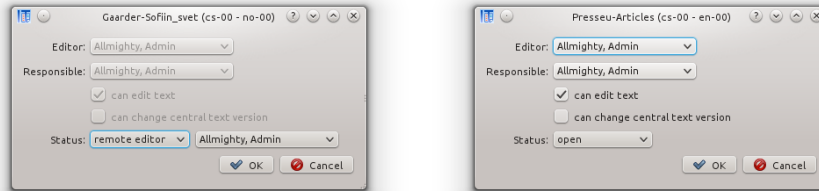
- *blue colour* marks remote alignments marked on the server as being edited remotely by an *INTERTEXT EDITOR* and therefore locked for changes in the web editor. All synchronized alignments should be marked by this colour.
- *red-violet colour* marks alignments marked as being edited by an *INTERTEXT EDITOR*, but another user
- *green colour* marks alignments marked as “finished” on the server
- *orange-red colour* marks alignments marked as “blocked”
- *white background* marks remote alignments marked as “open” (and also all local alignments)

Buttons on the right hand side may be used to: start *synchronization* (see chapter 12.1.2), *release* a synchronized alignment (see chapter 4.3), edit local *properties* of the alignment (see chapter 11.5) and edit *remote properties* (see below).

### 12.1.1 Changing remote alignment properties

Using the *remote properties* dialog, you can change the same server properties which you would be able to change through the web interface of the *INTERTEXT SERVER* (i.e. depending on your role and permissions).


You can set the editor, the responsible supervisor, set permissions to edit (change) text (of both text versions) and the additional permission to change structure in the



text version which is configured as “central” (or pivot) in your project (if any of the two versions is concerned at all). You can also change the status of the alignment on the server and possibly change the user assigned as “remote editor” if the alignment is blocked for remote editing in *INTERTEXT EDITOR*.

For more details read documentation for the *INTERTEXT SERVER*.

### 12.1.2 Synchronization with the remote server

Synchronization can be started for synchronized alignments any time when the computer is connected to the network and can access the configured *INTERTEXT SERVER*. It can either be run from the menu **ALIGNMENT > SYNCHRONIZE WITH SERVER**, using the  button on the toolbar or from the *remote alignments manager*.

In the *remote alignments manager* the **SYNC** button can also be used on remote alignments which are not synchronized yet to download them into the local repository and set their state on the server to “remote editor” with reservation for the current user.<sup>3</sup> It can also be used on local alignments to upload them to the server (if the server is configured to accept new uploads from the user).

The process of synchronization has the following three phases:

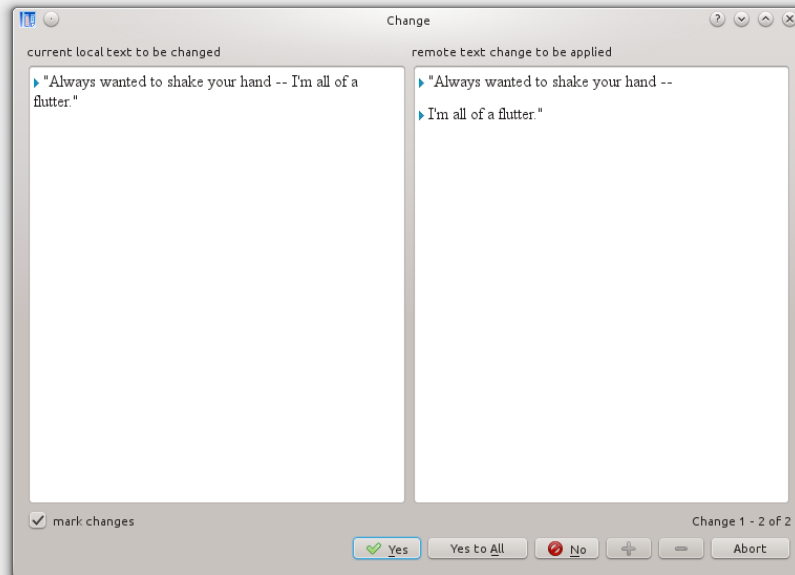
#### Download of changes from the server

If a text version has been modified on the server, you will be presented with a special dialog asking you to accept every single change from the server. Refusing some of the changes will keep your current local state of the concerned text unit(s) and in the next phase it will be submitted back to the server together with other local changes you may have made in the text, and in that way revert the change made on the server.

