

XLIFF:doc

The XLIFF Representation Guide for Documents

Introduction to XLIFF:doc	4
Why Use XLIFF:doc?	4
Getting Started	5
Conventions	5
Using XLIFF:doc	6
Project Meta Data	7
Organizing Source Files into XLIFF:doc Files	8
Multiple Source Files vs Single Source File	8
One Target Language Per XLIFF	8
Structure (Format), Content (Text), and Previews	9
Overview	9
Skeleton (Structure)	9
Content (Text)	9
HTML Previews	10
Joining and Splitting Translation Units	12
Limiting Joining and Splitting at the File Level	12
Limiting Joining at the Translation Unit Level	12
Describing Translation State	13
Translation Status	13
Recording Match Quality After a Match has been Applied	14
Last Edited Information	14
Usage Example: Translation Attributes and Workflows	14
Inline Formatting	16
Interpreting Tag Content for Users	16
Providing Text Representations of Standalone Tags	16
Providing HTML Representations of Tags	17
Notes	18
Translation Matches	19
Rules for TM Matches in <alt-trans>	19
Describing Match Quality	19
Describing Attributes of the Match	20
Usage Example	20

Terminology Matches	22
Term Embedding Mechanism	22
Embedding Term Matches	22
UTX as Supported by XLIFF:doc	23
Reading and Displaying Embedded Term Matches	24
QA Information	25
Usage Example 1	25
Usage Example 2	26
Revision History	27
Usage Example	27
Appendix A: Diagram of XLIFF:doc Elements	28
Appendix B: Reference Guide to XLIFF:doc	29
<alt-trans>	29
<body>	30
<dx:attribute>	30
<dx:attributes>	31
<dx:content>	31
<dx:external-preview>	31
<dx:internal-preview>	32
<dx:qa-hit>	32
<dx:qa-hits>	33
<dx:term-hit>	34
<dx:term-hits>	34
<dx:utx-comment>	34
<dx:utx-glossary>	34
<dx:utx-src>	35
<dx:utx-term>	35
<dx:utx-tgt>	36
<external-file>	36
<file>	36
<g>	38
<group>	39
<header>	39
<note>	39
<skl>	40
<source>	40

Interoperability

<target>	40
<tool>	41
<trans-unit>	41
<x>	43
<xliff>	44

Appendix C: [TEMP:] Misc. Questions 46

License	46
Ability to Create Target Files from the XLIFF (including structure in xliff)	46

Introduction to XLIFF:doc

XLIFF:doc is a reference guide to representing documents in XLIFF files. The primary goal of XLIFF:doc is to provide a mechanism to allow lossless exchange of XLIFF data between tools in the localization workflow.


Interoperability is achieved by:

- Using the shared namespace extension dx: with defined attributes and attribute values.
- Using only a subset of the XLIFF 1.2 specification, with pre-defined attribute values.

XLIFF:doc is designed to support the following file types:

- XML document file formats (DITA, structured Framemaker, etc.)
- Office documents (.doc, .docx, .rtf, .otd, .xls, etc.)
- FrameMaker
- InDesign
- HTML and XHTML

XLIFF:doc may not be ideal for every document type or localization challenge. For software string formats, the XLIFF:software representation guide is recommended. For web and interactive formats (Flash, HTML5, etc.), the XLIFF:web representation guide is recommended.

 *Note: the above paragraph contains forward looking statements. As of this time, XLIFF:software and XLIFF:web do not exist.*

Why Use XLIFF:doc?

XLIFF:doc is not the only way to write a bilingual localization file. It is not the only way to write XLIFFs. So why should you support it in your tool?

- Interoperability: Any tool that follows the XLIFF:doc representation guide can read and write XLIFF:doc files from your tool, with no data loss
- Less is more: XLIFF:doc only uses a fraction of the XLIFF 1.2 specification, so you can have a robust XLIFF solution without needing to support every last element and attribute of the XLIFF format.
- Tailor-made for documentation: XLIFF:doc was designed specifically to handle the translation of documents. The elements, attributes, and attribute values in the XLIFF:doc namespace ("dx:") bridge the gap between the promise of the XLIFF format, and the reality of needing to store more information than is provided for by the XLIFF 1.2 specification.
- Embedded terminology: XLIFF:doc provides an easy way for tools that are not terminology-aware, and are not hooked up to terminology databases, to show the linguist relevant terminology hits.
- Create previews from any tool: Any XLIFF:doc-aware tool can generate formatted previews that approximate the final file format, using either source or translation text.
- Fully-documented: You do not need to reverse engineer anything in order to start using XLIFF:doc. All elements, attributes, and even attribute values are fully documented, both for the subset of the XLIFF spec. used, and for the XLIFF:doc custom namespace extension.

Interoperability

Getting Started

It is not necessary to study the XLIFF 1.2 specification in detail before reading this document. However, readers should be comfortable with the relevant portions of the XLIFF 1.2 specification before developing any solutions based on XLIFF:doc.

The Using XLIFF:doc chapter is meant to be a guide to creating and reading XLIFFs. The next two chapters contain in-depth reference on the individual elements and attributes that make up the XLIFF:doc representation guide.

Conventions

Throughout this document, "Creation Tool" refers to a tool that creates an XLIFF:doc file from one or more source files. This would include segmentation, it may or may not include TM matching, TB matching, pre-translation, etc.

"Processing Tool" refers to any tool that reads information from an already-created XLIFF:doc file. Tools that modify XLIFF:doc files are Processing Tools, in the scope of this document.

Names of XLIFF and XLIFF:doc elements are formatted as follows: `<trans-unit>`.

Names of XLIFF and XLIFF:doc attributes are formatted as follows: `dx:qa-hit`.

Using XLIFF:doc

Project Meta Data

[how to store project meta info, where to do it]

[use dx:attributes and dx:attribute to store meta info.]

[should this be entirely freeform? Are there particular meta project attribute/value pairs we need to require tools to support? this part of the spec needs to come from folks working on the handoff between workflow, TM server, and translation editor tools]

[possibly include metrics including word counts, match rate breakdown, etc.]

[proper balance between what goes into the TIP package, and what goes into the XLIFF:doc has not yet been determined.]

[2011/08/12: preliminary decision: will plan to include some meta data in the future, but for 1.0, will leave all metadata in the TIP package, and see how that works in practical testing. We are not sure what meta data needs to be here vs in the TIPP.]

Organizing Source Files into XLIFF:doc Files

Multiple Source Files vs Single Source File

You can bundle multiple source files into a single XLIFF:doc, or you can use a model of one XLIFF:doc per source file.

One Target Language Per XLIFF

Tools processing XLIFF:doc files are expecting exactly one target language per XLIFF:doc. You must not use multiple `<file>` elements to include multiple versions of the source file (one for each target language in the project). If you have multiple target languages, you must create a different XLIFF:doc for each one.

Structure (Format), Content (Text), and Previews

Overview

In most interpretations of XLIFF, non-inline formatting data (structural data) is separated from the text data, and put into a separate file or element called a skeleton. Text data is separated from structural data, and placed into `<trans-unit>` elements. Once the file is translated, the data in the `<target>` element of each `<trans-unit>` is merged back into the skeleton to generate a translated file in the original file format. In an ideal world, every tool that processed an XLIFF file would have the capability to generate a formatted target language file. However, due to the complexity and variety of file formats that need translation, it is not easy to provide a skeleton file that can be used universally. Recognizing this, XLIFF:doc adds a preview element into which the Creation Tool is required to place enough information for any compliant tool to show the user a visual preview.

Skeleton (Structure)

In XLIFF:doc, skeleton files are stored outside of the XLIFF itself, in .skl files. Each XLIFF file will have a corresponding .skl file. This skeleton file can be placed alongside the XLIFF in the translation package, or it can be left on the server or in another location. This choice is up to the user or tool that creates the XLIFF. XLIFF:doc does not require the skeleton file to be available to every tool or user of the XLIFF. XLIFF:doc does not specify how a skeleton file should be created, stored, or structured.

Content (Text)

How exactly a Creation Tool extracts text from a source file and organizes it into `<trans-unit>` elements is not, in general, specified by XLIFF:doc. There is one exception, however: subflows. XLIFF 1.2 allows for subflows, to handle situations like a sentence containing an HTML link which has a title that must be translated in addition to the text of the sentence. For example:

January 24 is ``National Peanut Butter Day`` in the United States.

XLIFF:doc does not support the subflow `<sub>` element. To handle text that would be considered a subflow, extract it into a separate translation unit. For example, the above sentence might be rendered into 2 translation units as follows:

```
<group id="6">
  <trans-unit id="9">
    <source>January 24 is <g id="18" ctype="link" dx:orig-markup-open="&lt;a
href=&quot;http://www.allyourpeanutbutterarebelongtous.com/index.html&quot; ti-
tle=&quot;Somebody set up us the bomb&quot;&gt;"
dx:orig-markup-close="&lt;/a&gt;">National Peanut Butter Day</g> in the United
States.</source>
    <target state="new"></target>
  </trans-unit>
</group>
<group id="7">
  <trans-unit id="10">
    <source>Somebody set up us the bomb</source>
    <target state="new"></target>
  </trans-unit>
</group>
```

☼ Note: XLIFF:doc does not specify whether the subflow part comes before the TU containing the reference to the subflow. In other words, the contents of TU 9 and TU 10 could have been swapped in the example above.

HTML Previews

While XLIFF:doc does not guarantee that any compliant tool can generate a target from every XLIFF:doc, it does guarantee that any compliant tool can generate an HTML view of the original document (with source and/or target text showing). The fidelity of the preview could range from a simple table view, to an approximation of a document, with headers, bold text, images, etc., to a high-fidelity preview that is functionally and visually identical to the original file format. XLIFF:doc does not put any requirements around how "similar" a preview must be to the original file format. Fidelity will be determined by the complexity of the original file format, and the effort put into creating the preview by the Creation Tool. The preview can be embedded into the XLIFF file, or it can be external to the XLIFF.

The following items give an overview of the preview:

1. The preview will be constructed in HTML format. There are no requirements to use a specific version of HTML. However, cross-platform, cross-browser code is highly recommended (YAML, etc.).
2. The preview can reference images, but only if those images are either external (via URL) or relative to images included in the TIP package.
3. If the preview is placed in an external file, the HTML code does not need to be escaped. If the preview is being embedded in the XLIFF file, the HTML code will be escaped, as in this example:

```
&lt;html&gt;
&lt;head&gt;
&lt;title&gt;Translation Preview&lt;/title&gt;
&lt;/head&gt;
```
4. A `<dx:content>` element will be placed in the preview at any location translatable text has been extracted out into `<trans-unit>` elements. The TU ID of the `<trans-unit>` in question will be specified in the *id* attribute of the `<dx:content>` element, as follows, as in this example for TU ID "2":

```
&lt;div id="siteSub"&gt;<dx:content id="2"/>&lt;/div&gt;
```

Generating an HTML Preview from an XLIFF

The general outline for creating an HTML preview might resemble the following:

1. Obtain contents of the `<dx:internal-preview>` element and unescape it; or obtain the contents of the file referenced by the `<dx:external-preview>` element.
2. Replace `<dx:content>` elements with the contents of the `<target>` element. For inline tags, use the *dx:equiv-markup* and *equiv-text* attributes as appropriate. Do not use original markup unless the original file format was HTML. If no equivalent information is available for a placeholder, consider substituting a generic character or string (eg, "{tag}" for standalone placeholders, and underlining for paired tags. In that way, at least some information will be provided on the position of placeholders.
3. Display the resulting HTML to the user

The details of implementation are not specified by XLIFF:doc. You could use XSLT to accomplish it, or string-based code.

