TMX 1.4a Specification

OSCAR Recommendation, 10 July 2002

This version:
http://www.lisa.org/tmx/tmx14-20020710.htm

Latest version:
http://www.lisa.org/tmx/tmx.htm

Previous version:
http://www.lisa.org/tmx/tmx13.htm

Editor:
Yves Savourel <ysavourel@translate.com>

Abstract

This document defines the Translation Memory eXchange format (TMX). The purpose of the TMX

format is to provide a standard method to describe translation memory data that is being exchanged among tools and/or translation vendors, while introducing little or no loss of critical data during the process.

Status of this Document

This OSCAR Recommendation was approved for publication by the OSCAR Steering Committee. It is a stable document which represents the consensus of the committee. Comments may be sent to tmx@lisa.org.

Table of Contents

1. Introduction

TMX is defined in two parts:

- A specification of the format of the container (the higher-level elements that provide information about the file as a whole and about entries). In TMX, an entry consisting of aligned segments of text in two or more languages is called a Translation Unit (the `<tu>` element).
- A specification of a low-level meta-markup format for the content of a segment of translation-memory text. In TMX, an individual segment of translation-memory text in a particular language is denoted by a `<seg>` element. See the section Content Markup for more details.

TMX offers two levels of implementation:

- **Level 1** (Plain Text Only) - Support for the container only. The data inside each `<seg>` element is plain text, without Content Markup. This level is enough when the data do not have inline codes, for example software messages. It is not sufficient for documentation-type formats.
- **Level 2** (Content Markup) - Support for both container and content. The application uses the TMX Content Markup, to allow any other tools supporting also TMX Level 2 to re-create the translated version of an original document by using only the TMX document. This level does not take in account segmentation difference.

See the section Level 2 Implementation for more details.

1.1. XML Compliance

TMX is XML-compliant. It also uses various ISO standards for date/time, language codes, and country codes. See the References section for more details.

TMX files are intended to be created automatically by export routines and processed automatically by import routines. TMX files are "well-formed" XML documents that can be processed without explicit reference to the TMX DTD. However, a "valid" TMX file must conform to the TMX DTD, and any suspicious TMX file should be verified against the TMX DTD using a validating XML parser.

Since XML syntax is case sensitive, any XML application must define casing conventions. All elements and attributes names of TMX are defined in **lowercase**.

The TMX namespace is defined as "`http://www.lisa.org/tmx14`". For example, if you want to use TMX in another XML document you document would look something like:

```
<?xml version="1.0"?>
<myformat>
 <data>
  <tmx xmlns="http://www.lisa.org/tmx14"
       version="1.4">
   ... TMX data ...
  </tmx>
 </data>
</myformat>
```

1.2. Character Encoding

TMX files are always in Unicode. They can use either of three encoding methods: UTF-16 (16-bit files), UTF-8 (8-bit files) or ISO-646 [a.k.a. US-ASCII] (7-bit files).

In all cases, unlike in HTML, only the following five character entity references are allowed: `&amp;` (&), `&lt;` (<), `&gt;` (>), `&apos;` ('), and `&quot;` ("). For 7-bit files, extended (non-ASCII) characters are always represented by numeric character references. For example: `&#x0396;` or `&#918;` for a `GREEK CAPITAL LETTER DELTA`.

Since all XML processors must accept the UTF-8 and UTF-16 encodings and since US-ASCII is a subset of UTF-8, a TMX document can omit the encoding declaration in the XML declaration.

Note that UTF-16 files always start with the Unicode byte-order-mark (BOM) value: U+FEFF.

2. General Structure

A TMX document is enclosed in a `<tmx>` root element. The `<tmx>` element contains two elements: `<header>` and `<body>`.

2.1. Header

The `<header>` contains meta-data about the document. In addition to its attributes, `<header>` can also store document-level information in `<note>` and `<prop>` elements. It is also in this element that any user-defined characters can be listed, using `<ude>` elements.

2.2. Body

The `<body>` contains the collection of translation units (the `<tu>` elements). This collection is in no specific order.

Each `<tu>` element contains at least one translation unit variant, the `<tuv>` element. Each `<tuv>` contains the segment and the information pertaining to that segment for a given language.

The text itself is stored in the `<seg>` element, while `<note>` and `<prop>` allow you to store information specific to each `<tuv>`.

A segment can contain markup content elements: The `<bpt>`, `<ept>`, `<it>`, and `<ph>` elements allow you to encapsulate original native inline codes. The `<hi>` element allows you to add extra markup not related to existing inline codes. And the `<sub>` element, used inside encapsulated inline code, allows you to delimits embedded text.

See the Sample Document section for an example of TMX document.

3. Detailed Specifications

3.1. Elements

TMX elements are divided into two main categories: the structural elements (the container), and the inline elements (the content markup).

Structural elements	`<body>`, `<header>`, `<map/>`, `<note>`, `<prop>`, `<seg>`, `<tmx>`, `<tu>`, `<tuv>`, `<ude>`.
Inline elements	`<bpt>`, `<ept>`, `<hi>`, `<it>`, `<ph>`, `<sub>`, `<ut>`.

3.1.1. Structural Elements

The structural elements are the following:

<body>

Body - The `<body>` element encloses the main data, the set of `<tu>` elements that are comprised within the file.

