Technical Notes on FLEx Text Interlinear

August 27, 2025

Ken Zook

Contents

Technical Notes on FLEx Text Interlinear 1
1 Introduction 1
2 Flextext file structure 2
2.1 Flextext file 2
2.2 Interlinear-text 2
2.3 Paragraph 3
2.4 Phrase 4
2.5 Word 5
2.6 Morph 7
3 Mapping FLEx interlinear to flextext fields 7
4 FLEx Text export 8
5 FLEx Text import 8
5.1 Merge flextext into existing text 9
5.2 Create a new text from flextext 10
6 Sample data 12
6.1 File from FLEx: 12
6.2 File from ELAN 14
6.3 File from SayMore 15
7 FLEx Text XML Schema 16

1 Introduction

FLEx Text is an XML interchange format for importing and exporting Interlinear text in FieldWorks Language Explorer (FLEx) (https://software.sil.org/fieldworks) and other programs, such as ELAN (https://archive.mpi.nl/tla/elan) and SayMore (https://software.sil.org/saymore/). The file extension is flextext. A flextext file may hold multiple interlinear texts.

For information on how interlinear texts are stored in FLEx, see https://downloads.languagetechnology.org/fieldworks/Documentation/FLEx%209.1%20Conceptual%20Model.pdf.

When exporting from FLEx to flextext, the program is not directly accessing the FLEx data, but is exporting the data from the configured interlinear text view (Tools > Configure > Interlinear). By altering the options in Interlinear Configuration, it will alter the data and order that is included in the flextext file. Before exporting, make sure you have your interlinear text displaying the fields and writing systems you want to include in the flextext file. When importing flextext data to FLEx, it will import the data in the flextext file, but will only display data that is currently set in Interlinear Configuration. When importing a flextext file, make sure your project has the writing systems used in the flextext file.

2 Flextext file structure

This section does not describe every possibility in a flextext file, but covers the common elements. For a full description, see the XML Schema in section 77.

2.1 Flextext file

The flextext file is an XML file with this basic structure.

```
<?xml version="1.0" encoding="utf-8"?>
<document version="2">
  <interlinear-text guid="db49c7a0-63fc-4a0a-bb9f-d3def0feb543">
  </interlinear-text>
</document>
```

The file may have an optional U+FEFF BYTE ORDER MARK character at the beginning. There is one document element with an optional version attribute which is currently 2 In FLEx 9.2.11. A flextext file may contain multiple interlinear-text elements. Each of these elements represents one interlinear Text object in FLEx.

2.2 Interlinear-text

An interlinear-text element has this basic structure.

```
 <interlinear-text guid="db49c7a0-63fc-4a0a-bb9f-d3def0feb543">
   <item type="title" lang="en">The boy story</item>
   <paragraphs>
     <paragraph guid="904ea0c5-975f-400c-94be-0bb30f356b05">
     </paragraph>
   </paragraphs>
   <languages>
     <language lang="en" font="Times New Roman" />
     <language lang="fr" font="Times New Roman" vernacular="true" />
   </languages>
   <media-files offset-type="">
     <media guid="df7d67a9-00a6-4ed8-9241-c6177850686f"
location="file:///C:/Users/pa/Desktop/Collections/story.wav"/>
   </media-files>
 </interlinear-text>
```

An interlinear-text has

- An optional guid attribute that holds the guid of the FLEx Text object when a text is exported from FLEx to flextext. This is used during importing a flextext file to determine whether FLEx already has the text being imported, and if so it offers to merge the flextext data into the current Text, or create a new Text object with a new guid during the import.
- There can be any number of item elements with type “title”; one for each language (writing system). This is the title for the text.
- There is a single required paragraphs element containing zero or more paragraphs.
- There is also an optional languages element containing language elements. There should be one language element for each language (writing system) used in the interlinear-text. Each one has a “lang” attribute defining the writing system, an optional “font” attribute giving the default font for the writing system, and an optional

"vernacular" attribute set to "true" for the primary vernacular writing system for the text baseline.

- There is an optional media-files element with an "offset-type" attribute (normally null string: ""), and zero or more media elements with a "guid" attribute, and a "location" attribute giving the full path to the media file.

While FLEx does not provide UI for media files, it does store the data in the fwdata file so that it can roundtrip the information via flextext to ELAN, etc. Note that the interlinear-text guid and the media guid are maintained when importing from flextext, but the paragraph guids are not maintained.