The dialog shows changes in blocks of connected text units, which may be part of a single change. The left pane shows the current local state of the text units concerned, and the right pane shows the current state of the same units on the server. Accepting the suggested modification will change the local text according to the state on the server (the left hand side contents will be replaced with the right hand side contents).

Below the panes there is a **MARK CHANGES** checkbox on the left side. When checked, all positions containing any accepted changes will also be marked by a bookmark for later revision of both the text and possible changes applicable to the alignment as consequence to changed text structure. On the right side, there is an indicator showing how many changes there are in the queue and which one is the current one (it may be a block of several partial changes).

<sup>3</sup>See chapter 8.3 as well.



The button **YES** accepts the change(s), the button **NO** denies the changes (and they will be reverted on the server in the next phase). The button **ABORT** aborts the whole synchronization process and the whole local alignment will revert to the exactly same state as before the start of the synchronization.

The role of the other buttons is described in the following warning:



### Warning

*INTERTEXT EDITOR* tries automatically to match the downloaded changes with the corresponding block of local text units to be changed. However, in case of particularly complex structural changes and some types of texts (e.g. with repetitive text sequences) it may fail to match the text units correctly, so that accepting the change would result in irrecoverable damage of the text. Even though such situation is very improbable under normal circumstances, it is strongly advised *not* to use the button **YES TO ALL** blindly to automatically accept all changes, but to manually check each (block of) change(s) and confirm them one by one. If the blocks do not match, the buttons **+** and **-** can be used to add or remove additional text units from the block of units which are going to be replaced by the received modified text, and in this way match the contents manually.

### Submission of local text changes

When both text versions have been confronted with all changes made on the server, *INTERTEXT EDITOR* starts submitting any local changes back to the server, again for each of the text versions if both are synchronized. The changes are submitted one by one in the same way as if the user did the changes manually through the editing interface of the *INTERTEXT SERVER*. Therefore they are also logged into the changelog of the concerned text units on the server. The status bar shows the progress of the process. If the server

denies some of the updates (e.g. because of some permission conflict or conflict with another alignment), the process will be interrupted and the only solution is to fix the particular conflict on the server and then start synchronization again.

It is not important how many times a text unit has been changed locally. Only its current (final) state is submitted. If you split a sentence, merge it again and then synchronize with the server, the change will only appear as a possible change of text unit contents, because the structure is the same again, but *INTERTEXT* cannot be sure the contents were not changed during the splitting operation.<sup>4</sup> *INTERTEXT* does not check whether the actual text contents have really changed or not. (These principles also apply to changes downloaded from the server in the previous phase.)

### Upload of the alignment

When both texts have been fully synchronized and *INTERTEXT* can be sure that their structure (and contents) is identical both on the server and in the local repository, the alignment itself is submitted to the server, where it fully replaces the current alignment.





There is no real synchronization or updating of single changes: the original alignment on the server is just deleted and replaced with the newly uploaded one. Therefore the alignment on the server must be locked for changes, because they would be lost during the next synchronization. And therefore the synchronization cannot be easily restored after it has been “released” once: any changes possibly made on the server after unlocking the alignment must be downloaded again as a complete download of the alignment. Tracking single changes in alignment is only an optional feature of the *INTERTEXT SERVER* and it is not implemented by *INTERTEXT EDITOR* at all.

---

<sup>4</sup>Remember that splitting is done through editing the contents!

## Chapter 13

# Search and replace text contents

The menu **SEARCH** and the toolbar offer access to the 9 basic searches for previous, first and next bookmark, non-1:1 segment and unconfirmed segment. Additional searches are provided by the *search and replace toolbar* which appears at the bottom of the window when the button  is clicked on the toolbar or when triggering the menu **SEARCH > FIND** (or pressing **Ctrl/cmd + F<sup>1</sup>**) or **REPLACE** (or pressing **Ctrl/cmd + R**). Unless the last option is used, the search bar only appears in its reduced form for searching only. The bar can be expanded to the full *search and replace bar* by clicking the symbol  on the right hand side, and again reduced by clicking the symbol . It can be closed by clicking the symbol  on the left.