Required attributes:

None.

Optional attributes:

None.

Contents:

Zero, one or more `<tu>` elements.

<header>

File header - The `<header>` element contains information pertaining to the whole document.

Required attributes:

`creationtool`, `creationtoolversion`, `segtype`, `o-tmf`, `adminlang`, `srclang`, `datatype`.

Optional attributes:

`o-encoding`, `creationdate`, `creationid`, `changedate`, `changeid`.

Contents:

Zero, one or more `<note>`, `<ude>`, or `<prop>` elements in any order.

<map/>

Map - The `<map/>` element is used to specify a user-defined character and some of its properties.

Required attributes:

unicode.

Optional attributes:

code, ent and subst. At least one of these attributes should be specified. If the code attribute is specified, the parent <ude> element must specify a base attribute.

Contents:

Empty.

<note>

Note - The <note> element is used for comments.

Required attributes:

None.

Optional attributes:

xml:lang, o-encoding.

Contents:

Text.

<prop>

Property - The <prop> element is used to define the various properties of the parent element (or of the document when <prop> is used in the <header> element). These properties are not defined by the standard.

As your tool is fully responsible for handling the content of a <prop> element you can use it in any way you wish. For example the content can be a list of instructions your tool can parse, not only a simple text.

```
<prop type="user-defined">name:domain value:Computer science</prop>
<prop type="x-domain">Computer science</prop>
```

It is the responsibility of each tool provider to publish the types and values of the properties it uses. If the tool exports unpublished properties types, their values should begin with the prefix "x-".

Required attributes:

type.

Optional attributes:

xml:lang, o-encoding.

Contents:

Tool-specific data or text.

<seg>

Segment - The `<seg>` element contains the text of the given segment. There is no length limitation to the content of a `<seg>` element. All spacing and line-breaking characters are significant within a `<seg>` element.

Required attributes:

None.

Optional attributes:

None.

Contents:

Text data (without leading or trailing white spaces characters),
Zero, one or more of the following elements: `<bpt>`, `<ept>`, `<it>`, `<ph>`, and `<hi>`.
They can be in any order, except that each `<bpt>` element must have a subsequent corresponding `<ept>` element.

<tmx>

TMX document - The `<tmx>` element encloses all the other elements of the document.

Required attributes:

`version`.

Contents:

One `<header>` followed by
One `<body>` element.

<tu>

Translation unit - The `<tu>` element contains the data for a given translation unit.

Required attributes:

None.

Optional attributes:

`tuid`, `o-encoding`, `datatype`, `usagecount`, `lastusagedate`, `creationtool`, `creationtoolversion`, `creationdate`, `creationid`, `changedate`, `segtype`, `changeid`, `o-`

`tmf`, `srclang`.

Contents:

Zero, one or more `<note>`, or `<prop>` elements in any order, followed by
One or more `<tuv>` elements.
Logically, a complete translation-memory database will contain at least two `<tuv>` elements in each translation unit.

<tuv>

Translation Unit Variant - The `<tuv>` element specifies text in a given language.

Required attributes:

`xml:lang`.

Optional attributes:

`o-encoding`, `datatype`, `usagecount`, `lastusagedate`, `creationtool`, `creationtoolversion`, `creationdate`, `creationid`, `changedate`, `changeid`, `o-tmf`.

Contents:

Zero, one or more `<note>`, or `<prop>` elements in any order, followed by
One `<seg>` element.

<ude>

User-Defined Encoding - The `<ude>` element is used to specify a set of user-defined characters and/or, optionally their mapping from Unicode to the user-defined encoding.

Required attributes:

`name`.

Optional attributes:

`base` (required if one or more of the `<map/>` elements contains a `code` attribute).

Contents:

One or more `<map/>` elements.

3.1.2. Inline Elements

The inline elements are the elements that can appear inside the a segment. With the exception of the `<hi>` and `<sub>` element, they all enclose or replace any formatting or control codes that is not text but resides within the segment. See also the Content Markup section for more information.

The inline elements are the following:

<bpt>

Begin paired tag - The `<bpt>` element is used to delimit the beginning of a paired sequence of native codes. Each `<bpt>` has a corresponding `<ept>` element within the segment.

Required attributes:

`x`, `i`.

Optional attributes:

`type`.

Contents:

Code data,
Zero, one or more `<sub>` elements.

<ept>

End paired tag - The `<ept>` element is used to delimit the end of a paired sequence of native codes. Each `<ept>` has a corresponding `<bpt>` element within the segment.

Required attributes:

`i`.

Optional attributes:

None.

Contents:

Code data,
Zero, one or more `<sub>` elements.

<hi>

Highlight - The `<hi>` element delimits a section of text that has special meaning, such as a terminological unit, a proper name, an item that should not be modified, etc. It can be used for various processing tasks. For example, to indicate to a Machine Translation tool proper names that should not be translated; for terminology verification, to mark suspect expressions after a grammar checking.

Required attributes:

None.

Optional attributes:

`x`, `type`.

Contents:

Text data,
Zero, one or more of the following elements: `<bpt>`, `<ept>`, `<it>`, `<ph>`, and `<hi>`.
They can be in any order, except that each `<bpt>` element must have a subsequent corresponding `<ept>` element.