XSL Previews

In addition to HTML previews, XLIFF:doc provides a second, optional preview type based on XSLT. Creator Tools are not required to support this preview mechanism. If a Creator Tool does provide for an XSL preview, it must, however, supply the XSLT needed to generate the preview, and any Processing Tool that opens the XLIFF:doc must be able to generate a preview for the user using that XSL.

The following items give an overview of the XSL preview:

Interoperability

1. The Creator Tool embeds a link to an XSLT file relative to the XLIFF:doc file. This is done using the `<dx:xsl-external-file>` element.
2. The Creator Tool populates the `dx:xsl-hook` attribute in each `<trans-unit>`.
4. To generate a preview, a Processing Tool obtains the XSLT and applies it to the XLIFF:doc file itself, then displays the resulting HTML to the user.

Joining and Splitting Translation Units

XLIFF:doc supports the joining and splitting of translation units (segments). Creation Tools can decide whether or not to allow it for specific packages, source files, and individual translation units. Processing Tools must support joining and splitting, but must also respect the limitations set for a file by the Creation Tool. It is the responsibility of the Processing Tools to ensure that inline formatting tags (<g> and <x> placeholders) are transferred appropriately during split and join operations.

Limiting Joining and Splitting at the File Level

A Creation Tool can allow or disallow splitting and joining for an entire source file by setting the *dx:allow-tu-join* and *dx:allow-tu-split* attributes of the <file> element. Both attributes default to "yes", so they only need to be set if you do not want to allow joining and/or splitting. Processing Tools may not adjust these values, and must enable/disable joining and splitting accordingly to be compliant to XLIFF:doc.

Limiting Joining at the Translation Unit Level

Even if joining of TUs is permitted in general for a file, there may be specific reasons why certain TUs should not be joined. Typically, this would be because there is structural markup between the TUs that would cause the document to lose stability if the TUs were joined. In XLIFF:doc, the <group> element is used only to prevent specific TUs from being joined. The <body> element needs at least 1 <group> element, and all TUs have to be part of a <group>. A TU that is not in the same group as another TU cannot be joined with that TU.

- ☼ Example 1: If you had a purely text file (no markup), you would likely end up with a single <group> for the entire XLIFF. That group would contain all the TUs in the file, and any TU could be joined with the one before or after it.
- ☼ Example 2: If you had a simple XML document with 3 <para> elements, you might have three <group> elements in your XLIFF, one for each <para> element in the original document. That would allow the sentences within the para to be joined with each other, but joining would be prevented across the <para> element.
- ☼ *Note: Add something about where store restype, story ids, etc as used by Rainbow for InDesign docs. Will apply these to the TU object, and not add another grouping level.*

Note: if splitting is permitted for the file as a whole, it is permitted for every TU in the document, and cannot be overridden on a per-TU basis.

Describing Translation State

In the XLIFF 1.2 specification, each translation unit in an XLIFF file can have `<alt-trans>` translation matches, each of which has a preset match-rate described for it. Each project translation unit can also be assigned a "state", such as "new", "translated", etc. There is not, however, a built-in mechanism to describe the quality of the translation unit after a machine translation has been applied, or after an embedded TM match or a live TM lookup match has been applied. If you save an XLIFF after applying a match, no information is saved about where the translation came from, how close a match it was to the source, etc.

XLIFF:doc addresses these shortcomings in two ways: specifying exactly which status labels can be applied to a translation unit, and by adding new attributes to the `<trans-unit>` to enable it to maintain its own state better. Some information is stored on the `<trans-unit>` level, and some is stored on the `<target>` level.

Translation Status

XLIFF:doc uses the `state` attribute of the `<target>` to record the state of a translation. The XLIFF 1.2 specification includes 10 pre-defined states, and allows for unlimited user-defined states. This has meant, in practice, that very few tools and organizations use the same states. Or, if they do use the same states, they often don't use them the same way (to mean the same thing). To help foster interoperability between tools, XLIFF:doc specifies a much smaller set of states: "new", "translated", "final", "signed-off", and "needs-review-translation". Use of other states is not permitted within XLIFF:doc. How individual organizations interpret these statuses is outside the purview of XLIFF:doc, but one possible interpretation is as follows:

Description	XLIFF:doc state	UI Representation
New translations in a new XLIFF	new	New
A complete translation is present in the target	translated	Translated
The translation has been validated for accuracy, context, terminology, etc., according to the procedures and policies of the organization	final	Proofed
The final authority on the target language (a client's subject matter expert, for example) has reviewed the translation and approved it.	signed-off	Validated
An expert or other authority has reviewed the translation and rejected the proposed translation. This TU must be reviewed and modified as necessary.	needs-review-translation	Rejected

Addition Notes on Status Usage

- XLIFF:doc does not require any particular usage of the statuses provided. The examples provided in this document are just that: examples. However, statuses are important to passing projects smoothly between different users and groups, and so, to encourage more interoperability, XLIFF:doc limits the available statuses to the ones listed above.
- Depending on process, translations from a validated TM source that are placed during pre-translation by the Creation Tool may be set to "final" or "signed-off" from the start.
- Depending on process, setting the status of "translated" may play a role similar to "confirming" a translation in tools that support an on/off state for confirmation.
- The reviewer does not necessarily have to add a note when they mark a translation as Rejected (needs-review-translation), but in an ideal world, perhaps they would. This is not something XLIFF:doc specifies.
- The usage of the "final" and "signed-off" states may vary from situation to situation and organization to organization.
- For an organization with a simple Translation-Edit-Proof (TEP) process, The "Translate" step may end with all TUs marked "translated", the "Edit" step with all TUs marked "final", and the Proof step with all TUs marked as "signed-off".

Interoperability

- For an organization doing TEP followed by a client review, the Translation stage may end at "new", the Edit phase at "translated", the Proof step at "final", and the client's review phase at either "needs-review-translation" or "signed-off". Once all rejected TUs had been successfully reviewed and review comments were incorporated, all TUs might then be marked "signed-off".
- What "final" means to an organization will vary, but the basic idea should be the same: this translation is certified as accurate and correct for the context it is in, according to the procedures of the organization handling the translation.

Recording Match Quality After a Match has been Applied

If a translation has been applied to the TU, and the user has not subsequently edited the `<target>`, then information about that match can be recorded against the `<trans-unit>`. This allows users to identify how a translation was populated, whether by human editing, or from a live TM, live machine translation engine, embedded XLIFF match, etc. Use the `dx:match-quality`, `dx:glorious-match`, `dx:origin`, `dx:origin-shorttext`, and `dx:match-penalty` to imprint the TU with the status of the match applied to it.

☀ *Note: if a translator edits a translation, the match-related information must be changed, as it no longer accurately reflects the match that was applied. `dx:glorious-match`, `dx:match-quality`, and `dx:match-penalty` must be removed. `dx:origin` must be set to "manual" and `dx:origin-shorttext` to "Manual Translation".*

Last Edited Information

Use `dx:modified-by` and `dx:modified-at` to show when the TU was last edited, and who edited it. When a Creation Tool places translations ("pre-translation") in an XLIFF:doc, it must either specify the user who last edited the translation in the TM, or a general identifier (eg, "Pre-Translation", "Acme TM Server", etc.). This information will be used by the revision history functionality.

Usage Example: Translation Attributes and Workflows

XLIFF:doc does not define translation workflow. This example is provided merely as an example of how actions in real world workflows might be reflected in XLIFF:doc.

☀ *Note: The UI for a given tool can choose to represent the permitted translations states in whatever manner they want. In this example, "needs-review-translation" is shown as "Rejected", and "signed-off" is shown as "Validated".*

Step #	Workflow Step	XLIFF:doc Representation
1	TM server pre-translates file, finding exact, in-context matches for 25% of the TUs. The matches are coming out of a TM where all translations have been reviewed and validated.	<code><target></code> element is populated with translation from TM. <code>target/state</code> is set to "translated" <code>trans-unit/dx:match-quality</code> is set to "100" <code>trans-unit/dx:glorious-match</code> is set to "yes"
2	TM server pre-translates file, finding exact matches for 50% of the TUs.	<code><target></code> element is populated with translation from TM. <code>target/state</code> is set to "translated" <code>trans-unit/dx:match-quality</code> is set to "100"

Interoperability

Step #	Workflow Step	XLIFF:doc Representation
3	TM server finds no matches or only fuzzy matches for the remaining 25% of the TUs.	<target> element is left empty. target/state is set to "new" <alt-trans> elements are added with fuzzy matches
4	Translator reviews the matches listed as "Translated", and finds some that need to be modified to match context, subject matter, etc. Translator makes appropriate changes. Since these are already set to "Translated", Translator doesn't have to change their status when she edits them.	target/state stays at "translated". trans-unit/dx:match-quality is removed (it is now a manual translation).
5	Translator sees that 25% of the translations are in-context exact matches already marked as "Translated", and so skips past these translations.	n/a
6	Translator filters to "New" translations, then for each one, manually translates the source, or transfers fuzzy matches to the target, and modifies as needed. Translator sets status to "Translated"	target/state is set to "translated" trans-unit/dx:match-quality is removed (it is now a manual translation).
7	Editor receives file and confirms that all TUs are set to "Translated"	[all TUs have already been set to target/state="translated"]
8	For each translation he disagrees with, the Editor changes the translation as appropriate. Since the Editor is not planning on sending these back to the first linguist for approval, the Editor sets the status of these translations as "Proofed".	target/state is set to "final" If the the translation edited was previously a 100%, ICE, or fuzzy match, the trans-unit/dx:match-quality is removed (it is now a manual translation).
9	For each translation he agrees with, the Editor changes status to "Proofed".	target/state is set to "final"
10	Reviewer receives file and confirms that all TUs are set to "Proofed"	[all TUs have already been set to target/state="final"]
11	For each translation he disagrees with, the Reviewer adds a comment, and changes status to "Rejected".	target/state is set to "needs-review-translation" and a <note> element is added to the <trans-unit> element.
12	For each translation he agrees with, the Reviewer changes status to "Validated".	target/state is set to "signed-off"
13	File is returned to Translator, who checks all the TUs marked as Rejected. If she agrees with them all, she incorporates the suggestions found in the comments, and marks the TUs as "Validated". Any comment suggestions that she is unable to incorporate, she leaves the TU status as "Rejected", but adds a note to explain why the change would not be appropriate. Steps 11-13 then repeat until all differences have been resolved and all TUs have been marked as "Validated".	target/state is set to "signed-off" or a <note> element is added to the <trans-unit> element.