The select box on the left side of the search text field offers the following types of searching:

- **Substring (a=A)** – case insensitive search for a substring
- **Substring (a<>A)** – case sensitive search for a substring
- **Regular exp. (a=A)** – case insensitive search for a regular expression<sup>2</sup>
- **Regular exp. (a<>A)** – case sensitive search for a regular expression
- **Element ID (reg.exp.)** – search for a particular alignable element by its ID (using regular expressions)
- **Empty segment** – search for a segments containing no text units (on one or both sides – see below)
- **Segment <> 1** – search for a segments containing a different amount of units than one – i.e. no unit or more than just one (on one or both sides – see below)

The select box on the right side of the text field allows you to search either in the *left* side text version only, the *right* side text version only or in *both* versions at the same time.

Found occurrences of strings are highlighted by orange colour, replaced occurrences by green colour.<sup>3</sup>

The other buttons have the following functions:

- **Next** – skips to the next (following) occurrence from the currently selected position (or from the beginning of the text, if nothing is selected); is equivalent to the menu item **SEARCH > FIND NEXT** (or pressing **F3<sup>4</sup>**)

---

<sup>1</sup>The shortcuts may differ in different operating systems. Please check the menu for your system's shortcut.

<sup>2</sup>See chapter A for details on supported regular expressions.

<sup>3</sup>These colours may be configured in the settings. See chapter 14.2.

<sup>4</sup>The shortcuts may differ in different operating systems. Please check the menu for your system's shortcut.



- **Previous** – skips to the preceding occurrence found; is equivalent to the menu item SEARCH > FIND PREVIOUS (or pressing Shift + F3)
- **Find all** – highlights all found occurrences of the search string; highlighting is removed after another search is started (search conditions are changed) or by closing the search bar
- **Find & replace** – replaces the currently found occurrence (if any) and skips to the next (following) one
- **Replace** – replaces the currently found occurrence (if any), but does not search any further
- **Replace all** – replaces all occurrences at once (they will all be highlighted and the number of successful replacements will be reported)

This function can also be used to automatically *find and split* wrongly segmented elements (since version 1.1) – see appendix A. No *find and merge* function is available, though. You are suggested to check the segmentation of your texts before importing them into *INTERTEXT*.



#### Warning

The *search and replace* function does not differentiate between the pure text contents and a possible markup within the contents of the alignable text elements (which is not visible in the *HTML view* mode!). It may thus replace your XML mark-up elements (their names) or their attributes and/or values (if there are any and by accident they match the conditions). Since version 1.1, *INTERTEXT EDITOR* is able to automatically fix broken XML tags, so that damage of text mark-up may proceed unnoticed by the user.