<it>

Isolated tag - The `<it>` element is used to delimit a beginning/ending sequence of native codes that does not have its corresponding ending/beginning within the segment.

Required attributes:

`pos`.

Optional attributes:

`x`, `type`.

Contents:

Code data,
Zero, one or more `<sub>` elements.

<ph>

Placeholder - The `<ph>` element is used to delimit a sequence of native standalone codes in the segment.

Required attributes:

None.

Optional attributes:

`x`, `type`, `assoc`.

Contents:

Code data,
Zero, one or more `<sub>` elements.

<sub>

Sub-flow - The `<sub>` element is used to delimit sub-flow text inside a sequence of native code, for example: the definition of a footnote or the text of title in a HTML anchor element.

Here are some examples (translatable text underlined, sub-flow is bolded):

Footnote in RTF:

```
Original RTF:
Elephants{\cs16\super \chftn {\footnote \pard\plain
\s15\widctlpar \f4\fs20
{\cs16\super \chftn } An elephant is a very
large animal.}} are big.

TMX with content mark-up:
Elephants<ph type="fnote">{\cs16\super \chftn {\footnote \pard\plain
\s15\widctlpar \f4\fs20
{\cs16\super \chftn } <sub>An elephant is a very
large animal.</sub>}}</ph> are big.
```

Index marker in RTF:

```
Original RTF:
Elephants{\pard\plain \widctlpar
\v\f4\fs20 {\xe {Big animal\bxe }}} are big.

TMX with content mark-up:
Elephants<ph type="index">{\pard\plain \widctlpar
\v\f4\fs20 {\xe {<sub>Big animal</sub>\bxe }}}</ph> are big.
```

Text of an attribute in a HTML element:

```
Original HTML:
See the <A TITLE="Go to Notes"
HREF="notes.htm">Notes</A> for more details.

TMX with content mark-up:
See the <bpt i="1" type="link">&lt;A TITLE="<sub>Go to Notes</sub>"
HREF="notes.htm"></bpt>Notes<ept i="1">&lt;/A></ept> for more details.
```

Note that sub-flow are related to segmentation and can cause interoperability issues when one tool uses sub-flow within its main segment, while another extract the sub-flow text as an independent segment.

Required attributes:

None.

Optional attributes:

`datatype`, `type`.

Contents:

Text data,
Zero, one or more of the following elements: `<bpt>`, `<ept>`, `<it>`, `<ph>`, and `<hi>`.
They can be in any order, except that each `<bpt>` element must have a subsequent corresponding `<ept>` element.

<ut>

Unknown Tag - The `<ut>` element is used to delimit a sequence of native unknown codes in the segment.

This element has been DEPRECATED. Use the guidelines outlined in the Rules for Inline Elements section to choose which inline element to used instead of `<ut>`.

Required attributes:
None.

Optional attributes:
`x`.

Contents:
Code data,
Zero, one or more `<sub>` elements.

3.2. Attributes

This section lists the various attributes used in the TMX elements.

TMX attributes	`adminlang`, `assoc`, `base`, `changedate`, `changeid`, `code`, `creationdate`, `creationid`, `creationtool`, `creationtoolversion`, `datatype`, `ent`, `i`, `lastusagedate`, `name`, `o-encoding`, `o-tmf`, `pos`, `segtype`, `srclang`, `subst`, `tuid`, `type`, `unicode`, `usagecount`, `version`, `x`.
XML namespace attributes	`xml:lang`.

3.2.1. TMX Attributes

---

adminlang

Administrative language - Specifies the default language for the administrative and informative elements `<note>` and `<prop>`.

Value description:

A language code as described in the [RFC 3066]. Unlike the other TMX attributes, the values for `adminlang` are not case-sensitive.

Default value:

Undefined.

Used in:

`<header>`.

---

assoc

Association - Indicates the association of a `<ph>` with the text prior or after.

Value description:

"`p`" (the element is associated with the text preceding the element), "`f`" (the element is associated with the text following the element), or "`b`" (the element is associated with the text on both sides).

Default value:

Undefined.

Used in:

`<ph>`.

---

base

Base encoding - Specifies the encoding upon which the re-mapping of the `<ude>` element is based.

Value description:

One of the [IANA] recommended "charset identifier", if possible.

Default value:

Undefined.

Used in:

`<ude>`.

---

changedate

Change date - Specifies the date of the last modification of the element.

Value description:

Date in [ISO 8601] Format. The recommended pattern to use is: `YYYYMMDDThh:mm:ssZ`
Where: `YYYY` is the year (4 digits), `MM` is the month (2 digits), `DD` is the day (2 digits), `hh` is the hours (2 digits), `mm` is the minutes (2 digits), `ss` is the second (2 digits), and `Z` indicates the time is UTC time. For example:

```
date="20020125T21:06:00Z"
is January 25, 2002 at 9:06pm GMT
is January 25, 2002 at 2:06pm US Mountain Time
is January 26, 2002 at 6:06am Japan time
```

Default value:

Undefined.

Used in:

`<header>`, `<tu>`, `<tuv>`.

---

changeid

Change identifier - Specifies the identifier of the user who modified the element last.

Value description:

Text.

Default value:

Undefined.

Used in:

`<header>`, `<tu>`, `<tuv>`.

---

code

Code - Specifies, in a user-defined encoding, the code-point value corresponding to the `unicode` character of a given `<map/>` element.