Inline Formatting

In order to simplify the handling of inline formatting, all inline formatting in XLIFF:doc is handled via the `<g>` and `<x>` tags. The other elements provided by the full XLIFF 1.2 specification are not supported by XLIFF:doc. The original formatting must be included in the XLIFF. In addition, the *equiv-text* attribute must be specified when working with `<x>` tags.

Interpreting Tag Content for Users

XLIFF:doc can be used to represent content from hundreds of content creation systems, including not just commercial applications like MS Word, Adobe InDesign, Adobe FrameMaker, but also HTML and XML-based CMS systems. Obviously, inline formatting can vary widely from system to system. To help linguists correctly place tags when working in XLIFF:doc, it is important that XLIFF:doc Creation Tools provide as much information as possible about the inline formatting captured in `<g>` and `<x>` elements. In XLIFF:doc, this is accomplished by using the *ctype* attribute. Use the list of pre-defined values for *ctype* provided in the Reference Section.

☀ *Note: Although XLIFF:doc does not specify exactly how to map original formatting code to the predefined list of *ctype* values, the more effort a Creation Tool makes to map detailed formatting codes to easy-to-understand *ctype* values, the better able Processing Tools will be able to represent the document to users, and the better able the users will be to place the tags correctly. As an example, the following tags might all be described as "bold" in XLIFF:doc: RTF {b}, MS Word <w:b>, HTML , HTML , Custom XML <Bold>, Custom XML <TotallyCritical>.*

Providing Text Representations of Standalone Tags

In order to provide linguists with the information they need to evaluate a segment in XLIFF:doc, use the *equiv-text* attribute with your `<x>` tags. The idea behind *equiv-text* is to provide at least a rough approximation of how placeholders will be resolved in the final system. For example, if the `<x>` tag encapsulates an XML entity that a CMS style sheet will resolve to a unit of measurement in the final document, you might populate the *equiv-text* value with that unit of measurement. If the `<x>` tag represents a line-break, you might choose to insert a line-break symbol (or a space, etc.). If the `<x>` tag represents a non-breaking space in the source system, you might use a space character in *equiv-text*.

equiv-text Mapping Examples

Encapsulated Content	Meaning in Source System	equiv-text Value
&sys-volts;	Unit of Measurement: Volt	V
\n	Line break	¶
&sys-product-name;	Name of product	ProductName
<company-name type="non-local"/>	Name of company	XYZ Corporation, Inc.
<w:page-num/>	Current page number	#

☀ *Note: XLIFF:doc does not specify how you populate the *equiv-text* attribute, only that you do it. The above examples are not meant to be guidelines.*

equiv-text Readability Examples

Simplified Tags	No equiv-text Provided	With equiv-text
Ensure that the defibrillation lead impedance is greater than 20{1}.	Ensure that the defibrillation lead impedance is greater than 20.	Ensure that the defibrillation lead impedance is greater than 20 Ω.
69{1} x 51{2} x 15{3}	69 x 51 x 15	69 mm x 51 mm x 15 mm

Interoperability

Simplified Tags	No equiv-text Provided	With equiv-text
The Model {1} {2} device and the pacing leads and defibrillation leads constitute the implantable portion of the device system.	The Model device and the pacing leads and defibrillation leads constitute the implantable portion of the device system.	The Model Acme SuperDefibrillator device and the pacing leads and defibrillation leads constitute the implantable portion of the device system.
For a list of the features that are enabled at shipping, see the "Shipped" column of the tables in {1}.	For a list of the features that are enabled at shipping, see the "Shipped" column of the tables in .	For a list of the features that are enabled at shipping, see the "Shipped" column of the tables in {XREF}.
(02)-123-4567{1}Fax (02)-123-5678	(02)-123-4567Fax (02)-123-5678	(02)-123-4567 Fax (02)-123-5678

Providing HTML Representations of Tags

To give Processing Tools the ability to show formatted XLIFF:doc content to users, XLIFF:doc uses the *dx:equiv-markup*, *dx:equiv-markup-open*, and *dx:equiv-markup-close* attributes. Populate these attributes with HTML code that approximates the formatting of the original document format.

☼ *Note: if the original format is HTML, and the HTML formatting can be represented safely, it is not necessary to include *dx:equiv-markup* attributes. If not present, the *dx:orig-markup* will be used. See the section on *ctype* in the reference section for more information.*

See the table below for examples:

Original Document Formatting	XLIFF:doc Example
HTML: This is important .	This is <code><g id="1" ctype="bold" dx:orig-markup-open="&lt;em&gt;" dx:orig-markup-close="&lt;/em&gt;">important</g></code> . Might be rendered as: This is important .
XML: This is important .	This is <code><g id="1" ctype="bold" dx:orig-markup-open="&lt;emphasis&gt;" dx:orig-markup-close="&lt;/emphasis&gt;" dx:equiv-markup-open="&lt;em&gt;" dx:equiv-markup-close="&lt;/em&gt;">important</g></code> . Might be rendered as: This is important .
Docx: <code><w:r><w:t>This is</w:t></w:r><w:r><w:rpr><w:b /></w:rpr><w:t>Important</w:t></w:r></code>	This is <code><g id="1" ctype="x-other" dx:orig-markup-open="&lt;&lt;w:r&gt;&lt;w:t&gt;&gt;" dx:orig-markup-close="&lt;/w:t&gt;&lt;/w:r&gt;" dx:equiv-markup-open="&lt;" dx:equiv-markup-close="&gt;"><g id="2" ctype="bold" dx:orig-markup-open="&lt;w:r&gt;&lt;w:rpr&gt;&lt;w:b /&gt;&lt;/w:rpr&gt;&lt;w:t&gt;" dx:orig-markup-close="&lt;/w:t&gt;&lt;/w:r&gt;" dx:equiv-markup-open="&lt;b&gt;" dx:equiv-markup-close="&lt;/b&gt;">important</g></code> Might be rendered as: This is important


Notes

In XLIFF:doc, users can apply notes to two distinct parts of the XLIFF:

- a specific translation unit (either a current `<trans-unit>` or a leveraged `<alt-trans>`)
- the entire file

 *Note: Notes that refer to the file as a whole are entered against the `<header>` element.*

Notes are stored with the name or description of the user/system that inserted or last modified the note, and the last date/time associated with the note. XLIFF:doc uses the `<note>` element to store the actual note text, and the *from* attribute to define the "who" and "when" information.

 *Note: Due to a limitation in the XLIFF 1.2 specification, we are unable to add any additional attributes to the `<note>` element.*

Notes that refer to a specific TU can optionally refer to either the source or target. Unless specified otherwise, notes refer to the TU as a whole.

XLIFF:doc does not support notes tied to other XLIFF elements.

Translation Matches

When translating an XLIFF:doc file, translation memory matches can come from two sources:

1. A "live" TM: a TM available from a networked server, or a file-based TM the translators has on their desktop.
2. Embedded matches: TM matches pre-placed in the XLIFF:doc by a CAT tool.

☀ *Note: The use of live TMs to find translation matches is outside of the scope of XLIFF:doc, and is defined by the Processing Tool.*

Under XLIFF:doc, TM matches are embedded per-TU, using the `<alt-trans>` element. Any embedded matches will be available to any translator editing the XLIFF:doc using a compatible tool, regardless of whether or not they have access to a TM server or database file.

Rules for TM Matches in `<alt-trans>`

XLIFF:doc establishes several rules for embedded TM matches, and allows a Creation Tool to set other rules. These are the fixed rules which must be followed by any XLIFF:doc compliant tool:

- Only the Creation Tool can set the `dx:allow-tu-match-deletion` and `dx:allow-tu-match-addition` attributes
- Processing Tools must retain any TM matches embedded in an XLIFF:doc by the Creation Tool unless the `dx:allow-tu-match-deletion` attribute is set to "yes". If an XLIFF:doc file has 500 TM matches in it when a tool opens it, the tool must write it out with those same 500 matches.
- Processing Tools must not add TM matches to the XLIFF:doc, unless the `dx:allow-tu-match-addition` attribute is set to "yes". Processing Tools can always show users matches from other sources, but it cannot pass these matches along to subsequent tools unless permitted to do so by the Creation Tool.
- Processing Tools must display all embedded TM matches to users when the user requests them.
- Processing Tools must maintain the relationship between an embedded TM match, and the TU that it was originally assigned to. If a Processing Tool reads an XLIFF:doc into some other bilingual format native to the tool, and in doing so adds the embedded TM matches to a database, then the tool must retain the link between individual TM entries and the XLIFF TUs the Creation Tool embedded them into. The tool must display the formerly embedded matches relevant to the TUs they were originally embedded in, whenever the user requests to see matches for the TU. See the next rule.
- Processing Tools must display embedded TM matches associated with a particular TU, regardless of the user's live TM settings. For example, if a user has chosen to only display TM matches of 85% or higher, the tool must still display a 50% match to the user, if a 50% match was embedded for the TU in question. The reason for this is to honor the settings of the original TM server (Creation Tool) that placed the matches. However, a tool may have a user-controlled switch that toggles the display of embedded matches on and off.

☀ *Note: these limits apply only to TM matches in alt-trans. They do not apply to alt-trans used for revision history purposes.*

Describing Match Quality

The `match-quality` attribute is used to indicate how similar a translation candidate's source text is to the XLIFF TU's source text. Use values 0-100 (decimal numbers are acceptable).

☀ *Note: match-quality and dx:match-quality refer solely to the similarity of the source text to the match's source text.*

To indicate an in-context, exact match, use a `dx:match-quality` of 100, and set `dx:glorious-match` to "yes".

Interoperability

If the Creation Tool assigns an artificial penalty to translation matches, the amount of that penalty can be recorded in the `dx:match-penalty` attribute. Do not specify textual difference penalties here: this value is used to describe penalties associated with the match's origin or attributes.

- Example 1: An exact match is found in the TM, and the context matches. The match is from the same product line and business unit. The `match-quality` is set to 100%, `dx:glorious-match` is set to "yes", and `dx:match-penalty` is set to 0%. The CAT tool displays this match to the user as a 100% match.
- Example 2: An exact match is found in the TM, but the match is from a different product line and context. Additionally, there are placeholders in the TM match, but none in the XLIFF. The business unit is the same. The `match-quality` is set to 99.5%, `dx:glorious-match` is set to "no", and `dx:match-penalty` is set to 2%. The CAT tool displays this match to the user as a 97.5% match.
- Example 3: No match is found for a sentence in the TM, so a machine translation is applied. The "source" for the machine translation is an exact match, textually, to the source in the XLIFF. The `match-quality` is set to 100%, `dx:glorious-match` is set to "no", and `dx:match-penalty` is set to 10%. The CAT tool displays this match to the user as a 90% match.
- Note: Processing Tools can choose to combine this information to display an overall match rate (100% text match - 3% attribute mismatch penalty = 97% overall match), or they can display it separately. This behavior is not dictated by XLIFF:doc.

Use the `origin` and `dx:origin-shorttext` attributes to describe where the match candidate came from.