## Chapter 14

# Configuring and extending *INTERTEXT*

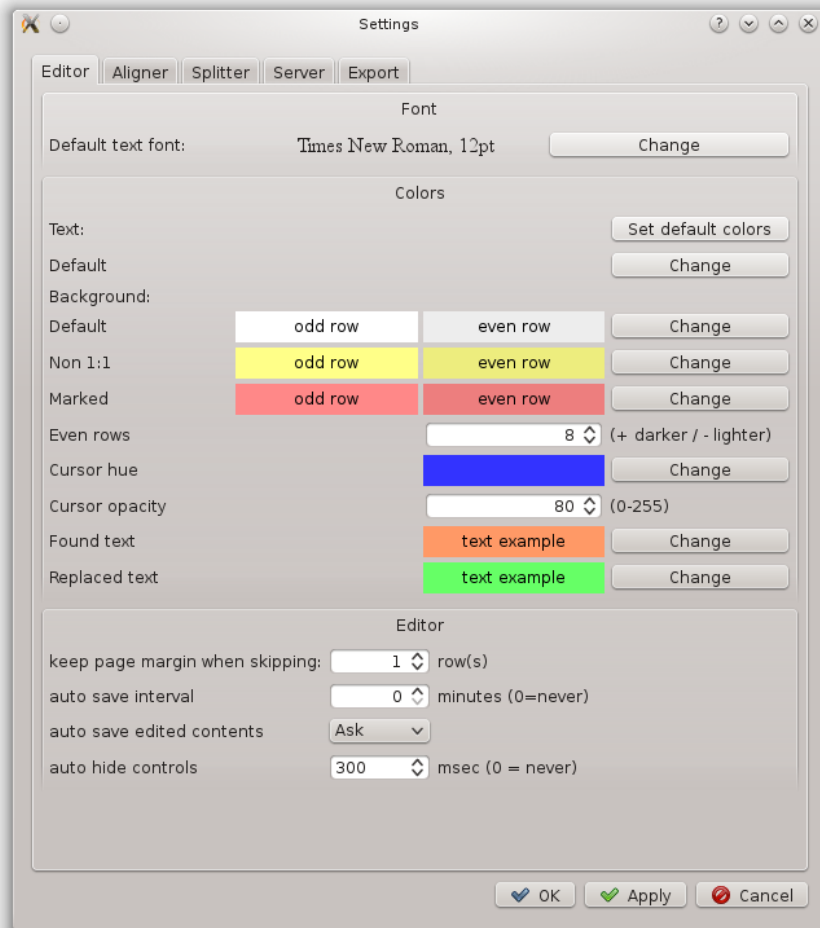
### 14.1 Options

Some basic options may be changed quickly through the menu **OPTIONS**. These are:

- **Auto update status** – automatically set alignment status to *confirmed* for the currently changed position and all preceding positions (see chapter 10.1)
- **HTML view** – toggles between the plain text view and HTML view of the text element contents; in HTML mode all XML/HTML elements are hidden and the markup elements conforming to HTML standards behave in the same way as in a HTML browser (e.g. `<b>` for bold font, `<i>` for italics, etc.); custom markup cannot be interpreted (rendered) by *INTERTEXT*; in plain-text mode everything is fully visible, including any HTML/XML tags
- **Highlight non-1:1** – toggles highlighting positions with another ratio of aligned text units than 1:1 (the colour can be configured in settings, see 14.2)
- **Highlight marked** – toggles highlighting (book)marked positions with a red background colour (the colour can be configured in settings, see 14.2); if not highlighted, the (book)marked positions are only indicated by the red star in the first column
- **Toolbar** – submenu where the size of the toolbar can be changed (or the toolbar can be completely *hidden*)
- **Controls** – submenu where the behaviour of the *floating controls* can be changed (whether they appear *on mouse move*, *on mouse click* or whether they should stay completely *hidden*) – see chapter 9
- **Settings** – opens the settings dialog for further configuration; see following chapters for details

### 14.2 Configuring *INTERTEXT EDITOR* interface

The **EDITOR** pane of the settings dialog allows you to customize the editor interface. You can select the default font used to render the text contents here. You may also modify the colours used in the alignment table.



The selected colour combinations are shown on samples in the middle of the dialog. The colours can be set for:

- *Default text colour*
- *Default background colour* – default if no other colours apply
- *Non 1:1* – background colour for segments aligning text units in a ratio other than 1:1
- *Marked* – background colour for (book)marked segments
- *Even rows* – modification of the shade of the background colour for even rows: 0 = no change, positive number = darker; negative number = lighter
- *Cursor hue* – the basic hue for the cursor in the table
- *Cursor opacity* – the opacity of the cursor colour (i.e. how much the *cursor hue* gets blended with the original background colour)
- *Found text* – background colour for found occurrences of searched strings (when using *search & replace*)
- *Replaced text* – background colour for replaced strings (when using *search &*