Value description:

Hexadecimal value prefixed with "`#x`". For example: `code="#x9F"`.

Default value:

Undefined.

Used in:

`<map/>`.

---

creationdate

Creation date - Specifies the date of creation of the element.

Value description:

Date in [ISO 8601] Format. The recommended pattern to use is: `YYYYMMDDThh:mm:ssZ`
Where: `YYYY` is the year (4 digits), `MM` is the month (2 digits), `DD` is the day (2 digits), `hh` is the hours (2 digits), `mm` is the minutes (2 digits), `ss` is the second (2 digits), and `Z` indicates the time is UTC time. For example:

```
date="20020125T21:06:00Z"
is January 25, 2002 at 9:06pm GMT
is January 25, 2002 at 2:06pm US Mountain Time
is January 26, 2002 at 6:06am Japan time
```

Default value:

Undefined.

Used in:

`<header>`, `<tu>`, `<tuv>`.

---

creationid

Creation identifier - Specifies the identifier of the user who created the element.

Value description:

Text.

Default value:

Undefined.

Used in:

`<header>`, `<tu>`, `<tuv>`.

---

creationtool

Creation tool - Identifies the tool that created the TMX document. Its possible values are not specified by the standard but each tool provider should publish the string identifier it uses.

Value description:

Text.

Default value:

Undefined.

Used in:

`<header>`, `<tu>`, `<tuv>`.

---

creationtoolversion

Creation tool version - Identifies the version of the tool that created the TMX document. Its possible values are not specified by the standard but each tool provider should publish the string identifier it uses.

Value description:

Text.

Default value:

Undefined.

Used in:

`<header>`, `<tu>`, `<tuv>`.

---

datatype

Data type - Specifies the type of data contained in the element. Depending on that type, you may apply different processes to the data.

Value description:

Text. The recommended values for the `datatype` attribute are as follow (this list is not exhaustive):
- "`unknown`" = undefined (default)
- "`alptext`" = WinJoust data.
- "`cdf`" = Channel Definition Format.
- "`cmx`" = Corel CMX Format.
- "`cpp`" = C and C++ style text.
- "`hptag`" = HP-Tag.
- "`html`" = HTML, DHTML, etc.
- "`interleaf`" = Interleaf documents.
- "`ipf`" = IPF/BookMaster.
- "`java`" = Java, source and property files.
- "`javascript`" = JavaScript, ECMAScript scripts.
- "`lisp`" = Lisp.
- "`mif`" = Framemaker MIF, MML, etc.
- "`opentag`" = OpenTag data.
- "`pascal`" = Pascal, Delphi style text.

- "plaintext" = Plain text.
- "pm" = PageMaker.
- "rtf" = Rich Text Format.
- "sgml" = SGML.
- "stf-f" = S-Tagger for FrameMaker.
- "stf-i" = S-Tagger for Interleaf.
- "transit" = Transit data.
- "vbscript" = Visual Basic scripts.
- "winres" = Windows resources from RC, DLL, EXE.
- "xml" = XML.
- "xptag" = Quark XPressTag.

Default value:
"unknown".

Used in:
<header>, <tu>, <tuv>, <sub>.

---

ent

Entity - Specifies the entity name of the character defined by a given <map/> element.

Value description:
Text in ASCII. For example: ent="copy".

Default value:
Undefined.

Used in:
<map/>.

---

i

Internal matching - The i attribute is used to pair the <bpt> elements with <ept> elements. This mechanism provides TMX with support to markup a possibly overlapping range of codes. Such constructions are not used often, however several formats allow them. For example, the following HTML segment, even if not strictly legal, is accepted by some HTML editors and usually interpreted correctly by the browsers.

For example:

```
[----------------------------]
<B>Bold <I>Bold and Italic</B> Italics</I>
```

```
        [--------------------------------]
```

With the TMX content mark-up, since the <ept> element does not have a type, it can be difficult to know which sequence of codes it closes as illustrated by the following segment:

```
TMX (with incomplete content mark-up):
<bpt>&lt;B></bpt>Bold,
<bpt>&lt;I></bpt>Bold+Italic<ept>&lt;/B></ept>,
Italic<ept>&lt;/I></ept>
```

The attribute i is used to specify which <ept> is closing which <bpt>:

```
TMX (with correct content mark-up):
<bpt i="1" x="1">&lt;B></bpt>Bold,
<bpt i="2" x="1">&lt;I></bpt>Bold+Italic<ept i="1">&lt;/B></ept>,
Italic<ept i="2">&lt;/I></ept>
```

Value description:

Number. Must be unique for each <bpt> within a given <seg> element.

Default value:

Undefined.

Used in:

<bpt>, <ept>.

---

lastusagedate

Last usage date - Specifies when the last time the content of a <tu> or <tuv> element was used in the original translation memory environment.

Value description:

Date in [ISO 8601] Format. The recommended pattern to use is: `YYYYMMDDThh:mm:ssZ`
Where: `YYYY` is the year (4 digits), `MM` is the month (2 digits), `DD` is the day (2 digits), `hh` is the hours (2 digits), `mm` is the minutes (2 digits), `ss` is the second (2 digits), and `Z` indicates the time is UTC time. For example:

```
date="20020125T21:06:00Z"
is January 25, 2002 at 9:06pm GMT
is January 25, 2002 at 2:06pm US Mountain Time
is January 26, 2002 at 6:06am Japan time
```

Default value:

Undefined.