Use the XLIFF `<alt-trans><target>` element's `state` attribute to indicate the status of the proposed match (if one was assigned to it in the originating TM, etc.). See the reference section for a list of the acceptable values for this attribute.

Describing Attributes of the Match

Use the XLIFF:doc elements `<dx:attributes>` and `<dx:attribute>` to provide meta information to the user and Processing Tool about the match candidates. This could include user definable pairs (eg, "Product", "Company", "Project", etc.), as well as system-defined information such as the username of the person who last edited the translation, timestamp for the last edit, etc.

Usage Example


```
<alt-trans alttranstype="proposal" state="signed-off" match-quality="100" origin="TM"
dx:origin-shorttext="Master TM 1" dx:modified-by="smithj1" dx:modified-at="20040805T00:12:34Z">
  <source>Risks normally associated with the implantation of a pacemaker include:</source>
  <target>Les risques normalement associés à l'implantation d'un stimulateur cardiaque
incluent :</target>
  <note from="Jane Doe" dx:modified-at="20101012T00:00:00Z">This is a really old translation!</
note>
  <dx:attributes>
    <dx:attribute name="Company">Medtronic, Inc.</dx:attribute>
    <dx:attribute name="Business Unit">CRDM</dx:attribute>
    <dx:attribute name="Style">Clinician Manual</dx:attribute>
    <dx:attribute name="Product">Lead 4195,Lead 4296,Lead 4396</dx:attribute>
    <dx:attribute name="Project">4195 Tech Man r000 9/17/08,4296 Tech Man RA rev 2009-09-03,4396
Tech Man RA 2009-11-05</dx:attribute>
    <dx:attribute name="Project ID">GTSID001990,GTSID002968,GTSID003131</dx:attribute>
  </dx:attributes>
</alt-trans>

<alt-trans alttranstype="proposal" match-quality="100" origin="MT" dx:match-penalty="15"
dx:origin-shorttext="MT Engine 2" dx:modified-by="mtengine2" dx:modified-at="20101013T00:00:00Z">
  <source>Risks normally associated with the implantation of a pacemaker include:</source>
  <target>Les risques normalement associés à l'implantation d'un stimulateur cardiaque
comprennent:</target>
</alt-trans>
```


Terminology Matches

XLIFF:doc supports the ability for any tool in the process to embed terminology data specific to the project. In practice, this would likely be done by the Creation Tool.

XLIFF:doc supports the concept of TU-specific terminology. In contrast to the traditional model of simply providing a termbase and expecting the translation editor application to find the appropriate matches from that termbase, XLIFF:doc pre-embeds the appropriate term matches for each TU. As a result, it does not matter how the translation editor application analyzes and matches terminology against a source string, the pre-defined terms will always be shown. In fact, the translation editor does not have to know anything about stemming terms, or how to read a TBX file. All it needs to do is read the IDs from the pre-embedded term candidates for each TU, look the IDs up against the included termbase, and show the hits to the end user. This not only makes it easier for tools developers to support terminology in translation editing tools, but it gives more control to the owner of the terminology: the “right” terms will always be shown. In theory, terminology can be embedded by any tool in the project workflow. In practice, it is most likely going to be the job of the XLIFF Creation Tool.

 *Note: The ability to embed terminology does not mean a translation tool cannot also take advantage of a general termbase. The Translation Interoperability Protocol (TIP) Package provides for TBX files to be included in the translation package, alongside the XLIFF. While compatible tools are required to process and display any embedded term matches present within the XLIFF, there is no requirement that matches actually be embedded by the Creation Tool. Additionally, a tool could display term matches from multiple sources (a term server, a TBX, etc.), not just the embedded terms.*

Term Embedding Mechanism

XLIFF:doc uses a glossary and a look-up model to embed terminology data. A UTX-format glossary consisting only of terms referenced in the XLIFF:doc is embedded into a `<dx:utx-glossary>` element. Each TU with a relevant term candidate has a `<dx:term-hits>` element, which contains one or more `<dx:term-hit>` elements.

Embedding Term Matches

To embed terminology data in an XLIFF:doc file, the general process might be something like this:


Read terminology source (a termbase file, data from a terminology server, a csv file, etc.).

- Apply filter(s) as necessary
- Match terminology against each TU in the XLIFF, applying filter(s) as necessary/available.
- For each match found, embed one `<dx:term-hit>` element in the `<trans-unit>` for the TU in question, and include a line in the glossary for that term.

 *Note: Do not include the same term more than once in the glossary!*

- Insert the data from UTX format glossary you have compiled from the term candidates into the `<dx:utx-glossary>` element (under the `<xliff>` root element).

If the termbase is concept-based, and you want to make synonyms within a matched concept available to the end user, enter each synonym in a different `<dx:term-hit>` element.

 *Note: It is the responsibility of the Creation Tool to pick the best terminology match. If there are synonyms, it is the responsibility of the Creation Tool to do the "right thing". That may mean using the meta information available to it to pick the most appropriate synonym, or it may mean including all synonyms. This behavior is not dictated by XLIFF:doc. XLIFF:doc does require all Processing Tools to make all term matches embedded in the file available to users.*

Interoperability

If there are no term matches for a given TU, do not include a `<dx:term-hits>` element in that TU.

Mapping TBX to UTX

XLIFF:doc uses the [Universal Terminology Exchange \(UTX\)](#) format for embedded terms. UTX is a simple tab-delineated format that provides easy access to basic terminology information. Before the UTX can be embedded in XLIFF:doc, it needs to be mapped to XLIFF:doc xml elements. The table below will help you map between TBX, UTX, and XLIFF:doc UTX elements.

TBX	UTX before XLIFF:doc conversion	<dx:utx-glossary>
termEntry	[a line in the UTX glossary]	<dx:utx-term>
termEntry@id	"concept ID" column	dx:utx-term@concept-id
termEntry/langSet	Use to map the termEntry/langSet/tig/term to either the "src" or "tgt" column	dx:utx-term@src-lang and dx:utx-term@tgt-lang
termEntry/langSet/tig/term	"src" or "tgt" column	dx:utx-src or dx:utx-tgt
termEntry/langSet/tig/termNote	"comment" column	dx:utx-comment
termEntry/langSet/tig/termNote@type	n/a	n/a
termEntry/langSet/tig/termDescrip	n/a (or map to "comment" column in place of termNote)	n/a or map to dx:utx-comment
termEntry/langSet/tig/termDescrip@type	n/a	n/a

UTX as Supported by XLIFF:doc

In order to provide interoperability, XLIFF:doc defines the following requirements for embedded UTX glossaries:

- The UTX must be mapped to the `<dx:utx-glossary>` element and its children, as defined in XLIFF:doc. It is not valid to include a UTX as a block of tab-separated text.
- XLIFF:doc requires an 8 column UTX: src, tgt, src:pos, term status, comment, concept ID, entry ID.
- The "entry ID" column value must be unique within the UTX file. This is the value that will be used in the `<dx:term-hit>` *id* attribute, to generate a reference from the TU to the glossary.
- The value of the "term status" column must match the status of the translation entry, not the source entry.
- XLIFF:doc does not specify what a tool must do with the src:pos, comment, or concept ID values, other than make them available to the user.

☀ *Note: One possible use for the concept ID value is to refer back to a TBX glossary included as reference in a TIP package.*

Usage Example

```
<trans-unit id="10">
  <source>The Commodore 64 is the best-selling single personal computer model of all time. </
source>
  <target state="new"></target>
  <dx:term-hits>
    <dx:term-hit id="0"/>
    <dx:term-hit id="1"/>
  </dx:term-hits>
</trans-unit>
<trans-unit id="11">
  <source>The C-64 or C=64 as it was known to fans had impressive graphics at an affordable
price. </source>
  <target state="new"></target>
  <dx:term-hits>
    <dx:term-hit id="0"/>
    <dx:term-hit id="1"/>
  </dx:term-hits>
</trans-unit>
```


Interoperability

```
</dx:term-hits>
</trans-unit>
<trans-unit id="12">
  <source>Its successor, the Commodore Amiga, amazed the computing world when it was launched,
  but was never able to capture the market share enjoyed by the Commodore 64. </source>
  <target state="new"></target>
  <dx:term-hits>
    <dx:term-hit id="0"/>
    <dx:term-hit id="1"/>
    <dx:term-hit id="2"/>
    <dx:term-hit id="3"/>
  </dx:term-hits>
</trans-unit>
```

... (dx:utx-glossary is a child of the <xliff> element.) ...

```
<dx:utx-glossary version="1.11" src-lang="en-US" tgt-lang="ja-JP"
date-created="2011-08-17T19:00:00Z" creator="Acme Corporation">
  <dx:utx-term entry-id="0" concept-id="150">
    <dx:utx-src pos="noun">Commodore 64</dx:utx-src>
    <dx:utx-tgt status="approved">コモドール64</dx:utx-tgt>
    <dx:utx-comment>The Commodore 64 is an 8-bit home computer introduced by Commodore Interna-
    tional in January 1982.</dx:utx-comment>
  </dx:utx-term>
  <dx:utx-term entry-id="1" concept-id="150">
    <dx:utx-src pos="noun">C=64</dx:utx-src>
    <dx:utx-tgt status="forbidden">C=64</dx:utx-tgt>
    <dx:utx-comment>The C=64 had a terrible keyboard (unless, of course you compare it to an IBM
    PCjr keyboard).</dx:utx-comment>
  </dx:utx-term>
  <dx:utx-term entry-id="2" concept-id="172">
    <dx:utx-src pos="noun">Amiga</dx:utx-src>
    <dx:utx-tgt status="approved">アミーガ</dx:utx-tgt>
    <dx:utx-comment>The Amiga is a family of personal computers sold by Commodore in the 1980s
    and 1990s. The first model was launched in 1985 as a high-end home computer and became popular
    for its impressive graphical, audio and multi-tasking abilities.</dx:utx-comment>
  </dx:utx-term>
  <dx:utx-term entry-id="3" concept-id="172">
    <dx:utx-src pos="noun">Amiga</dx:utx-src>
    <dx:utx-tgt status="non-standard">Amiga</dx:utx-tgt>
    <dx:utx-comment>The Amiga is a family of personal computers sold by Commodore in the 1980s
    and 1990s. The first model was launched in 1985 as a high-end home computer and became popular
    for its impressive graphical, audio and multi-tasking abilities.</dx:utx-comment>
  </dx:utx-term>
</dx:utx-glossary>
```

Reading and Displaying Embedded Term Matches

XLIFF:doc does not require any particular behavior on the part of the tools reading the terms, other than to make them available to the user.

QA Information

XLIFF:doc allows for any tool in the chain to add and remove QA messages to the XLIFF file. QA messages are applied to the `<trans-unit>` element as a whole, because a QA issue often involves both the source and the target. Each TU can have at most one `<dx:qa-hits>` element, but each `<dx:qa-hits>` element can have as many `<dx:qa-hit>` elements as necessary.