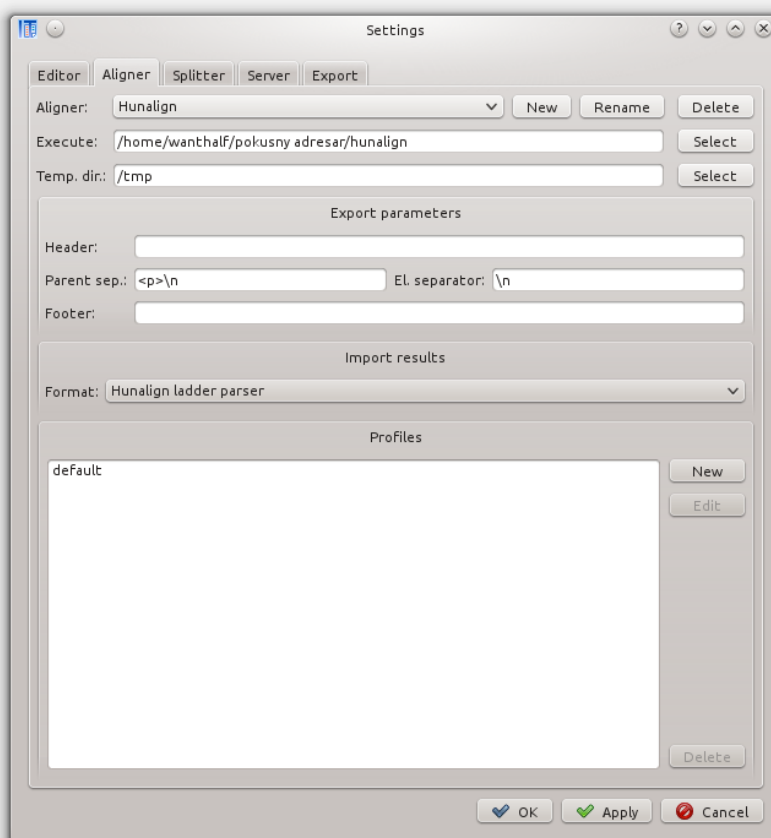
*replace*)

In addition, the behaviour may be configured for:

- *keep page margin when skipping* – sets the number of positions used as overlap when moving up and down the table using the PageUp and PageDown keys; when you page down or up, there will always be this number of positions shown from the previous page, so that you can be sure you do not overlook anything
- *auto save interval* – sets the interval in minutes when *INTERTEXT* automatically saves you current state of work to the repository (so that you won't lose your work e.g. in case of sudden power outage or software crash); setting the value to 0 turns the auto-save function off
- *auto save edited contents* – sets the behaviour when leaving an open text editor without pressing one of the keys Esc or F2: the changed contents can automatically be *saved*, *discarded* or the user should be *asked* each time (see chapter 10.3)
- *auto hide controls* – sets the time interval (in milliseconds) after that the *floating controls* (see chapter 9) become hidden again when the mouse cursor stops moving (only applied when they are set to appear on mouse move – see previous section)

## 14.3 Configuring automatic aligner

The ALIGNER pane of the settings dialog allows you to configure external automatic aligners. However, *INTERTEXT* needs to know how to import and interpret the results of an automatic aligner, which cannot be easily configured. At the moment, the only implemented method for importing and interpreting results is the *hunalign ladder parser*.



The ALIGNER select box selects the aligner configuration to be edited in the form below. A new aligner can be added by pressing the New button (you will be prompted for its name). Current aligner configuration can be *renamed* or *deleted* using the corresponding buttons.

The field EXECUTE shows the path to the aligner executable. It can be just entered into the field or selected from a file selector dialog by using the button SELECT on the right side. In the same way the temporary working directory can be configured. This directory will be used to export the data, process them by the aligner and to import the resulting output from.

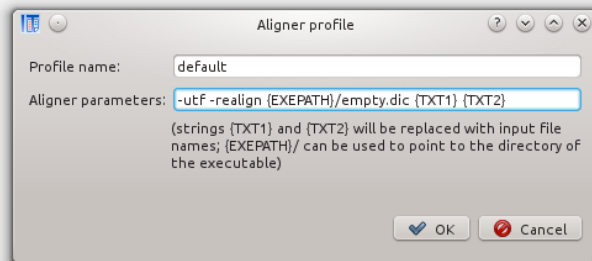
*Export parameters* can be configured for the aligner. *INTERTEXT* expects that the aligner needs the plain text contents of all the alignable elements. You can define the

element separator, parent element separator (e.g. paragraph break, if the aligner can use it for better alignment efficiency), a header and footer (if required by the aligner).