This is the relevant data in fwdata for saving this information

```
<rt class="Text" guid="db49c7a0-63fc-4a0a-bb9f-d3def0feb543">
<MediaFiles>
<objsur guid="308cb504-a117-4441-a2cb-0d35ad7c5923" t="o" />
</MediaFiles>
</rt>
```

The interlinear text is in a Text object. The Text owns a single CmMediaContainer in the MediaFiles property.

```
<rt class="CmMediaContainer" guid="308cb504-a117-4441-a2cb-0d35ad7c5923"
ownerguid="db49c7a0-63fc-4a0a-bb9f-d3def0feb543">
<MediaURIs>
<objsur guid="df7d67a9-00a6-4ed8-9241-c6177850686f" t="o" />
</MediaURIs>
</rt>
```

The CmMediaContainer owns zero or more CmMediaURI objects.

```
<rt class="CmMediaURI" guid="df7d67a9-00a6-4ed8-9241-c6177850686f"
ownerguid="308cb504-a117-4441-a2cb-0d35ad7c5923">
<MediaURI>
<Uni>file:///C:/Users/pa/Desktop/Collections/story.wav</Uni>
</MediaURI>
</rt>
```

The CmMediaURI has a Unicode string in the MediaURI property for the path to the media file.

2.3 Paragraph

A paragraph element has this structure.

```
<paragraph guid="e3a65332-afe5-49f0-afc4-334871e0a663">
  <phrases>
    <phrase guid="e637f22c-9020-4bd7-8292-48b3dd6823fa">
    </phrase>
  </phrases>
</paragraph>
```

The paragraph element has an optional guid attribute which corresponds to the StTxtPara object that was exported from FLEx to flextext.

A paragraph element has a single phrases element that holds any number of phrase elements.

2.4 Phrase

A phrase element from FLEx has this structure.

```
<phrase guid="e637f22c-9020-4bd7-8292-48b3dd6823fa">
  <item type="txt" lang="fr">Le garçon.</item>
  <item type="segnum" lang="en">1</item>
  <words>
    <word guid="e9a04fd4-6f47-48ae-97b5-9344c6f78049">
    </word>
  </words>
  <item type="gls" lang="en">The boy.</item>
</phrase>
```

The phrase element has

- An optional guid attribute which corresponds to the Segment in FLEx.
- Since a FLEx baseline is a String instead of MultiString, there will only be one item element with type “txt” with the vernacular writing system coming from the Word interlinear line, and this is optional as long as there are word elements specifying the text. Also, as of FW9.2.11, flextext does not support writing system and style embeddings in the baseline even though these are supported in FLEx.
- There is an optional item with type “segnum”. This is the segment number which in FLEx is calculated automatically based on paragraph and segment numbers, and appears at the beginning of the Paragraph in the Word interlinear line.
- There can be any number of item elements with type “gls”; one for each analysis writing system. This is the free translation from the Free interlinear line (FLEx Segment FreeTranslation).
- There can be any number of item elements with type “lit”; one for each analysis writing system. This is the literal translation from the Lit. interlinear line (FLEx Segment LiteralTranslation.
- There can be any number of item elements with type “note”; one for each analysis writing system. This is the Note from the Note interlinear line (FLEx Segment Notes Note Content).
- There is an optional single words element that holds any number of word elements.

Translation and Note fields in FLEx are MultiStrings, meaning a user can have translations in multiple writing systems and can embed writing systems and styles in these strings. However, as of FW9.2.11, flextext does not support this embedding. It does provide for translations and notes in different writing systems, but the entire string will be in one writing system with no embedded styles.

There is also an issue with notes when you use multiple writing systems. A separate “note” item will be included for each translation. If you start out with two notes, each with two writing systems, the export will have four note items:

```
<item type="note" lang="en">This is a note about this Bold sentence.</item>
<item type="note" lang="es">This is the Spanish note with English embedded.</item>
<item type="note" lang="en">This is a second English note.</item>
<item type="note" lang="es">The second Spanish</item>
```

Since flextext does not currently have a way to indicate that these are really two notes, when it gets imported into FLEx, the result will be four notes, each with one writing system, instead of two notes.

In ELAN and SayMore files, additional phrase attributes are used to display information about the location in a media file.

```
<phrase begin-time-offset="316926"
   end-time-offset="317266"
   guid="76713b5a-2ee6-4b6e-938a-e6f5af498cb3"
   media-file="df7d67a9-00a6-4ed8-9241-c6177850686f" speaker="John Doe">
   <item lang="ew-fonipa" type="txt">dona</item>
   <item lang="en" type="gls">It is finished.</item>
</phrase>
```

Attributes “begin-time-offset” and “end-time-offset” specifies an index into a media file for this phrase. A “media-file” attribute holds the guid for the media for this segment. The media file information should match one of the media elements in the media-files element of the interlinear-text.