Used in:

`<tu>`, `<tuv>`.

---

name

Name - Specifies the name of a `<ude>` element. Its value is not defined by the standard, but tools providers should publish the values they use.

Value description:

Text.

Default value:

Undefined.

Used in:

`<ude>`.

---

o-encoding

Original encoding - As stated in the Encoding section, all TMX documents are in Unicode. However, it is sometimes useful to know what code set was used to encode text that was converted to Unicode for purposes of interchange. The `o-encoding` attribute specifies the original or preferred code set of the data of the element in case it is to be re-encoded in a non-Unicode code set.

Value description:

One of the [IANA] recommended "charset identifier", if possible.

Default value:

Undefined.

Used in:

`<header>`, `<tu>`, `<tuv>`, `<note>`, `<prop>`.

---

o-tmf

Original translation memory format - Specifies the format of the translation memory file from which the TMX document or segment thereof have been generated.

Value description:

Text.

Default value:

Undefined.

Used in:
`<header>`, `<tu>`, `<tuv>`.

---

pos

Position - Indicates whether an isolated tag `<it>` is a beginning or and ending tag.

Value description:
"`begin`" or "`end`".

Default value:
Undefined.

Used in:
`<it>`.

---

segtype

Segment type - Specifies the kind of segmentation used in the `<tu>` element. If a `<tu>` element does not have a `segtype` attribute specified, it uses the one defined in the `<header>` element.

The "`block`" value is used when the segment does not correspond to one of the other values, for example when you want to store a chapter composed of several paragraphs in a single `<tu>`.

```
<tu segtype="block">
 <prop type="x-sentbreak">$#$</prop>
 <tuv xml:lang="en"><seg>This is the first paragraph of a big section.$#$
This is the second paragraph.$#$This is the third.</seg></tuv>
</tu>
```

In the example above the property "`x-sentbreak`" defines the token used to indicate the separation between sentences within the block of text. You can therefore easily break down the segment into smaller units if needed. You can imagine many other ways to use this mechanism.

A TMX file can include sentence level segmentation for maximum portability, so it is recommended that you use such segmentation rather than a specific, proprietary method like the one above.

Value description:
"`block`", "`paragraph`", "`sentence`", or "`phrase`".

Default value:

Undefined.

Used in:

`<header>`, `<tu>`.

---

srclang

Source language - Specifies the language of the source text. In other words, the `<tuv>` holding the source segment will have its `xml:lang` attribute set to the same value as `srclang`. (except if `srclang` is set to "`*all*`"). If a `<tu>` element does not have a `srclang` attribute specified, it uses the one defined in the `<header>` element.

Value description:

A language code as described in the [RFC 3066], or the value "`*all*`" if any language can be used as the source language. Unlike the other TMX attributes, the values for `srclang` are not case-sensitive.

Default value:

Undefined.

Used in:

`<header>`, `<tu>`.

---

subst

Substitution text - Specifies an alternative string for the character defined in a given `<map/>` element.

Value description:

A text in ASCII. For example: `subst="(c)"` for the copyright sign.

Default value:

Undefined.

Used in:

`<map/>`.

---

tuid

Translation unit identifier - Specifies an identifier for the `<tu>` element. Its value is not defined by the standard (it could be unique or not, numeric or alphanumeric, etc.).

Value description:

Text without white spaces.

Default value:

Undefined.

Used in:

`<tu>`.

---

type

Type - Specifies the kind of data a `<prop>`, `<bpt>`, `<ph>`, `<hi>`, `<sub>` or `<it>` element represents.

"index" = Index marker "date" = Date "time" = Time "fnote" = Footnote "enote" = End-note "alt" = Alternate text "image" = Image "pb" = Page break "lb" = Line break "cb" = Column break "inset" = Inset

Value description:

Text. Depends on the element where the attribute is used.
The recommended values for the `type` attribute, when used in `<bpt>` and `<it>` are as follow (this list is not exhaustive):
- "`bold`" = Bold.
- "`color`" = Color change.
- "`dulined`" = Doubled-underlined.
- "`font`" = Font change.
- "`italic`" = Italic.
- "`link`" = Linked text.
- "`scap`" = Small caps.
- "`struct`" = XML/SGML structure.
- "`ulined`" = Underlined.

The recommended values for the `type` attribute, when used in `<ph>` are as follow (this list is not exhaustive):
- "`index`" = Index marker.
- "`date`" = Date.
- "`time`" = Time.
- "`fnote`" = Footnote.
- "`enote`" = End-note.
- "`alt`" = Alternate text.
- "`image`" = Image.
- "`pb`" = Page break.
- "`lb`" = Line break.
- "`cb`" = Column break.
- "`inset`" = Inset.

Default value:

Undefined.

Used in:

<prop>, <bpt>, <ph>, <hi>, <sub>, <it> .

---

unicode

Unicode code-point - Specifies the Unicode character value of a <map/> element. Its value must be a

Value description:

A valid Unicode value (including values in the Private Use areas) in hexadecimal format. For example: `unicode="#xF8FF"`.

Default value:

Undefined.

Used in:

<map/>.