The method used to interpret the result of the aligner must be selected. At the moment, the only implemented option is the *Hunalign ladder parser* which imports the “ladder” output from *hunalign*. The option *Plain 1:1 aligner* selects the fake internal “aligner” creating a basic linear 1:1 alignment. It does not run any external tool and ignores all other settings.

Below, you may configure different profiles – e.g. different configurations for the aligner when run with different specific language combinations (e.g. to force it to use some dictionary, lemmatizer, etc.). When running an automatic aligner, you may always manually select which profile (configuration) to use.<sup>1</sup> New profiles can be *added* or *removed* from the list.

The profile may be edited in a separate dialog window:



There are just two text fields in the profile form: the profile *name* and specific command line *parameters* to run the aligner with when the profile is selected. In the parameters, one can use special placeholders which will be replaced with current values: {TXT1} and {TXT2} will be replaced with the names of the files containing the text contents exported from the first and second text version (according to the aligner’s export settings in the main form); {EXEPATH}/ can be used to insert the path to the aligner executable if you want to locate some additional files placed in the same directory.

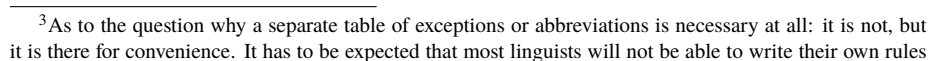
## 14.4 Configuring sentence splitter

The SPLITTER pane of the settings dialog allows you to configure different profiles for the integrated rule-based sentence splitter, which can be applied to the imported texts when creating new alignments.<sup>2</sup>

At the top you may choose the profile to be shown and edited in the rest of the form, add a *new* profile or *rename* or delete the current one. The form then consist of two lists: the list of regular expression rules and the list of exceptions (“Abbreviations” is the title, because that is the most common use for exceptions when segmenting into sentences). The method will then apply all the replacement rules from the first table on the whole

<sup>1</sup> See chapter 10.2.

<sup>2</sup> Of course it is not limited to splitting sentences only – the text can be split into any custom text units which can be defined by clear formal rules using regular expressions.



break inappropriately created by the rules. Therefore the order is not important here. You can only add *new* items to the list, or *edit* or *delete* the selected ones.

## 14.5 Configuring *INTERTEXT SERVER* connection

The **SERVER** pane of the settings dialog allows you to configure the connection to your *INTERTEXT SERVER*. The configuration is simple and it has already been fully described in chapter 8.3.

You may add several *new* servers you want to connect to, or *rename* or *delete* the selected configuration. All configured servers will appear as items in the submenu **ALIGNMENT > REMOTE SERVER**.

## 14.6 Configuring text export formats

The **EXPORT** pane of the settings dialog allows you to configure profiles for exporting text contents of your alignments into custom formats. All configured profiles appear as items in the submenu **ALIGNMENT > EXPORT TEXTS AS**. This method only allows you to export the text contents of the alignable text elements, their boundaries (e.g. sentence boundaries) and parent element boundaries (e.g. paragraph breaks) and possibly also the alignment grouping into segments (segment boundaries), but nothing more from the XML structure or metadata your text may possibly contain.<sup>4</sup>

The **PROFILE** selectbox is used to switch the profile currently shown (and accessible for editing) in the rest of the form. You can create *new* profiles, or *rename* or *delete* the currently selected one. With the **NEW** button you can create a new profile using the values from the currently selected one, so that you can easily create modified copies of the existing profiles.

The form is too long to fit into the window, so that it has to be rolled through using the mouse.

First you have to define general settings for the *file*: a *header* and *footer* enclosing the whole contents, the default *file extension* and encoding you want to use for export (“UTF-8” is the most common Unicode encoding, at least for western languages, but obsolete software such as ParaConc may require you to use some legacy language specific 8-bit encoding).

Next, you have to declare how to treat *segments*: first of all choose whether to *keep segments* at all or just ignore the alignment and only care about the text contents. If you want to keep the segments, you may either enter the strings you want to be inserted at the *start* and *end* of each segment<sup>5</sup> or alternatively just a segment *separator*<sup>6</sup>. You can also turn on the options to *skip empty segments* in case you want to avoid exporting segments missing text (units) in any of the two versions, or *skip unconfirmed segments* to only export confirmed segments. The *empty segment filler* is a text string which will be inserted into otherwise empty segments (segments not containing any text units).

for segmentation or modify the default ones, and hopefully they often will not need to, but nearly always they will want to add or remove exceptions (e.g. abbreviations), which is quite easy using this separation.