While FLEx does not provide UI for media file information, it does store the data in the fwdata file so that it can roundtrip the information via flextext. The phrase guid is maintained in the Segment guid during import. These are the relevant fields in the Segment.

```
<rt class="Segment" guid="76713b5a-2ee6-4b6e-938a-e6f5af498cb3" ownerguid="451dcf94-
d3cc-4b1c-b919-4d487a25ad06">
<BeginTimeOffset>
<Uni>316926</Uni>
</BeginTimeOffset>
<EndTimeOffset>
<Uni>317266</Uni>
</EndTimeOffset>
<MediaURI>
<objsur guid="df7d67a9-00a6-4 ed8-9241-c6177850686f" t="r" />
</MediaURI>
<Speaker>
<objsur guid="d0d0dda8-7354-431d-bba0-739bed01a2c3" t="r" />
</Speaker>
</rt>
```

The time offsets are millisecond values stored as Unicode strings in BeginTimeOffset and EndTimeOffset. The MediaURI property references a single CmMediaURI object in the Text. The Speaker property references a single CmPerson object in the People list. On import, CmPerson objects will be added to the People list if they are not already present.

2.5 Word

When exporting FLEx data to flextext, it will only export data that you have enabled in Interlinear Configuration.

Word elements are not required. If there are multiple words in the phrase “txt” item, and there are no Word elements in flextext, FLEx will parse the txt item into unanalyzed wordforms during import. FLEx allows multiple words in a wordform (e.g., “kick the bucket”). This will be exported correctly as long as there are word elements with the

multiple words. However, if you are importing a baseline without word elements, each word will be a separate wordform.

An unanalyzed word element has this structure.

```
<word guid="e124f206-d156-4c2c-b046-457ee720b143">
   <item type="txt" lang="fr">Le</item>
</word>
```

The FLEx Analyses link is to the WfiWordform containing the form of the word. In flextext, the word has one or more item elements with type “txt” and “lang” attribute for the writing system; one for each vernacular writing system. The body of the item is the Word interlinear line (FLEx WfiWordform Form).

The FLEx Analyses link is to the WfiGloss for a fully analyzed word. In flextext, the word element has this structure.

```
<word guid="e9a04fd4-6f47-48ae-97b5-9344c6f78049">
   <item type="txt" lang="fr">garçon</item>
   <morphemes>
      <morph type="stem" guid="d7f713e8-e8cf-11d3-9764-00c04f186933">
      </morph>
   </morphemes>
   <item type="gls" lang="en">boy</item>
   <item type="pos" lang="en">n</item>
 </word>
```

The word element has an optional guid attribute which corresponds to the WfiGloss object in FLEx. The word element has the following content

- one or more “txt” items; one for each vernacular writing system coming from the Word interlinear line (FLEx WfiWordform Form).
- a morphemes element that holds zero or more morpheme elements within the FLEx WfiWordform Analyses.
- zero or more “gls” items for the word gloss; one for each analysis writing system from the Word Gloss interlinear line (FLEx WfiGloss Form).
- zero or more “pos” items for the word category; one for each analysis writing system that comes from the Word Cat. interlinear line (FLEx PartOfSpeech referenced from WfiAnalysis Category).

The FLEx Analyses link is to the WfiAnalysis for a partially analyzed word. The export to flextext is similar to a fully analyzed word except the “gls” element will usually be missing.

A punctuation (or text in a writing system other than the top vernacular writing system) has this structure

```
<word>
   <item type="punct" lang="fr">.</item>
</word>
```

The guid is not shown here, but the FLEx object being exported is a PunctuationForm object that stores the punctuation text and writing system. The punctuation in flextext is exported as an item with type “punct” and “lang” giving the writing system code. The

content of the item is the punctuation or text that is in a writing system other than the top vernacular writing system (FLEx PunctuationForm Form).

2.6 Morph

A morph element represents a morpheme within a word and has this structure.

```
<morph type="stem" guid="d7f713e8-e8cf-11d3-9764-00c04f186933">
  <item type="txt" lang="fr">garçon</item>
  <item type="cf" lang="fr">garçon</item>
  <item type="hn" lang="fr">1</item>
  <item type="gls" lang="en">boy</item>
  <item type="msa" lang="en">n</item>
</morph>
```

The morph element has an optional “type” attribute listing the FLEx MoMorphType Name, and optional “guid” attribute listing the FLEx MoMorphType guid. The morph element has the following content

- one or more “txt” items for the word; one for each vernacular writing system from the Morphemes interlinear line (FLEx LexEntry LexemeForm MoForm Form).
- zero or more “cf” items for the Lex. Entries interlinear line (FLEx LexEntry LexemeForm MoForm Form). (The abbreviation was probably for citation form at some point, but it is really the lexeme form, so should probably be “lf”. However it’s too hard to change at this point.)
- an optional “hm” item for a homograph number from LexEntry HomographNumber property. In the display this is appended to the lexeme form.
- zero or more “gls” items for the sense gloss; one for each analysis writing system from the Lex. Gloss interlinear line (FLEx LexSense Gloss).
- zero or more “msa” items for the grammatical category; one for each analysis writing system from the Lex. Gram. Info. interlinear line (FLEx LexSense MorphoSyntaxAnalsis PartOfSpeech Abbreviation property.