---

usagecount

Usage count - Specifies the number of times a <tu> or the content of the <tuv> element has been accessed in the original TM environment.

Value description:

Number.

Default value:

Undefined.

Used in:

<tu>, <tuv>.

---

version

TMX version - The `version` attribute indicates the version of the TMX format to which the document conforms.

Value description:

Fixed text: the major version number, a period, and the minor version number. For example: `version="1.4"`.

Default value:

`"1.4"`

Used in:

`<tmx>`.

---

x

External matching - The `x` attribute is used to match inline elements `<bpt>`, `<it>`, `<ph>`, and `<hi>` between each `<tuv>` element of a given `<tu>` element. This mechanism facilitates the pairing of allied codes in source and target text, even if the order of code occurrence differs between the two because of the translation syntax. Note that an `<ept>` element is matched based on `x` attribute of its corresponding `<bpt>` element.

For example:

```
<seg>The <bpt i="1" x="1">{\b </bpt>black<ept i="1">}</ept>
 <bpt i="2" x="2">{\i </bpt>cat<ept i="2">}</ept> eats.</seg>

<seg>Le <bpt i="1" x="2">{\i </bpt>chat<ept i="1">}</ept>
 <bpt i="2" x="1">{\b </bpt>noir<ept i="2">}</ept> mange.</seg>
```

Value description:

Number.

Default value:

Undefined.

Used in:

`<bpt>`, `<it>`, `<ph>`, `<hi>`.

3.2.2. XML Namespace Attributes

---

xml:lang

Language - The `xml:lang` attribute specifies the locale of the text of a given element.

Value description:

A language code as described in the [RFC 3066]. This declared value is considered to apply to all elements within the content of the element where it is specified, unless overridden with another instance of the `xml:lang` attribute. Unlike the other TMX attributes, the values for `xml:lang` are not case-sensitive. For more information see the section on `xml:lang` in the XML specification, and the erratum E11 (which replaces RFC 1766 by RFC 3066).

Default value:

Undefined.

Used in:

<tuv>, <note>, <prop>.

4. Content Markup

4.1. Overview

Each TM system uses a different method of marking up the formatting. Formats are constantly evolving, and new formats will be introduced on a regular basis. Attempting to collect, interpret, disseminate and maintain finite descriptions of each formatting tag used at any given time by any of the TM systems is not possible.

The best way to deal with these native codes is to delimit them by a specific set of elements that convey where they begin and end, and possibly additional information about what they are (bold, italic, footnote, etc.).

Native codes can be grouped into four categories:

1. Codes that either begin or end an instruction, and whose beginning and ending functions both appear within a single segment. For example, an instruction to begin embolden for a range of words which is then followed in the same segment by an instruction to end bold formatting.
2. Codes that either begin or end an instruction, but whose beginning and ending functions are not both contained within a single segment. For example, an instruction to embolden text may apply to the first three sentences in a paragraph, but the instruction to turn off bolding may only appear at the end of the third sentence. Its beginning instruction is present in the first segment, while its closing tag is present in the third segment.
3. Codes that represent self-contained functions that do not require explicit ending instructions. An image or cross-reference token are examples of these standalone codes, or codes that have unknown behavior.

Respectively, the TMX vocabulary provides elements to mark up each category of native code sequences:

1. The <bpt> and <ept> elements demark paired sequences of native code which begin and end in the same <seg> element.
2. The <it> element demarks a paired native code that is isolated from its partner, possibly due to segmentation.
3. The <ph> element demarks a standalone native code or a native code that cannot be identified.

An additional element, <sub>, is provided to delimit sub-flow text within a sequence of native codes. For instance, if the text content of a footnote is defined within the footnote marker code, it may be demarked with the <sub> element.

The <ut> element has been deprecated.

Examples:

```
Without Content mark-up tags:
Text in {\i italics}.

With Content mark-up tags (content markup in bold red):
Text in <bpt i="1" x="1" type="italic">{\i </bpt>italics<ept i="1">}</ept>.
```

Such a mechanism allows tools to perform matching at several levels:

1. **Ignoring the codes**. Since the code parts of the data stored in the segment are well delimited with the TMX elements, the tools can simply ignore them. It is equivalent to working at a plain text level. This solution may be the least efficient, but it does allow better matching results between units than when the codes get in the way.
2. **Matching the content mark-up tags**. A second and more efficient way to proceed is to recognize the TMX content mark-up elements and use them as part of the matching criteria. This additional step gives you better accuracy, and there is normally no need to know what the actual native codes (the content of the `<bpt>`, `<ept>`, `<it>`, and `<ph>` elements) are.

For example, here are four segments differing only by the formatting codes:

```
Plain text: Special text
    RTF v1: {\b Special} text
    RTF v2: {\cf7 Special} text
      HTML: <B>Special</B> text
```

The same samples with the TMX content mark-up tags:

```
Plain text: Special text
    RTF v1: <bpt i="1" x="1">{\b </bpt>Special<ept i="1">}</ept> text
    RTF v2: <bpt i="1" x="1">{\cf7 </bpt>Special<ept i="1">}</ept> text
      HTML: <bpt i="1" x="1">&lt;B></bpt>Special<ept i="1">&lt;/B></ept> text
```

- Native codes (RTF, HTML, etc.) need not be parsed before using the segments; the TMX elements allow you to make the distinction between code and text.
- Comparisons across file formats can be achieved, and leverage of native code of source into target can be performed. This not only leverages the translated text but also the correct formatting, even if it was originally in a different source file format.