<sup>4</sup>For that purpose you would need a custom XSLT template applied to the standard export from *INTERTEXT*. But such processing is beyond the ambitions of *INTERTEXT*.

<sup>5</sup>Such as the starting `<seg>` and ending `</seg>` tags for ParaConc.

<sup>6</sup>Such as the *new-line* character for the *new-line* aligned format



Then you can configure how to mark *element* (sentence) boundaries in the output. Again, you can either define text strings to be placed at the *start* and *end* of each element, or you can just define an element *separator* string.

Next, you can choose how to treat parent elements (*containers*, e.g. paragraphs) of the alignable text elements. You can choose whether to *keep* their breaks at all and then possibly which particular string sequence to use as container *separator*.

You can also define how to treat the two *text versions*: select whether you want to *export both text versions into single file* (e.g. for TMX) or export them separately in two files (e.g. for ParaConc). If you export both versions into a single file, the contents from both versions will be exported segment by segment, all text units from both versions in a single segment, but separated by the *separator* defined here.

In addition, you may define a set of *text replacement rules* which will be applied to the text contents of each text unit before it is exported.<sup>7</sup>

In all text fields you may use regular expressions and standard C-style symbols for special characters (see appendix A). You may as well use the following placeholders to be replaced with actual values from the exported texts:

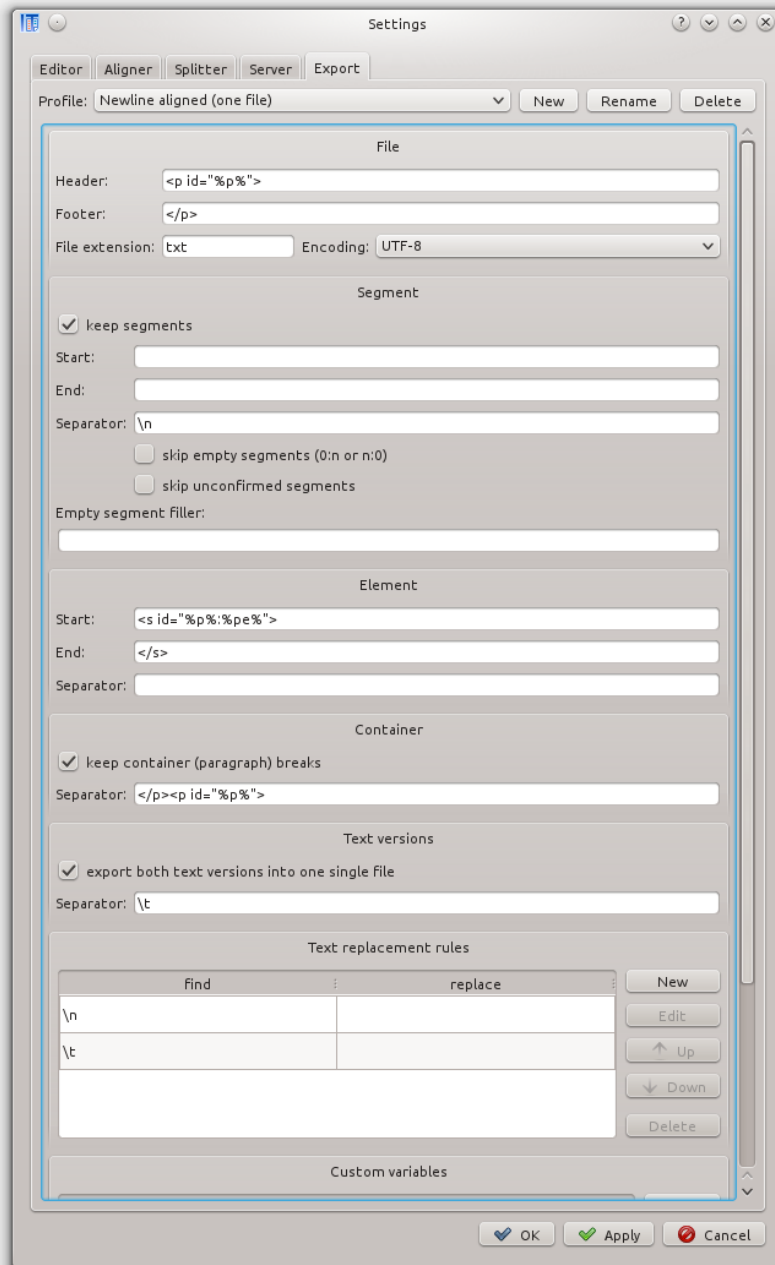
- %t% – text name
- %v% – current version name
- %v1% – left side version name
- %v2% – right side version name
- %s% – segment number (continuously incremented)
- %p% – parent element/container (paragraph) number (continuously incremented)
- %e% – element (sentence) number (continuously incremented)
- %pe% – element (sentence) number within the current container (paragraph)