3 Mapping FLEx interlinear to flextext fields

The various fields that get exported from configured data are exported into item elements with a “type” attribute specifying the type of data, and a “lang” attribute specifying the writing system. This table shows the correspondence between the configuration labels and the type used in flextext.

Interlinear Label	Element	Item Type
Word	word	txt
Morphemes	morph	txt
Lex. Entries	morph	cf / hn
Lex. Gloss	morph	gls
Lex. Gram. Info.	morph	msa
Word Gloss	word	gls
Word Cat.	word	pos
Free Translation	phrase	gls
Literal Translation	phrase	lit

Note	phrase	note

FLEx exports guids from the fwdata file to a flextext file. The following chart maps between flextext and FLEx guids:

Flextext Element guid	FLEx Class guid
interlinear-text	Text
paragraph	StTxtPara
phrase	Segment
media	CmMediaURI
word	WfiWordform/WfiAnalysis/WfiGloss
morph-type	MoMorphType

The word guid corresponds to the Segment Analyses reference links. When unanalyzed, this will be a WfiWordform. When partially analyzed, it will be a WfiAnalysis. When fully analyzed, it will be a WfiGloss. Punctuation (or text in a writing system other than the writing system for the text) will be a "punct" word element without a guid even though the underlying PunctuationForm object does have a guid.

When importing and merging, FLEx attempts to update the existing data with the existing guids. When importing as a new text, these guids are ignored and new owning objects with guids will be created.

4 FLEx Text export

When exporting to flextext from FLEx, the program is not directly accessing the FLEx data, but is exporting the data from the Interlinear Text configured view. By altering the options in Interlinear Configuration, it will alter the data that is included in the flextext file.

To export a flextext file from FLEx:

1. In the Texts & Words area, use the Interlinear Texts tool to open an interlinear text.
2. Go to the Gloss, Analyze, Tagging, or Print View tab on the text.
3. Choose File > Export Interlinear from the main menu to open the Export Interlinear dialog.
4. Click the "ELAN, SayMore, FLEx FLEXTEXT" option, then click Export.
5. A Choose Texts dialog will open where you can choose the text(s) to export. (Note in FW9.2.10 this takes about 25 seconds the first time LT- 22247.) The current text is preselected. Click OK to start the export.
6. In the "Export to FLEXTEXText" dialog that opens, specify the directory and file name, then click Save to export the file.

When FLEx exports a text, it always writes out the current guids for text objects including classes for Text, StTxtPara, Segment, and WfiWordform, WfiAnalysis, or WfiGloss, depending on the state of analysis.

5 FLEx Text import

Before importing a flextext file, you should verify that FLEx has the correct vernacular and analysis writing systems used in the file. A flextext import will not create missing

writing systems, and will only display the imported data if the desired writing systems are enabled in configuration.

To import a flextext file into FLEx:

1. From the main menu in the Texts & Words area, choose File > Import > FLEx Text Interlinear
2. Select the flextext file to import in the FieldWorks interlinear (FLEx Text) file dialog and click OK to start the import.
3. If the flextext came from a previous FLEx export, a "Duplicate text found" dialog will ask if you want to merge the flextext text with the existing text.
   Click "Yes" if you want to merge the flextext file with the existing text. This essentially makes the FLEx copy identical to the flextext copy.
   Click No if you want to create a new text.
   If you are importing multiple texts, this dialog will repeat for each duplicate text found. Unfortunately, it doesn't tell which text is being overwritten, so this could be confusing (LT-22167).

Note: FW9.2.2 through FW9.2.10 had a serious bug that damaged existing data during either choice (LT-22249). This has been corrected in FW9.2.11 as described below.

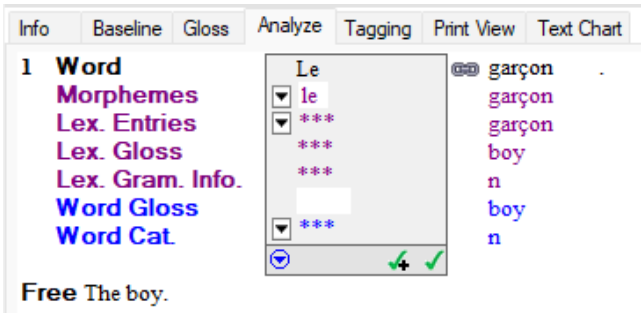
When importing wordforms, if the flextext file has multiple writing systems for words, it will use the primary writing system to find an existing wordform. If the existing wordform does not have writing systems included in the flextext file, the import will add the additional writing systems to the existing wordform. However, if any of the additional writing systems are already in the existing wordform, the writing system from flextext will be ignored rather than overwriting the FLEx WfiWordform writing system.