4.2. Rules for Inlines Elements

The rules to use the `<bpt>`, `<ept>`, `<it>`, and `<ph>` elements are the following:

1. Use `<bpt>` for opening each code that has a corresponding closing code in the segment.

2. Use `<ept>` for closing each code that has a corresponding opening code in the segment.

3. Use <it> for opening or closing each code that has no corresponding closing or opening code in the segment.
   In some cases, because of the segmentation, you may have opening and closing codes that have no corresponding closing or opening codes within the same segment. Use <it> to encapsulate those codes. <it> has a mandatory attribute pos that should be set to "begin" or "end" depending on whether the isolated code is an opening or a closing code.

4. Use <ph> for standalone codes.
   Standalone codes are codes that are not opening or closing of a pair, for example empty elements in XML.

Examples:

```
The <bpt i="1" x="1">&lt;i></bpt><bpt i="2" x="2">&lt;b></bpt>
big<ept i="2">&lt;/b></ept> black<ept i="1">&lt;/i></ept> cat.

The icon <ph x="1">&lt;img src="testNode.gif"/></ph> represents
a conditional node.
```

4.3. Level 2 Implementation

TMX Level 2 is defined as follow:

- Assuming:
  - The segmentation is unambiguous. (For example, each paragraph is a sentence).
  - The translated segments does not have more or less tags than the source segments. (No additional tags was added, or removed).
  - The tags <hi> and <sub> are stripped out of all segments.
- Given:
  - An original document with inline codes (for example an HTML file) translated by a tool XYZ.
  - The translation memory of that document saved in TMX Level 2, using <bpt>, <ept>, <it>, and <ph> elements where appropriate. The use of <ut> is not allowed in compliant documents.

The tool XYZ supports TMX Level 2 Export if any tool compliant with TMX Level 2 Import is able to load the TMX document created by tool XYZ and re-create the translated document without loss of text or inline codes.

The tool XYZ supports TMX Level 2 Import if it any TMX Level 2 document created with a tool compliant with TMX Level 2 Export can be imported in tool XYZ and allow to re-create the translated document without loss of text or inline codes.

A tools that offers both import and export features must support both TMX Level 2 Import and TMX Level 2 Export to be TMX Level 2 compliant.

Verification of compliance can be done using a set of original documents and their corresponding TMX Level 2 files, and executing the following steps for each test file:

Import:

1. Import in the tool the model TMX file (`ImportTestN.tmx`) corresponding to a give original file (`ImportTestN.ext`).
2. Translate the original document (`ImportTestN.ext`) using the imported TM.
3. Compare the file translated by the tool with the model translated file (`ImportTestN_LANG.ext`).

Export:

1. Translate the original file (`ExportTestN.ext`) with the tool.
2. Export the corresponding TMX Level 2 document (e.g. `ExportTestNTool.tmx`).
3. Use TMXCheck to validate the tool's TMX document (`ExportTestNTool.tmx`) and compare it the with its corresponding model TMX file (`ExportTestN.tmx`).
   This comparison is done by:
   1. Verifying that for each source segment in the model TMX document there is an exact match source segment in the tool's TMX document.
      Two TMX Level 2 segments are identical if:
      - They have identical sets of character sequences, in the same order.
      - They have the same number of inline elements, in the same order and of the same kind (`<bpt>`, `<ept>`, etc.).
   2. Verifying that for each source segment matched in the tool's TM there is a target segment where inline codes are transferred correctly.
      Inline codes are transferred correctly if:
      - In the source segment, each element with a given attribute `x` (or the attribute `x` of its corresponding `<bpt>` in the case of an `<ept>` element) has a matching element in the target segment.

A Compliance Kit, that includes TMXCheck, a set of test files, and a detailed process document, is provided so anyone can verify the compliance of a given tool.

5. Miscellaneous

5.1. Grouping `<tu>` Elements

If you want to indicate that several `<tu>` elements belong to a logical group, you can specify a `<prop>` element for each of the `<tu>` which comprise the group.

```
<tu>
 <prop type="group">1</prop>
 <tuv xml:lang="en"><seg>First segment</seg><tuv>
 <tuv xml:lang="fr"><seg>Premier segment</seg><tuv>
</tu>
<tu>
 <prop type="group">1</prop>
 <tuv xml:lang="en"><seg>Second segment</seg><tuv>
 <tuv xml:lang="fr"><seg>Second segment</seg><tuv>
</tu>
```

TMX does not implement the notion of order. If the order of the `<tu>` elements is relevant, you may want to use the `tuid` attribute or a `<prop>` element to reflect it. For example:

```
<tu>
 <!-- Group 1, first item -->
 <prop type="group">1-1</prop>
 <tuv xml:lang="en"><seg>First segment</seg><tuv>
 <tuv xml:lang="fr"><seg>Premier segment</seg><tuv>
</tu>
<tu>
 <!-- Group 1, second item -->
 <prop type="group">1-2</prop>
 <tuv xml:lang="en"><seg>Second segment</seg><tuv>
 <tuv xml:lang="fr"><seg>Second segment</seg><tuv>
</tu>
```

Keeping track of how `<tu>` or `<tuv>` elements are grouped is not part of TMX's original design.