Not all placeholders make sense in all fields, of course.

You may define your own additional placeholders, or *custom variables*, at the bottom of the form. You will always be asked to enter custom values for these variables before each export. The table shows the list of the custom variables in three columns: *symbol* is the string used as the placeholder (i.e. the string to be replaced with the actual value), *description* is the description shown in the prompt dialog when asking you to enter the custom value and *default value* is the value which will be pre-entered by default in the dialog.

---

<sup>7</sup>The form is here based on the same principles as the rules defined in the configuration of sentence splitter in chapter 14.4.



## **Part IV**

# **Appendix**

## Appendix A

# Regular expressions and special characters

When entering strings in *INTERTEXT* (search, configuration forms, etc.), you may use standard C symbols for any special characters, such as:

- `\n` for unix new-line break
- `\r` for *carriage-return* (used in MS DOS/Windows line break combination `\r\n`)
- `\t` to insert a TAB character

Where regular expressions can be used, the expressions supported by the *Qt toolkit* are available, including back-references. For more details visit <http://qt-project.org/doc/qt-4.8/qregexp.html#details>. Unfortunately, *Qt* currently does not support Unicode character classes.

Back-references can be used in replacement strings as well, using the sequence `\#`, where `#` is the ordinal number of the text captured by the (search) regular expression's grouping parentheses (i.e. `#`-th opening parenthesis from the left). The replacement string may also contain the line-break sequence `\n`. With the double line-break sequence (`\n\n`), you can also use the search and replace to split existing elements!

## Appendix B

# TEI XML alignment file structure

While the text documents used by *INTERTEXT* can be just any arbitrary (valid) XML files, the format of the alignment is fixed. *INTERTEXT* uses a slightly extended TEI XML format for stand-off alignment as TCA2. The structure is as follows:

The file must contain one single root element called `linkGrp` with two attributes: `toDoc` and `fromDoc`. Their values must be the filenames of the two separate documents. The `fromDoc` attribute refers to the left side document, the `toDoc` attribute to the right side document.

The `linkGrp` element contains `link` elements, each corresponding to one single position (segment) in the alignment, with the following attributes:

- `xtargets` is a space and semicolon separated list of ID-values of elements which are grouped into the same segment; first there is a space-separated list of element IDs from the `toDoc` document, then follows the semicolon and after that a space-separated list of element IDs from the `fromDoc` document<sup>1</sup>
- `status` is an optional attribute indicating the status of the link – known values are:
  - `man` for manually confirmed link
  - `auto` for automatically aligned segments
  - `plain` for unaligned / unconfirmed / unknown status
- `mark` is used internally to preserve user bookmarks from the editor, only values 0 and 1 are known at the moment, but for the value 0 no attribute is generated by *INTERTEXT* at all
- `type` is only generated on export for convenience, it gives a dash separated count of elements grouped together by the link (e.g. "1-2"); it is ignored on import

---

<sup>1</sup>Do not let yourself confuse by the fact that the blocks of IDs have the opposite order than on the screen: first (i.e. on the left side of the semicolon) come the *right* side IDs and then (right of the semicolon) the *left* side IDs.