When importing morphs, if the fields in the flextext morph match corresponding fields in a FLEx analysis (WfiMorphBundle), it will use that analysis in FLEx. If the flextext morph has additional fields that are not in the analysis in FLEx, then a new analysis will be created in FLEx rather than adding to an existing analysis.

5.1 Merge flextext into existing text

When you choose "Yes" to the "Duplicate Text Found" dialog, FLEx will use the guids in the flextext file to attempt to modify any text elements based on the flextext file. This process should not change any existing analyses that could be used by other texts, but it will attempt to make the existing text a copy of the flextext data. Existing analyses should not be changed in this operation. If there are new morphs, it works similar to making a copy of the text. If existing analyses are found that match the flextext morph, the import will link to those analyses during the import. Otherwise, as of FW9.2.11, the wordform will remain unanalyzed. In some later version of FLEx, we hope to create entries and analyses that are new to this text, as long as morph properties are included.

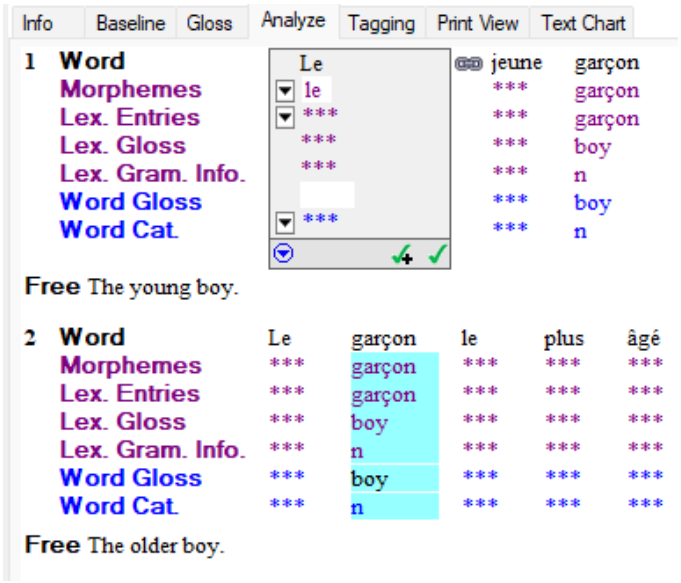
Here is an example of changes in flextext being merged into the existing text. This is the original text.



I exported the above text to flextext, then in flextext:

- Changed the baseline to “Le jeune garçon.”
- Changed the free translation to “The young boy.”
- Added a second paragraph with this baseline “Le garçon le plus âgé.” with no word elements.
- And this free translation “The older boy”

I then imported the modified flextext file, merging into the existing text giving this result.



The second paragraph did not display the interlinear bundles until I went to another text and then back again, so there is a refresh issue, but otherwise it worked as expected.

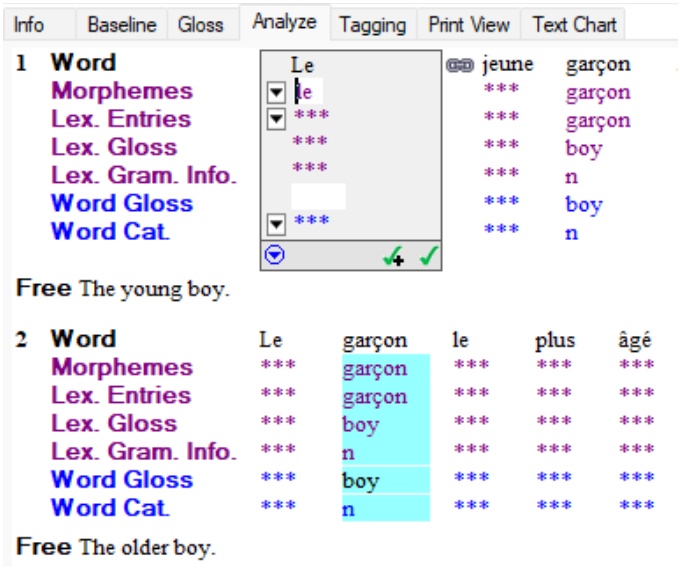
- The original “garçon” wordform remained approved, and an unanalyzed “jeune” was inserted. The original translation was changed to what was in the flextext file.
- The second paragraph was added with the translation. The analysis for “garçon” was not approved, but the guesser had the correct analysis ready to approve.