☼ *Note: XLIFF:doc does not specify the logic a Creation Tool uses to identify errors and generate QA messages, or what a Processing Tool does with the QA messages.*

If the QA hit can be traced to a particular spot in the source or target (or both), the position of the first character of the affected text, and the length of the affected text, can be placed in these attributes to provide the Processing Tool an opportunity to highlight, link, or otherwise indicate to the user the text related to the QA hit. The `dx:qa-tgt-startpos` (or `dx:qa-src-startpos`) is the position from the start of the segment, with 1 as the first position. The `dx:qa-tgt-length` is the length in characters of the affected text. If the QA hit is related to text in both the source and text (`dx:qa-origin="both"`), then both pairs of attributes must be filled out. Both the `dx:qa-tgt-startpos` and `dx:qa-tgt-length` attributes must be filled out. The Processing Tool must not attempt to visually highlight the QA hit text, if either or both of these attributes are set to "0".

Beyond this, XLIFF:doc does not specify what a Processing Tool should do with this information.

☼ *Note: `dx:qa-tgt-startpos` and `dx:qa-tgt-length` are both 1-based integers, and both refer to Unicode characters, not necessarily bytes.*

☼ *Note: Processing Tools are not to perform any Unicode normalization on text. Normalization can result in invalidating the position markers from tool to tool. See [Unicode Standard Annex #15: Unicode Normalization Forms](#) for more information.*

☼ *Note: when specifying start position and length, you must account for tags when setting the start position and length: add 1 character for each tag. See the second example below.*

☼ *Note: if the translation is modified by a tool or by a user at any point after the QA information is embedded, the QA information may no longer be accurate. The position of the text referred to by the QA hit may have moved, the text may have been deleted, etc. QA hits are less permanent in nature than TM and terminology matches, and Processing Tools should take that into consideration when processing and displaying them.*

If the XLIFF:doc flags set by the Creation Tool allow it, tools must give users the ability to remove QA hits the user no longer wishes to see. See the Reference section for more information.

Usage Example 1

Source: The pump internal volume (internal tubing volume) for all models of the pump is 300 µL.

Target: Het interne pompslangvolume bedraagt voor alle pompmodellen 300 ml.

Resulting Markup:

```
<dx:qa-hits>
  <dx:qa-hit dx:qa-origin="source" dx:qa-code="005" dx:qa-ignorable="no" dx:qa-shorttext="Number
  found in source, not found in target: &quot;300 µl&quot;" dx:qa-category="number check"
  dx:qa-level="error" dx:qa-src-startpos="81" dx:qa-src-length="6" />
  <dx:qa-hit dx:qa-origin="target" dx:qa-code="005" dx:qa-ignorable="no" dx:qa-shorttext="Number
  found in target, not found in source: &quot;300 ml&quot;" dx:qa-category="number check"
  dx:qa-level="error" dx:qa-tgt-startpos="61" dx:qa-tgt-length="6" />
</dx:qa-hits>
```

☼ *Note: the list of acceptable values for the `dx:qa-category` attribute can be found in the Reference Section*

Usage Example 2

Source: `<g id="1" ctype="x-xml-element" dx:orig-markup-open="<acronym>" dx:orig-markup-close="</acronym>">IEC</g> 601-1-2.`

Target: `<g id="1" ctype="x-xml-element" dx:orig-markup-open="<acronym>" dx:orig-markup-close="</acronym>">IEC</g> 601-1-1`

Resulting Markup:

```
<dx:qa-hits>
  <dx:qa-hit dx:qa-origin="source" dx:qa-code="025" dx:qa-ignorable="no" dx:qa-shorttext="IEC
    Standard found in source, not found in target: &quot;IEC 601-1-2&quot;;"
    dx:qa-category="pattern" dx:qa-level="error" dx:qa-src-startpos="2" dx:qa-src-length="13" />
  <dx:qa-hit dx:qa-origin="target" dx:qa-code="025" dx:qa-ignorable="no" dx:qa-shorttext="IEC
    Standard found in target, not found in source: &quot;IEC 601-1-1&quot;;"
    dx:qa-category="pattern" dx:qa-level="error" dx:qa-tgt-startpos="2" dx:qa-src-length="12" />
  <dx:qa-hit dx:qa-origin="source" dx:qa-code="031" dx:qa-ignorable="no" dx:qa-shorttext="End of
    sentence punctuation in source different from target: &quot;.&quot;;"
    dx:qa-category="punctuation" dx:qa-level="warning" dx:qa-src-startpos="12"
    dx:qa-src-length="1" />
  <dx:qa-hit dx:qa-origin="source" dx:qa-code="029" dx:qa-ignorable="yes" dx:qa-shorttext="Two
    or more continuous spaces in source" dx:qa-category="punctuation" dx:qa-level="warning"
    dx:qa-src-startpos="4" dx:qa-src-length="2" />
</dx:qa-hits>
```

Revision History

Tool developers may choose to include revision history for each TU in the XLIFF. This would allow linguists, project managers, etc., to review the history of a given translation unit as it travels through the translation workflow. Revision history is only captured for the target, since the source is immutable in XLIFF:doc.

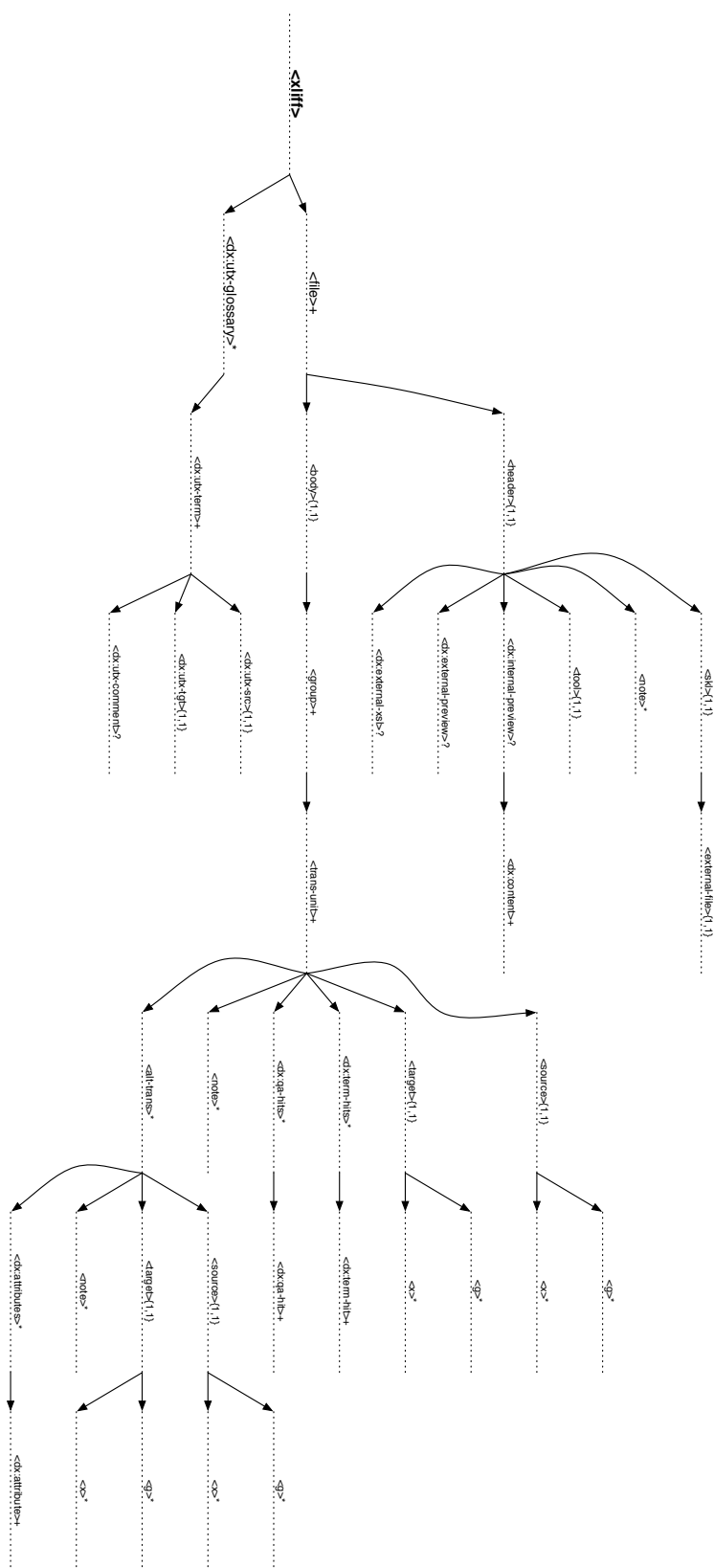
To record translation history, use the `<alt-trans>` element, with the `alttranstype` attribute set to "previous-version". Specify the user who created/edited the previous translation, using the `dx:modified-by` attribute. Specify the date and time the previous translation was last modified (created, edited, etc.), using the `dx:modified-at` attribute.

☀ *Note: `match-quality` must not be specified when using `<alt-trans>` to record revision history. `match-quality` for a previous translation is always functionally equivalent to `100 + dx:glorious-match="yes"` (exact, in-context match).*

Usage Example

```
<alt-trans alttranstype="previous-version" dx:modified-by="JaneDoe"
dx:modified-at="20101013T21:06:00Z">
  <target>Ja</target>
</alt-trans>
```

Appendix A: Diagram of XLIFF:doc Elements



Appendix B: Reference Guide to XLIFF:doc

- ☼ *Note: all elements mentioned here must be processed and acted upon by all Creation Tools and Processing Tools. All required attributes must be processed. All optional attributes must be at least passed through without loss, if the tool is not going to act on the attribute. Any element or attribute not specified here must not be used.*
- ☼ *Note: By necessity, this reference includes information on the XLIFF namespace. This is not meant to be a reference to the entire XLIFF namespace, only the portion used by XLIFF:doc. Please refer to the [XLIFF 1.2 specifications](#) for more information.*
- ☼ *Note: this reference section uses regex-inspired conventions to denote how many child elements a particular XML element should have. The following examples may help:*
 - `{1,1}` a single child element is required
 - `{1,3}` at least one child element is required, no more than 3
 - `+` at least one child element is required (no upper limit)
 - `?` optionally, one child element
 - `*` optionally, one or more child elements

<alt-trans>

In XLIFF:doc, alt-trans is used for 2 purposes. Requirements are different depending on usage:

1. Embedded translation matches from TM, MT, etc. Use *alttranstype* = "proposal".

☼ *Note: Required attributes for this usage: match-quality, dx:modified-by, dx:modified-at*

☼ *Example: <alt-trans alttranstype="proposal" match-quality="99.75" dx:match-penalty="0.25" origin="TM" dx:origin-shorttext="Master TM 1" dx:modified-by="smithj1" dx:modified-at="20101012T00:00:00Z">...</alt-trans>*

2. Translation revision history. Use *alttranstype* = "previous-version". Do not specify *match-quality*, *origin*, or *dx:origin-shorttext*. Do not include a <source> element.