A. Sample Document

Notational conventions: The restrictions on the number of occurrences of each element and whether an attribute is mandatory within an element are indicated by:

- **`BOLD`** for the items that are mandatory.
- *`ITALIC`* for the items that can be specified zero or one times.
- `NORMAL` for the items that can be specified zero, one or more times.

In this example of a TMX document the indentations are only there for ease of reading, and the different types of notation are mixed to illustrate the various possibilities.

```
<?xml version="1.0"?>
<!-- Example of TMX document -->
<tmx version="1.4">
 <header
  creationtool="XYZTool"
  creationtoolversion="1.01-023"
  datatype="PlainText"
  segtype="sentence"
  adminlang="en-us"
  srclang="EN"
  o-tmf="ABCTransMem"
  creationdate="20020101T163812Z"
  creationid="ThomasJ"
  changedate="20020413T023401Z"
  changeid="Amity"
  o-encoding="iso-8859-1"
 >
  <note>This is a note at document level.</note>
  <prop type="RTFPreamble">{\rtf1\ansi\tag etc...{\fonttbl}</prop>
  <ude name="MacRoman" base="Macintosh">
   <map unicode="#xF8FF" code="#xF0" ent="Apple_logo" subst="[Apple]"/>
  </ude>
 </header>
 <body>
  <tu
   tuid="0001"
   datatype="Text"
   usagecount="2"
   lastusagedate="19970314T023401Z"
  >
   <note>Text of a note at the TU level.</note>
   <prop type="x-Domain">Computing</prop>
   <prop type="x-Project">P&#x00E6;gasus</prop>
   <tuv
    xml:lang="EN"
    creationdate="19970212T153400Z"
    creationid="BobW"
   >
    <seg>data (with a non-standard character: &#xF8FF;).</seg>
   </tuv>
   <tuv
    xml:lang="FR-CA"
    creationdate="19970309T021145Z"
    creationid="BobW"
    changedate="19970314T023401Z"
    changeid="ManonD"
   >
    <prop type="Origin">MT</prop>
    <seg>donn&#xE9;es (avec un caract&#xE8;re non standard: &#xF8FF;).</seg>
   </tuv>
  </tu>
  <tu
   tuid="0002"
   srclang="*all*"
  >
   <prop type="Domain">Cooking</prop>
   <tuv xml:lang="EN">
    <seg>menu</seg>
   </tuv>
   <tuv xml:lang="FR-CA">
    <seg>menu</seg>
   </tuv>
   <tuv xml:lang="FR-FR">
    <seg>menu</seg>
   </tuv>
  </tu>
 </body>
</tmx>
```

B. Document Type Definition for TMX

The document type definition file for TMX is available at: http://www.lisa.org/tmx/tmx14.dtd.

C. Changes Since Previous Version (Non-Normative)

The changes in this version (1.4) relative to the previous version (1.3) are as follows:

- Deprecated the `<ut>` element.
- Made the attribute `x` required instead of optional.
- Re-defined the TMX Levels.
- Re-formatted the whole document with style sheet.
- Revised the Content Markup section.
- Added several sections such as Abstract, General Structure, Miscellaneous, Level 2 Implementation, etc.
- Updated the References and Glossary sections.
- Updated all the examples.
- Integrated the Implementation Notes file with each relevant element or attribute, or in the new Miscellaneous section.
- Added requirement for uniqueness of the value for the `i` attributes in `<bpt>` elements within each `<seg>` element.

Additional changes for version 1.4a:

- Reversed the attribute `x` to optional.

D. Glossary

DTD
An SGML document has an associated Document Type Definition (DTD) that specifies the rules for the structure of the document. Several industries have standardized on various DTDs for the different types of documents that they share.

OSCAR
LISA special interest group (Open Standards for Container/Content Allowing Re-use).

SGML
SGML stands for Standard Generalized Markup Language. An ISO standard (ISO-8879) allows the definition of structured formats. SGML is not a format by itself, but a set of rules to define formats. SGML mark-up systems are defined in Document Type Definition files (DTDs).

UTC
UTC stands for Coordinated Universal Time.

XML
XML stands for Extensible Markup Language. XML is a simplified and restricted subset of SGML.

E. References

Normative

[IANA Charsets]
IANA Names for Character Sets. IANA (Internet Assigned Numbers Authority), Aug 2001

[ISO 639]
Codes for the Representation of Names of Languages. ISO (International Standards Organization), Nov 2001.

[ISO 3166]
Codes for the representation of names of countries and their subdivisions. ISO (International Organization for Standardization), Jun 2000.

[ISO 8601]
Representation of dates and times. ISO (International Organization for Standardization), Dec 2000.

[RFC 3066]
RFC 3066 Tags for the Identification of Languages. IETF (Internet Engineering Task Force), Jan 2001.

[XML 1.0]
Extensible Markup Language (XML) 1.0 Second Edition. W3C (World Wide Web Consortium), Oct 2000.

Non-Normative

[ISO]
International Organization for Standardization Web site.

[LISA]
Localisation Industry Standards Association Web site.

[Unicode]
Unicode Consortium Web site.

[W3C]
World Wide Web Consortium Web site.

-end-