5.2 Create a new text from flextext

When you choose “No” to the “Duplicate Text Found” dialog, the guids in the flextext file will be ignored and new objects will be created in FLEx for all objects owned by the new text. If existing analyses are found that match the flextext morph, the import will link to those analyses during the import. Otherwise, new analyses will be added to the

wordform. In some later version of FLEx, we hope to create entries and analyses that are new to this text, as long as morph elements are included. Nothing in existing texts with approved analyses should change when creating a new text. For user unapproved analyses, an analysis guesser tries to find an appropriate analysis and display it with a blue background. When importing a new text, even though the actual data for the existing text is not changed, it's possible that the guesser may come up with a different guess. In this case, the original analysis is still intact, and can be selected and approved via the normal manual interlinearization options.

In the above example, when choosing not to merge, the original text was unchanged, and this was the new text.



6 Sample data

6.1 File from FLEx:

This is a simple text in FLEx.

This is the exported flextext file for this simple text.

```
<document version="2">
  <interlinear-text guid="23dab191-81f0-4588-a15b-2246432c993f">
    <item type="title" lang="en">The boy story</item>
    <paragraphs>
      <paragraph guid="e3a65332-afe5-49f0-afc4-334871e0a663">
<phrases>
  <phrase guid="e637f22c-9020-4bd7-8292-48b3dd6823fa">
    <item type="txt" lang="fr">Le garçon.</item>
    <item type="segnum" lang="en">1</item>
    <words>
      <word guid="e124f206-d156-4c2c-b046-457ee720b143">
<item type="txt" lang="fr">Le</item>
<item type="txt" lang="fr-fonipa">lə</item>
      </word>
      <word guid="e9a04fd4-6f47-48ae-97b5-9344c6f78049">
<item type="txt" lang="fr">garçon</item>
<item type="txt" lang="fr-fonipa">ɡaʁsɔ̃</item>
<morphemes>
  <morph type="stem" guid="d7f713e8-e8cf-11d3-9764-00c04f186933">
    <item type="txt" lang="fr">garçon</item>
    <item type="txt" lang="fr-fonipa">ɡaʁsɔ̃</item>
    <item type="cf" lang="fr">garçon</item>
    <item type="gls" lang="en">boy</item>
    <item type="gls" lang="es">niño</item>
    <item type="msa" lang="en">n</item>
  </morph>
</morphemes>
<item type="gls" lang="en">boy</item>
<item type="gls" lang="es">niño</item>
<item type="pos" lang="en">n</item>
      </word>
```

```
      <word>
<item type="punct" lang="fr">.</item>
      </word>
    </words>
    <item type="gls" lang="en">The boy.</item>
    <item type="gls" lang="es">El niño.</item>
    <item type="note" lang="en">This is a note about this sentence.</item>
    <item type="note" lang="es">Esta es una nota sobre esta oración.</item>
  </phrase>
</phrases>
      </paragraph>
    </paragraphs>
    <languages>
      <language lang="en" font="Times New Roman" />
      <language lang="fr" font="Times New Roman" vernacular="true" />
      <language lang="fr-fonipa" font="" vernacular="true" />
      <language lang="es" font="Times New Roman" />
    </languages>
  </interlinear-text>
</document>
```

6.2 File from ELAN

This is a sample flextext file generated in ELAN. Note it includes media file and offset information that is stored in FLEx so that it can be exported from FLEx, but it is not supported in the FLEx UI. Files from ELAN normally do not have morpheme information, unless it was supplied at some point from FLEx.


```
<?xml version="1.0" encoding="UTF-8"?>
<document version="2">
   <interlinear-text guid="db49c7a0-63fc-4a0a-bb9f-d3def0feb543">
<paragraphs>
  <paragraph guid="904ea0c5-975f-400c-94be-0bb30f356b05">
<phrases>
   <phrase begin-time-offset="316926"
end-time-offset="317266"
guid="76713b5a-2ee6-4b6e-938a-e6f5af498cb3"
media-file="df7d67a9-00a6-4ed8-9241-c6177850686f" speaker="John Doe">
<item lang="ew-fonipa" type="txt">dona</item>
<item lang="ne" type="gls">यो समाप्त भयो।</item>
<item lang="en" type="gls">It is finished.</item>
   </phrase>
</phrases>
   </paragraph>
</paragraphs>
<media-files offset-type="">
   <media guid="df7d67a9-00a6-4ed8-9241-c6177850686f"
location="file:///C:/Users/pa/Desktop/Collections/story.wav"/>
</media-files>
   </interlinear-text>
</document>
```