☼ *Note: Required attributes for this usage: alttranstype, dx:modified-by, dx:modified-at*

☼ *Example: <alt-trans alttranstype="previous-version" dx:modified-by="JaneDoe" dx:modified-at="20101013T21:06:00Z">...</alt-trans>*

Parent Elements: <trans-unit>

Child Elements: <source>{1,1}, <target>{1,1}, <note>*, <dx:attributes>*

Required Attributes: *alttranstype*, *match-quality*, *dx:glorious-match*, *dx:match-penalty*, *origin*, *dx:origin-shorttext*, *dx:modified-by*, *dx:modified-at*

alttranstype

Acceptable values: "proposal", "previous-version"

match-quality

Use values 0-100. The use of decimal numbers is acceptable, but only a single digit of precision is allowed. For example, "99.5" is acceptable, but "99.52" is not. Do not use a percent sign. Higher numbers refer to better matches than lower numbers.

dx:glorious-match

Interoperability

Indicates whether the match is an exact, in-context match (as determined by the tool that generated the match). Acceptable values: "yes", "no". Default value: "no".

origin

Acceptable values: "TM", "MT", "XLIFF", "manual", "sourcetext", "pseudo-trans". See the *dx:origin* attribute of the `<trans-unit>` element for more information on how to assign and interpret these values.

dx:origin-shorttext

Human-readable string describing the match origin. This could be the name of the TM it came out of, the name of an MT engine, etc. Examples: "Master TM 1", "Google MT", "myDocument.docx.xlf".

dx:match-penalty

Use values 0-100. The use of decimal numbers is acceptable, but only a single digit of precision is allowed. For example, "0.5" is acceptable, but "0.48" is not. Do not use a percent sign. Higher numbers refer to larger penalties than lower numbers.

dx:modified-by

String value identifying the user (human or automatic) responsible for the most recent modification.

dx:modified-at

Time and date the translation was last modified. This can represent the date/time it was created, editing, or processed in some other way.

☀ *Note: Date/time must be specified in ISO 8601 format. Time must be specified in UTC (Zulu) time zone. [See the XLIFF date element for more information.](#)*

Optional Attributes: none

`<body>`

Parent Elements: `<file>`

Child Elements: `<group>+`

Required Attributes: none

Optional Attributes: none

`<dx:attribute>`

Provides key-value style meta information about the parent element. Use the *dx:name* element for the key, and place the value into the text child.

Parent Elements: `<dx:attributes>`

Child Elements: none

Required Attributes: *dx:name*

name

The "key" part of the attribute key/value pair. May be used by a Processing Tool to display the attribute to the user.

`<dx:attributes>`

```
<dx:attribute dx:name="Company">Acme, Inc.</dx:attribute>
<dx:attribute dx:name="Business Unit">Advanced Rockets and Anvils</dx:attribute>
</dx:attributes>
```

Optional Attributes: none

Interoperability

<dx:attributes>

Container for holding individual attributes (<dx:attribute>) associated with the parent object.

Parent Elements: <alt-trans>

Child Elements: <dx:attribute>+

Required Attributes: none

Optional Attributes: none

<dx:content>

A child-less element that marks the place in a preview element or file where a translation unit's source or target should be placed to create a preview.

Parent Elements: <dx:internal-preview>

Child Elements: none

Required Attributes: *id*

id

Reference to the *id* attribute of the <trans-unit> whose <source> or <target> content will be substituted in place of the <dx:content> element in the preview.

Optional Attributes: none

<dx:external-preview>

The content of this element must be a URL to the HTML code that will be used to generate a source or target preview. The URL must be a relative path to a local file, as XLIFF:doc requires that the target of the URL be available to all tools that process the XLIFF. When using an external file, the HTML code must not be escaped. The preview code must include one <dx:content> element for each <trans-unit> in each <file> element.

If you include this element in a <file> element, you cannot include a <dx:internal-preview> element.

Parent Elements: <header>

Child Elements: none

Required Attributes: none

Optional Attributes: none

<dx:external-xsl>

The content of this element must be a URL to the XSL code that will be used to generate a source or target preview. The URL must be a relative path to a local file, as XLIFF:doc requires that the target of the URL be available to all tools that process the XLIFF. The XSL code must not be escaped.

Parent Elements: <header>

Child Elements: none

Required Attributes: none

Optional Attributes: none

Interoperability

<dx:internal-preview>

The content of this element is the HTML code that will be used to generate a source or target preview. When embedding the preview in the XLIFF:doc, the HTML code must be escaped. The preview code must include one <dx:content> element for each <trans-unit> in each <file> element.

If you include this element in a <file> element, you cannot include a <dx:external-preview> element.

Parent Elements: <header>

Child Elements: <dx:content>+

Required Attributes: none

Optional Attributes: none

<dx:qa-hit>

Parent Elements: <dx:qa-hits>

Child Elements: none

Required Attributes: *dx:qa-origin*, *dx:qa-ignorable*, *x:qa-shorttext*, *dx:qa-category*, *dx:qa-level*

dx:qa-origin

Indicates whether the QA hit originates from the source, the target, or from both. If the QA system identifies not just errors and warnings, but also non-errors (the correct thing was done by the linguist), then it can use a value of "both" here, and collapse the source-to-target hit, and the target-to-source hit into a single hit.

Accepted values: "source", "target", "both"

dx:qa-ignorable

Provides information to the tool reading the file as to whether this QA message can safely be ignored by the user. This determination is made by the tool that creates the QA message. The information must be made available to the tool user, but further behavior is not specified or required.

Accepted values: "yes", "no"

x:qa-shorttext

Human-readable description of the QA message. Must include specific information whenever possible, rather than a general description of the error.

dx:qa-category

Categorizes the type of QA message involved. The following table may be of assistance in matching your tool's QA categories to those supported in XLIFF:doc:

Category	Description
date	A date present in one language was missing or different in the other language
inconsistency	Two or more TUs have the same source but different translations; two or more TUs have the same translation but different sources
internationalization	Corrupt characters; spacing before units of measurement; formatting of numbers does not match the standards for the target language; punctuation marks were not properly translated into the target language; translation is incorrect for other cultural (non-linguistic) reasons

Interoperability

Category	Description
number	A number found in the source is different or missing in the translation; a number found in the translation is missing or different in the source; a number is found in both source and translation, but the units of measurement do not match.
omission	Empty translations; incomplete translations
other	Significant difference in length of source text compared to translation; every other type of QA hit not covered by one of the specific categories
pattern	A regular-expression pattern or other pattern (as defined in the QA checker) found a match (or failed to find a match, depending on usage)
punctuation	Differences in spacing, punctuation, brackets, tab characters, or capitalization
tags	One or more tags (placeholders/inline formatting) is missing or different in either the source or target; one or more tags is duplicated in the translation
terminology	Terminology usage in either the source or translation is not consistent with the termbase; terms were used that were flagged in the termbase as not-for-use
time	A time of day present in one language was missing or different in the other language

Acceptable values: "number", "pattern", "punctuation", "tags", "date", "time", "internationalization", "omission", "inconsistency", "terminology", "other"

dx:qa-level

Defines the degree of seriousness of the QA message.

Accepted values: "error" (most serious), "warning", "non-error" (proactive match/confirmation)

Optional Attributes: *dx:qa-code*, *dx:qa-tgt-startpos*, *dx:qa-tgt-length*, *dx:qa-src-startpos*, *dx:qa-src-length*

dx:qa-code

Tool-specific ID code for the error type. XLIFF:doc does not specify any limitations for the value of this optional attribute.

☼ *Note: this is not an ID number for the specific QA hit instance. There is no ID attribute in XLIFF:doc for a QA hit instance.*

dx:qa-tgt-startpos

The position, in unicode characters, from the start of the target segment, with 1 as the first position. Set to "0" if the position is unknown.

dx:qa-tgt-length

The length, in unicode characters, of the text in the target related to the QA hit. Set to "0" if the length is unknown.

dx:qa-src-startpos

The position, in unicode characters, from the start of the source segment, with 1 as the first position. Set to "0" if the position is unknown.

dx:qa-src-length

The length, in unicode characters, of the text in the source related to the QA hit. Set to "0" if the length is unknown.

<dx:qa-hits>

Container for individual QA hit elements.

Interoperability

Parent Elements: [<trans-unit>](#)

Child Elements: [<dx:qa-hit>+](#)

Required Attributes: none

Optional Attributes: none

[<dx:term-hit>](#)

This element has one function: to provide a look-up reference to the embedded UTX glossary.

Parent Elements: [<dx:term-hits>](#)

Child Elements: none

Required Attributes: *id*

id

Reference to the *entry-id* attribute of the [<dx:utx-term>](#) that contains a term designated as a match or hit for this element's grand-parent [<trans-unit>](#).

Optional Attributes: none

[<dx:term-hits>](#)

Container for individual terminology hit elements.

Parent Elements: [<trans-unit>](#)

Child Elements: [<dx:term-hit>+](#)

Required Attributes: none

Optional Attributes: none

[<dx:utx-comment>](#)

This element maps optional comment information from a UTX glossary. This information must be presented to the user. What you put into the comment, or whether you include one at all, is not defined by XLIFF:doc.

Parent Elements: [<dx:utx-term>](#)

Child Elements: none

Required Attributes: none

Optional Attributes: none

[<dx:utx-glossary>](#)

This element contains data from a UTX glossary, mapped to XML. The contents of this element will be referenced by individual terminology hits within each TU.



Example:

```
<dx:utx-glossary version="1.11" src-lang="en-US" tgt-lang="ja-JP" date-created="2011-08-17T19:00:00Z" creator="Acme Corporation">...</dx:utx-glossary>
```

Parent Elements: [<xliiff>](#)

Child Elements: [<dx:utx-term>+](#)

Required Attributes: *version, src-lang, tgt-lang, date-created, creator*

Interoperability

version

Maps to the version item in the header of the UTX file (position 1)

src-lang

Maps to the source language item in the header of the UTX file (position 2 - before slash). ISO 639 and 3166 formats.

tgt-lang

Maps to the target language item in the header of the UTX file (position 2 - after slash). ISO 639 and 3166 formats.

date-created

Maps to the date created item in the header of the UTX file (position 3). ISO 8601 format.

creator

Maps to the creator item in the header of the UTX file (position 4)

Optional Attributes: none

<dx:utx-src>

This element maps information related to the source language, from a UTX glossary (column 1). The text contents of the element must consist of the term as represented in the source language.

Parent Elements: [<dx:utx-term>](#)

Child Elements: none

Required Attributes: *pos*

pos

This is the source part-of-speech from the UTX glossary (column 3). See the UTX standard for more information.

Acceptable values: noun, properNoun, verb, adjective, adverb, sentence

Optional Attributes: none

<dx:utx-term>

This element has one function: to provide a look-up reference to the embedded UTX glossary. It must contain 1 [<dx:utx-src>](#) element and 1 [<dx:utx-tgt>](#) element. It may optionally have 1 [<dx:utx-comment>](#) element.

Parent Elements: [<dx:utx-glossary>](#)

Child Elements: [<dx:utx-src>](#){1,1}, [<dx:utx-tgt>](#){1,1}, [<dx:utx-comment>](#)?

Required Attributes: *entry-id*

entry-id

This is a unique id that will be referenced by every TU that has a hit for this term.

Acceptable values: any value that is unique within the XLIFF:doc file.