6.3 File from SayMore

This is a sample flextext file generated in SayMore. Note it includes media file and offset information that is stored in FLEx so that it can be exported from FLEx, but it is not supported in the FLEx UI. Also note that these files do not include morpheme information.

```
<?xml version="1.0" encoding="utf-8"?>
<document>
  <interlinear-text>
    <item type="title" lang="fr">Feuille de baobab.wav</item>
    <paragraphs>
      <paragraph>
<phrases>
  <phrase begin-time-offset="68902" end-time-offset="70502" media-file="a0d6501b-15de-
469f-b3e3-e5d585bb5b7c">
    <item type="segnum" lang="fr">19</item>
    <item type="txt" lang="mxl">mo ɛ no dagbe o ne</item>
    <item type="gls" lang="fr">c'est comme ça on prepare</item>
    <words />
  </phrase>
</phrases>
      </paragraph>
    </paragraphs>
    <languages>
      <language lang="mxl" />
      <language lang="fr" />
    </languages>
    <media-files offset-type="">
      <media guid="a0d6501b-15de-469f-b3e3-e5d585bb5b7c"
location="C:\Users\Joe\Documents\SayMore\Maxi recipes\Sessions\Feuille de baobab\Feuille
de baobab.wav" />
    </media-files>
  </interlinear-text>
</document>
```

7 FLEx Text XML Schema

The flextext technical file structure is defined in FlexInterlinear.xsd located in C:\Program Files\SIL\FieldWorks 9\Language Explorer\Export Templates\Interlinear. The content of the xsd is included below. This can be used to validate a flextext file to make sure it has the correct structure.

Unfortunately, FLEx does not give helpful information when a flextext file is not structured properly or has invalid data. A start would be to validate the flextext file to make sure it has valid structure. There are various online and offline tools you can use to validate a file with this XSD schema. One online option is https://www.freeformatter.com/xml-validator-xsd.html

If this does not solve the problem, you'll need to get technical help from flex-errors@sil.org to import the data in a debugger to determine where it is failing, or you could attempt to use the dnSpy debugger without having a FLEx development system. See section 3.5 of https://downloads.languagetechnology.org/fieldworks/Documentation/FLEx%209.1%20Conceptual%20Model.pdf for more information on this.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" id="document">
  <xs:element name="document">
 <xs:complexType>
   <xs:sequence>
  <xs:element name="interlinear-text" minOccurs="1" maxOccurs="unbounded">
    <xs:complexType>
   <xs:sequence minOccurs="0" maxOccurs="unbounded">
     <xs:element ref="item" minOccurs="0" maxOccurs="unbounded"/>
     <xs:element name="paragraphs" minOccurs="1" maxOccurs="1">
    <xs:complexType>
      <xs:sequence minOccurs="1" maxOccurs="1">
     <xs:element name="paragraph" minOccurs="0" maxOccurs="unbounded">
       <xs:complexType>
      <xs:all>
<xs:element name="phrases" minOccurs="1" maxOccurs="1">
       <xs:complexType>
 <xs:sequence>
<xs:element name="phrase" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
 <xs:sequence>
   <xs:element ref="item" minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="words" minOccurs="1" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
   <xs:element name="scrMilestone" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
    <xs:attribute name="chapter" type="xs:integer" use="required" />
    <xs:attribute name="verse" type="xs:integer" use="required" />
     </xs:complexType>
   </xs:element>
   <xs:element name="word" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
```

```
      <xs:element ref="item" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="morphemes" minOccurs="0" maxOccurs="1">
     <xs:complexType>
       <xs:sequence>
      <xs:element name="morph" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
      <xs:sequence>
      <xs:element ref="item" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="type" type="morphTypes"/>
      <xs:attribute name="guid" type="xs:string"/>
</xs:complexType>
     </xs:element>
      </xs:sequence>
      <xs:attribute name="analysisStatus" type="analysisStatusTypes" use="optional"/>
    </xs:complexType>
     </xs:element>
   </xs:choice>
   <xs:attribute name="guid" type="xs:string"/>
   <xs:attribute name="type" type="xs:string" fixed="phrase" use="optional" />
    </xs:complexType>
  </xs:element> <!-- word -->
   </xs:sequence>
 </xs:complexType>
  </xs:element> <!-- words -->
  <xs:element ref="item" minOccurs="0" maxOccurs="unbounded"/>
 </xs:sequence>
 <!-- media-file should match a guid of one of the media elements for the encl. text -->
 <xs:attribute name="media-file" type="xs:string" use="optional"/>
 <xs:attribute name="begin-time-offset" type="xs:string" use="optional"/>
 <xs:attribute name="end-time-offset" type="xs:string" use="optional"/>
 <xs:attribute name="guid" type="xs:string"/>
 <xs:attribute name="speaker" type="xs:string" use="optional"/>
  </xs:complexType>
</xs:element> <!-- phrase -->
 </xs:sequence>
      </xs:complexType>
</xs:element> <!-- phrases -->
     </xs:all>
     <xs:attribute name="guid" type="xs:string"/>
      </xs:complexType>
    </xs:element> <!-- paragraph -->
     </xs:sequence>
   </xs:complexType>
    </xs:element> <!-- paragraphs -->
    <xs:element name="languages" minOccurs="0" maxOccurs="1">
   <xs:complexType>
     <xs:sequence>
    <xs:element name="language" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
     <xs:attribute name="lang" type="xs:string" use="required"/>
     <xs:attribute name="encoding" type="xs:string" use="optional"/>
     <xs:attribute name="font" type="xs:string"/>
     <xs:attribute name="vernacular" type="xs:boolean"/>
      </xs:complexType>
    </xs:element>
```