Optional Attributes: *concept-id*

concept-id

This is an optional reference to an external glossary. XLIFF:doc does not require a tool to act on this id if provided, by, for example, providing lookup against external terminology sources, such as TBX files included in the TIP, a term server, etc.

Interoperability

<dx:utx-tgt>

This element maps information related to the target language, from a UTX glossary (column 2). The text contents of the element must consist of the term as represented in the target language.

Parent Elements: <dx:utx-term>

Child Elements: none

Required Attributes: *status*

status

This is the status of the target language term, from the UTX glossary (optional column in UTX, not optional in XLIFF:doc). See the UTX standard for more information.

Acceptable values: provisional, approved, non-standard, forbidden

Optional Attributes: none

<external-file>

The content of this element must be a URL to the skeleton file containing the original file's structural code. This can be an absolute path to a file on a server, a relative path to a local file, etc. This element must be specified in XLIFF:doc.

XLIFF:doc does not, however, require that the target of the URL be available to all tools that process the XLIFF. This gives a Creation Tool freedom to include the skl file in the file package with the XLIFF where any user of the XLIFF can access it, or to leave it on the server where it can only be accessed by authorized users.

Parent Elements: <skl>

Child Elements: none

Required Attributes: none

Optional Attributes: none

<file>

Parent Elements: <xliff>

Child Elements: <header>{1,1}, <body>{1,1}

Required Attributes: *original*, *source-language*, *datatype*, *target-language*, *xml:space*, *dx:allow-tu-join*, *dx:allow-tu-split*, *dx:allow-tm-match-addition*, *dx:allow-tm-match-deletion*, *dx:allow-tb-match-addition*, *dx:allow-tb-match-deletion*, *dx:allow-qa-addition*, *dx:allow-qa-deletion*

xml:space

Controls whether whitespace is preserved for all TUs under this <file>. For more information [see the section on xml:space in the XML specification](#).

Acceptable values: "default", "preserve"

☼ *Note: xml:space is only allowed in the <file> element in XLIFF:doc.*

original

Reference to the original source file. Details of specification are not defined by XLIFF:doc, and can be decided on by the Creation Tool.

Acceptable values: not defined, but must be a string.

Interoperability

datatype

See the XLIFF 1.2 specification. XLIFF:doc does not define a preset list of values for this attribute. We recommend the following pattern: "x-[mimetype]". For example, "x-application/vnd.oasis.opendocument.text".

source-language, target-language

Acceptable values: as defined by ISO 639-1 (2-letter codes) or ISO 639-2T (3-letter codes). The use of a country code extension (based on ISO 3166) to the language code is acceptable. If extensions are used, they must use the dash "-" character, not the underscore. eg, "source-language="en-US". Lower case language codes and upper case country codes are recommended, but all tools must treat the values as case insensitive.

☀ *Note: See [BCP 47](#) for more information.*

dx:allow-tu-join

Determines whether or not TU joining is allowed in principle, for the file in question. Actual join ability may be further limited by document structure. Processing Tools are not permitted to modify this value. Default value: "yes".

dx:allow-tu-split

Determines whether or not TU splitting is allowed for the file in question. Processing Tools are not permitted to modify this value. Default value: "yes".

dx:allow-tm-match-addition

Determines whether or not a Processing Tool is allowed to embed additional TM matches to the file. If this value is "no", then only the Creation Tool can embed TM matches in a file. If the value is "no", then every compatible tool is allowed to embed its own matches. Processing Tools are not permitted to modify this value. Default value: "no".

dx:allow-tm-match-deletion

Determines whether or not a Processing Tool is allowed to delete embedded TM matches from the file. If this value is "yes", then tools can allow users to delete TM matches from the XLIFF. Processing Tools are not permitted to modify this value. Default value: "no".

dx:allow-tb-match-addition

Determines whether or not a Processing Tool is allowed to embed additional terminology matches to the file. If this value is "no", then only the Creation Tool can embed terminology matches in a file. If the value is "no", then every compatible tool is allowed to embed its own matches. Processing Tools are not permitted to modify this value. Default value: "no".

dx:allow-tb-match-deletion

Determines whether or not a Processing Tool is allowed to delete embedded terminology matches from the file. If this value is "yes", then tools can allow users to delete terminology matches from the XLIFF. Processing Tools are not permitted to modify this value. Default value: "no".

dx:allow-qa-addition

Determines whether or not a Processing Tool is allowed to embed additional QA hits to the file. If this value is "no", then only the Creation Tool can embed QA hits in a file. If the value is "no", then every compatible tool is allowed to embed its own QA hits. Processing Tools are not permitted to modify this value. Default value: "no".

dx:allow-qa-deletion

Determines whether or not a Processing Tool is allowed to delete embedded QA hits from the file. If this value is "yes", then tools can allow users to delete QA hits from the XLIFF. Processing Tools are not permitted to modify this value. Default value: "no".

Interoperability

dx:allow-revision-history-deletion

Determines whether or not a Processing Tool is allowed to delete revision history data. If this value is "yes", then tools can provide users with a mechanism to delete one or all <alt-trans> elements that have alttranstype = "previous-version". Processing Tools are not permitted to modify this value. Default value: "no".

Optional Attributes: none

<g>

Parent Elements: <source>, <target>

Child Elements: none

Required Attributes: id, dx:orig-markup-open, dx:orig-markup-close, dx:equiv-markup-open, dx:equiv-markup-close, ctype

id

The id is the primary mechanism for matching placeholders from source to target. The id must be unique within the <trans-unit>, but ids can be recycled from trans-unit to trans-unit. An id used for a <g> element cannot be re-used for an <x> element within the same <trans-unit>.

Acceptable values: non-negative integers

dx:orig-markup-open

(fill this content)

dx:orig-markup-close

(fill this content)

dx:equiv-markup-open

(fill this content)

dx:equiv-markup-close

(fill this content)



Example:
some code example here.

ctype

XLIFF:doc allows the following values for ctype in the <g> element:

Value	Description	Original HTML Markup Allowed?
bold	Indicates a run of bolded text.	Yes
italic	Indicates a run of text in italics.	Yes
underlined	Indicates a run of underlined text.	Yes
link	Indicates a run of hyper-text.	No
x-superscript	Indicates a run of superscripted text.	Yes
x-subscript	Indicates a run of subscripted text.	Yes

Interoperability

x-fixed	Indicates a run of fixed-width text (for displaying software code, etc.)	Yes
x-font	Indicates a change in font.	Yes
x-fontsize	Indicates a change in font-size.	Yes
x-header	Indicates header type formatting. In HTML, for example, H1, H2, H3, etc.	Yes
x-title	Indicates that the enclosed text is identified as the title of some element. For example, the title attribute of an HTML <a> element.	No
x-other-format	All character level formatting not covered by specific ctype values.	No
x-other	All other types.	No

Optional Attributes: *equiv-text*

equiv-text

(fill this content)

<group>

In XLIFF:doc, the <group> element is used for a single purpose: to group <trans-unit> elements that can be joined without damaging the target document structure. Any sequential TUs within a <group> can be joined with each other. TUs from different groups may not be joined.

Parent Elements: <body>

Child Elements: <trans-unit>+

Required Attributes: *id*

id

The *id* must be unique within the <file>, but its ids can be recycled from file to file.

Acceptable values: any value that is unique within the <file> element.

Optional Attributes: none

<header>

Either <dx:internal-preview> or <dx:external-preview> must be specified by the Creation Tool. Do not specify both.

Parent Elements: <file>

Child Elements: <skl>{1,1}, <note>*, <tool>{1,1}, <dx:internal-preview>?, <dx:external-preview>?, <dx:external-xsl>?

Required Attributes: none

Optional Attributes: none

<note>

In XLIFF:doc, the note element can be applied to the file itself (via the <header> element), and translation pairs (via the <trans-unit> and <alt-trans> elements).

Interoperability



Example:

```
<note from="20110114T09:30:24Z-JoeTranslator" annotates="source">"driver" = typo in the
source? The reference material shows a cat in a scuba suit, not in a race car.</note>
<note from="20110114T11:42:43Z-JaneContentAuthor" annotates="source">Good catch, thanks.
Will fix to read "diver" in next revision.</note>
```

Parent Elements: `<trans-unit>`, `<alt-trans>`, `<header>`

Child Elements: none

Required Attributes: *from*, *annotates*

from

Due to the fact that XLIFF 1.2 does not allow us to add external XML namespaces to the `<note>` element, the *from* attribute must contain two pieces of information in XLIFF:doc, separated by a dash character (\x2D):

- Date/time modified: Time and date the note was last modified. This can represent the date/time it was created, editing, or processed in some other way.



Note: Date/time must be specified in ISO 8601 format. Time must be specified in UTC (Zulu) time zone. Do not use dashes to separate years, months, and days. Use colons to separate hours, minutes, and seconds. [See the XLIFF date element for more information.](#)

- User identifier: the user ID, name, role, or description of the person/system that created the note.

Acceptable values: as above.

annotates

Notes can optionally be defined as referring to specific text in the source or target. If specific text is to be associated with the note, the *annotates* attribute must be set, and set to either source or target.

Acceptable values: source, target, general. Default value: general.

Optional Attributes: none

`<skl>`

Use `<external-file>` to point to the original skeleton file (structural content of the original source file). XLIFF:doc does not support `<internal-file>` (embedded structure content within the XLIFF).

Parent Elements: `<header>`

Child Elements: `<external-file>`{1,1}

Required Attributes: none

Optional Attributes: none

`<source>`

Parent Elements: `<trans-unit>`, `<alt-trans>`

Child Elements: `<g>`*, `<x>`*

Required Attributes: none

Optional Attributes: none

`<target>`

Parent Elements: `<trans-unit>`, `<alt-trans>`

Interoperability

Child Elements: `<g>*`, `<x>*`

Required Attributes: *state*

state

Acceptable values: "new", "translated", "final", "signed-off", "needs-review-translation". Default value: "new".

Optional Attributes: none

`<tool>`

See the XLIFF 1.2 specification for more information. In XLIFF:doc, all 4 attributes are required, and must refer to the name of the tool that originally created the XLIFF. The values must not be modified by Processing Tools (to refer to themselves).



Example:

```
<tool tool-id="1" tool-name="MyTMServer" tool-version="v2.0.1" tool-company="Acme  
Rocketry Corporation" />
```

Parent Elements: `<header>`

Child Elements: none

Required Attributes: *tool-id*, *tool-name*, *tool-version*, *tool-company*

Optional Attributes: none

`<trans-unit>`

In XLIFF:doc, the TU can be assigned information about the match that was applied to it (if any). This information must be assigned to a TU if a match is applied from any source: embedded `<alt-trans>` matches, live TM lookup, live MT application, etc. If, however, subsequent to applying a match, the user edits the `<target>` element, then the match-related attributes (*alttranstype*, *dx:match-quality*, *dx:glorious-match*, *dx:match-penalty*, *origin*) must be removed, as the translation no longer represents the match as applied.