```
      </xs:sequence>
    </xs:complexType>
    <xs:key name="langId">
      <xs:selector xpath="language"/>
      <xs:field xpath="@lang"/>
    </xs:key>
     </xs:element>
     <xs:element name="media-files" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
      <xs:sequence>
     <xs:element name="media" minOccurs="0" maxOccurs="unbounded">
       <xs:complexType>
      <xs:attribute name="guid" type="xs:string" use="required"/>
      <!-- location should follow the media fragment specification -->
      <xs:attribute name="location" type="xs:anyURI" use="required"/>
       </xs:complexType>
     </xs:element>
      </xs:sequence>
      <xs:attribute name="offset-type" type="xs:string"/>
    </xs:complexType>
     </xs:element>
   </xs:sequence>
   <xs:attribute name="guid" type="xs:string" use="optional"/>
   <xs:attribute name="scrSectionType" type="scrSectionTypes" use="optional"/>
   <xs:attribute name="scrBook" type="xs:string" use="optional"/>
    </xs:complexType>
  </xs:element> <!-- interlinear-text -->
   </xs:sequence>
   <xs:attribute name="exportSource" type="xs:string" use="optional"/>
   <xs:attribute name="exportTarget" type="xs:string" use="optional"/>
   <xs:attribute name="version" type="xs:string"/>
 </xs:complexType>
  </xs:element> <!-- document -->
  <xs:element name="item" nillable="true">
 <xs:complexType>
   <xs:simpleContent>
  <xs:extension base="xs:string">
    <xs:attribute name="type" type="itemTypes" use="required"/>
    <xs:attribute name="lang" type="xs:string" use="required"/>
    <xs:attribute name="analysisStatus" type="analysisStatusTypes" use="optional"/>
  </xs:extension>
   </xs:simpleContent>
 </xs:complexType>
  </xs:element>
 <xs:simpleType name="knownItemTypes">
 <xs:restriction base="xs:string">
   <xs:enumeration value="txt"/>
   <xs:enumeration value="cf"/>
   <xs:enumeration value="hn"/>
   <xs:enumeration value="variantTypes"/>
   <xs:enumeration value="gls"/>
   <xs:enumeration value="msa"/>
   <xs:enumeration value="pos"/>
   <xs:enumeration value="title"/>
   <xs:enumeration value="title-abbreviation"/>
   <xs:enumeration value="source"/>
```

```
    <xs:enumeration value="comment"/>
    <xs:enumeration value="text-is-translation"/>
    <xs:enumeration value="description"/>
    <xs:enumeration value="punct"/>
  </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="itemTypes">
 <xs:union memberTypes="knownItemTypes xs:string"/>
 </xs:simpleType>
 <xs:simpleType name="morphTypes">
 <xs:restriction base="xs:string">
    <xs:enumeration value="particle"/>
    <xs:enumeration value="infix"/>
    <xs:enumeration value="prefix"/>
    <xs:enumeration value="simulfix"/>
    <xs:enumeration value="suffix"/>
    <xs:enumeration value="suprafix"/>
    <xs:enumeration value="circumfix"/>
    <xs:enumeration value="clitic"/>
    <xs:enumeration value="enclitic"/>
    <xs:enumeration value="proclitic"/>
    <xs:enumeration value="bound root"/>
    <xs:enumeration value="root"/>
    <xs:enumeration value="bound stem"/>
    <xs:enumeration value="stem"/>
    <xs:enumeration value="infixing interfix"/>
    <xs:enumeration value="prefixing interfix"/>
    <xs:enumeration value="suffixing interfix"/>
    <xs:enumeration value="phrase"/>
    <xs:enumeration value="discontiguous phrase"/>
  </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="scrSectionTypes">
 <xs:restriction base="xs:string">
    <xs:enumeration value="title"/>
    <xs:enumeration value="heading"/>
    <xs:enumeration value="verseText"/>
  </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="analysisStatusTypes">
 <xs:restriction base="xs:string">
    <xs:enumeration value="humanApproved"/>
    <xs:enumeration value="guess"/>
    <xs:enumeration value="guessByHumanApproved"/>
    <xs:enumeration value="guessByStatisticalAnalysis"/>
  </xs:restriction>
 </xs:simpleType>
</xs:schema>
```