Parent Elements: `<group>`

Child Elements: `<source>{1,1}`, `<target>{1,1}`, `<alt-trans>*`, `<note>*`, `<dx:qa-hits>*`, `<dx:term-hits>*`

Required Attributes: *id*, *dx:match-quality*, *dx:glorious-match*, *dx:repetition*, *dx:match-penalty*, *dx:origin*, *dx:origin-shorttext*, *dx:modified-by*, *dx:modified-at*

id

Acceptable values: any value that is unique within the XLIFF:doc file.

dx:match-quality

Use values 0-100. The use of decimal numbers is acceptable, but only a single digit of precision is allowed. For example, "99.5" is acceptable, but "99.52" is not. Do not use a percent sign. Higher numbers refer to better matches than lower numbers.

Remove this attribute if the user manually edits the `<target>` after the match was placed.



Note: this mimics the *match-quality* attribute of `<alt-trans>` in XLIFF 1.2, giving us the same functionality for the `<trans-unit>` element.

dx:glorious-match

Interoperability

Indicates whether the match is an exact, in-context match (as determined by the tool that generated the match). Remove this attribute if the user manually edits the `<target>` after the match was placed. Acceptable values: "yes", "no". Default value: "no".

dx:repetition

Indicates whether this translation unit has been classified as a repetition (same exact source as other translations in the same project). XLIFF:doc does not specify how to determine if a TU is a repetition or not (2 occurrences vs 3, etc.).

XLIFF:doc does not distinguish between first instance of a repetition and subsequent instances. Acceptable values:

"yes", "no". Default value: "no".

dx:origin

Acceptable values: "TM", "MT", "XLIFF", "manual", "sourcetext", "pseudo-trans". If the match was pulled from the `<alt-trans>` in the XLIFF:doc doc, this value must match the origin value of the `<alt-trans>` match.

Value	Description
TM	Indicates the match is from translation memory
MT	Match is from machine translation of some type
XLIFF	Match was copied directly from another XLIFF file
manual	A user edited the <code><target></code> element directly (by typing, pasting text, etc.)
sourcetext	The source was copied to the target via some automated means (a pre-translation task, or user selected Edit > Copy Source to Target, etc.). If a user copies the source text themselves, and pastes into the target, that would be "manual".
pseudo-trans	Some type of pseudo-translation function has populated the target. For example, an "X" was placed in the target for each character in the source text.

☀ Note: this mimics the *origin* attribute of XLIFF 1.2, and allows it to be used in the `<trans-unit>` element.

dx:origin-shorttext

Human-readable string describing the match origin. This could be the name of the TM it came out of, the name of an MT engine, etc. Examples: "Master TM 1", "Google MT", "myDocument.docx.xlf", "Manual Translation". Exact values are not defined by XLIFF:doc.

dx:match-penalty

Identifies what, if any, artificial (non-textual) penalty was applied to the match by the tool that generated the match. Use values 0-100. The use of decimal numbers is acceptable, but only a single digit of precision is allowed. For example, "0.5" is acceptable, but "0.48" is not. Do not use a percent sign. Higher numbers refer to larger penalties than lower numbers.

Remove this attribute if the user manually edits the `<target>` after the match was placed.

dx:modified-by

String value identifying the user (human or automatic) responsible for the most recent modification. Modification could include changing an attribute of the `<trans-unit>` element, or any change to the `<target>` element.

dx:modified-at

Interoperability

Time and date the `<trans-unit>` was last modified. This can represent the date/time the translation was edited, or when a TM match was applied, or when the `<trans-unit>` was processed in some other way.

☀ *Note: Date/time must be specified in ISO 8601 format. Time must be specified in UTC (Zulu) time zone. [See the XLIFF date element for more information.](#)*

Optional Attributes: *translate*, *dx:xsl-hook*

translate

Acceptable values: "yes", "no". If set to "no", this TU will not be editable by tool users. Default value: "yes".

☀ *Note: It is up to the Processing Tool to determine if users can see non-translatable TUs or not. The only thing XLIFF:doc requires is that users cannot edit them.*

dx:xsl-hook

This attribute is used to optionally provide information to an XSLT file, about the translation unit. This information would typically be used to help determine how the contents of the translation unit should be displayed in a preview. XLIFF:doc does not specify how this attribute should be populated or used.

Acceptable values: undefined. Must be a string value.

`<x>`

Parent Elements: `<source>`, `<target>`

Child Elements: none

Required Attributes: *id*, *dx:orig-markup*, *dx:equiv-markup*, *ctype*

id

The *id* is the primary mechanism for matching placeholders from source to target. The *id* must be unique within the `<trans-unit>`, but its ids can be recycled from trans-unit to trans-unit. An *id* used for a `<g>` element cannot be re-used for an `<x>` element within the same `<trans-unit>`.

Acceptable values: non-negative integers

dx:orig-markup

(fill this content)

dx:equiv-markup

(fill this content)

☀ *Example:*
some code example here.

ctype

XLIFF:doc allows the following values for *ctype* in the `<x>` element:

Value	Description	Original HTML Markup Allowed?
image	Indicates a inline image.	Yes
pb	Indicates a page break.	No
lb	Indicates a line break.	No

Interoperability

x-xref	Indicates a cross-reference.	No
x-tab	Indicates a tab character.	No
x-other	All other types.	No

Optional Attributes: *equiv-text*

equiv-text

(fill this content)

<xliff>

Parent Elements: none

Child Elements: <file>+, <dx:utx-glossary>*

Required Attributes: *dx:version*, *version*, *xmlns*, *xmlns:dx*, *xmlns:xsi*, *xsi:schemaLocation*

In XLIFF:doc, the <xliff> element contains critical information which must be set in order for an XLIFF:doc to be recognized as an XLIFF:doc file, and also to allow for proper validation of the XML. With the exception of the *dx:version* attribute, all attributes and attribute values are fixed.



Example:

```
<xliff version="1.2" xmlns="urn:oasis:names:tc:xliff:document:1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:dx="http://www.interoperability-now.org/schema" xsi:schemaLocation="urn:oasis:names:tc:xliff:document:1.2 xliff-doc-1_0_extensions.xsd" dx:version="1.4">
```

version

The *version* attribute specifies the version of XLIFF. At the current time, the only allowed version is “1.2”.

Acceptable values: “1.2”

dx:version

The *dx:version* attribute specifies the version of the XLIFF:doc XSD used.

Acceptable values: non-negative numbers corresponding to published XLIFF:doc XSD versions.

xmlns

The *xmlns* namespace declaration for XLIFF must be included.

Acceptable values: “urn:oasis:names:tc:xliff:document:1.2”

xmlns:dx

The *xmlns:dx* namespace declaration for XLIFF:doc must be included.

Acceptable values: “http://www.interoperability-now.org/schema”

xmlns:xsi

The *xmlns:xsi* namespace declaration is currently required.

Acceptable values: “http://www.w3.org/2001/XMLSchema-instance”

xmlns:xsi=“http://www.w3.org/2001/XMLSchema-instance”

xsi:schemaLocation

Interoperability

XLIFF:doc currently requires the following XSI callout, which helps manage the limitations XLIFF:doc places on the XLIFF 1.2 specification.

Acceptable values: "urn:oasis:names:tc:xliff:document:1.2 xliff-doc-1_0_extensions.xsd"

Appendix C: [TEMP:] Misc. Questions

This is for questions we want to track during the development of the reference guide, which are not captured elsewhere in the document.

License

Sven: I know it's early for this, but I still think it's important to be clear here: What license should be used for this document?

Micah: I'm not too qualified to answer that, but wouldn't we want to aim for the more open end of the spectrum? MIT license for example?

Ability to Create Target Files from the XLIFF (including structure in xliiff)

Micah: Gergely Vandro doesn't believe this is practical. I'd like to talk about it, because this is what we do with WS 9.0.4 xliiffs, and we had to reverse engineer those. It shouldn't be *that* hard if we control the format ourselves. But... I'm sure there are some issues I haven't thought through...

Sven: I'm not sure of people want this. It might make sense in some cases, in others, companies might not want to have the information "between the lines" being sent to the world.

Micah: I'll give you the two Medtronic use cases:

- 1) we need translators to be able to translate "in context". That means they can see where the segment they are working on comes from, what's before and after it, if it's in a caption, table cell, paragraph, etc. If the structure is not included in the package, it's not possible for the translation editor tool to give that information to the translator. One of the main reasons we stopped using the WS offline workbench is because there was no context info present, so translators always had to do multiple target generation/check cycles, which they didn't always do, and that affected quality.
- 2) we don't currently have the ability for translators to touch the system directly, for security/VPN, and CFR Part 11 Compliance reasons. That means it can take several days for translators to be able to get a target that shows their edits. (translator in france > vendor PM in beijing > MDT in minneapolis > WS > beijing > france). So having a complete bundle that is self-contained is a huge boost to anyone in a similar situation.
- 3) (and this is related to software, not docs) we have what we call "simulators" that talk to our translation editor. The simulators are software test programs. They let engineers simulate various inputs into a device, but they have also been modified to be localization tools. So a linguist working on a software package can hit a button, generate a target, and see the changed text in the simulator UI instantly (and other syncing features). But if the tool can't generate a target, it won't work.

Sven:

As written earlier, I'm still not sure, if this is best for all cases. As much as I like it for some cases, I also see some drawbacks:

- * it might be against the interest of a client to give all involved parties access to the complete source
- * It might blow up the files enormously and might lead to problems with some editors.

I think it's important to specify how the skl is stored and limit the variations of this to 1 for external and 1 for internal.

But there should be an attribute specifying for the document if skl is included or not.

Interoperability

Micah: I definitely think there should be an option for the generating tool to either include, or exclude the structure information. Based on user (owner) desire (some, as above, may not want that info out there). Having dealt with big fat embedded files a lot, I would prefer to have the skl info there, but in a separate file I (as the tool) could handle in a separate operation.

Sven: Perhaps it would make sense to allow attaching things like a special preview XSL.

Micah: so not sure what exactly you had in mind here, but one thought would be for the Creation Tool to NOT include the full skl, but to include an xhtml structure that approximated the original document (no matter what format that was). That way, any tool that operated on the xliif could generate a pseudo target, or preview target, without access to the actual structure. However, it would be a bastardized target, and I can see a lot of linguists bashing it. And, we'd have to dictate the format of that XHTML to the extent necessary to make it clear to tools how to repopulate it with target data.

QA Hits:

Gabór is going to review this and suggest some changes to how we indicate the position of the QA hit within the source or target. Concern about what happens when the translation gets changed, which would throw off the offset and/or the length.

Usage Examples:

Chase's email of march 16:

- In general, the specification is lacking in the level of precision that makes consistent implementation possible. In general, I would encourage the spec to focus on a few use cases, and get specific about what tools should do with various data. There is a reference in the spec to the "localization workflow", but no discussion of what this actually is and how the format supports it.

A good test for the spec might be to consider some non-mainstream use cases and see whether they would cause any confusion or ambiguity. For example, would XLIFF:doc support a translator starting work, then leveraging a TM, then going back to translating? What could get lost or corrupted in this series of transformations? Should XLIFF:doc try to support this?

Micah: do we still need the usage example section?