XLIFF Version 2.2

Bryan Schnabel
Individual

<bryan.s.schnabel@tektronix.com>

Edited by David Filip and Rodolfo Raya

$Id: spectools-docbook-template-wd03.xml,v 1.4 2010/07/08 12:28:15 admin Exp$

Standards Track Work Product

http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/xliff-core-v2.2-csprd.html

http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/xliff-core-v2.2-csprd.pdf

http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/xliff-core-v2.2-csprd.xml

http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd01/xliff-core-v2.2-csprd01.html

http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd01/xliff-core-v2.2-csprd01.pdf

http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd01/xliff-core-v2.2-csprd01.xml

http://docs.oasis-open.org/xliff/xliff-core/v2.2/xliff-core-v2.2.html

http://docs.oasis-open.org/xliff/xliff-core/v2.2/xliff-core-v2.2.pdf

http://docs.oasis-open.org/xliff/xliff-core/v2.2/xliff-core-v2.2.xml

OASIS XML Localisation Interchange File Format (XLIFF) TC
[https://www.oasis-open.org/committees/xliff/]

Additional artifacts

This prose specification is one component of a Work Product that also includes:

- Declarative validation artifacts accessible from http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/

Related Work

This specification replaces or supersedes:

- *XLIFF Version 2.0.* Edited by Tom Comerford, David Filip, Rodolfo M. Raya, and Yves Savourel. 05 August 2014. OASIS Standard. http://docs.oasis-open.org/xliff/xliff-core/v2.0/os/xliff-core-v2.0-os.html

Declared XML Namespaces

- urn:oasis:names:tc:xliff:document:2.0
- urn:oasis:names:tc:xliff:matches:2.0
- urn:oasis:names:tc:xliff:glossary:2.0
- urn:oasis:names:tc:xliff:fs:2.0
- urn:oasis:names:tc:xliff:metadata:2.0
- urn:oasis:names:tc:xliff:resourcedata:2.0
- urn:oasis:names:tc:xliff:sizerestriction:2.0
- urn:oasis:names:tc:xliff:validation:2.0
- http://www.w3.org/2005/11/its

- urn:oasis:names:tc:xliff:itsm:2.

Status

This document was last revised or approved by the OASIS XML Localisation Interchange File Format (XLIFF) TC on the above date. The level of approval is also listed above. Check the “Latest version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment [https://www.oasis-open.org/committees/comments/index.php?wg_abbrev=xliff]" button on the Technical Committee's web page at https://www.oasis-open.org/committees/xliff/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (https://www.oasis-open.org/committees/xliff/ipr.php).

Note for any machine-readable content (aka Computer Language Definitions) declared Normative for this Work Product that is provided in separate plain text files, in the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format

When referencing this specification the following citation format should be used:

[XLIFF-2.2]

XLIFF Version 2.2. Edited by David Filip, Tom Comerford, Soroush Saadatfar, Felix Sasaki, and Yves Savourel. 18 April 2021. Specification Draft. http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/xliff-core-v2.2-csprd.html. Latest version: http://docs.oasis-open.org/xliff/xliff-core/v2.2/xliff-core-v2.2.html.

Notices

18 April 2021

Abstract

This document defines version 2.2 of the XML Localization Interchange File Format (XLIFF). The purpose of this vocabulary is to store localizable data and carry it from one step of the localization process to the other, while allowing interoperability between and among tools.

Table of Contents

Introduction

XLIFF is the *XML Localization Interchange File Format* designed by a group of multilingual content publishers, software providers, localization service providers, localization tools providers, and researchers. It is intended to give any multilingual content owner a single interchange file format that can be understood by any localization provider, using any conformant localization tool. While the primary focus is on being a lossless interchange format, usage of XLIFF as a processing format is neither encouraged nor discouraged or prohibited.

All text is normative unless otherwise labeled. The following common methods are used for labeling portions of this specification as informative and hence non-normative:

Appendices and sections marked as "(Informative)" or "Non-Normative" in title,
Notes (sections with the "Note" title),
Warnings (sections with the "Warning" title),
Examples (mainly example code listings, tree diagrams, but also any inline examples or illustrative exemplary lists in otherwise normative text),
Schema and other validation artifacts listings (the corresponding artifacts are normative, not their listings).

Terminology

Key words

The key words *must*, *must not*, *required*, *shall*, *shall not*, *should*, *should not*, *recommended*, *may*, and *optional* are to be interpreted as described in [RFC 2119].

Definitions

Agent — any application or tool that generates (creates), reads, edits, writes, processes, stores, renders or otherwise handles *XLIFF Documents*.

Agent is the most general application conformance target that subsumes all other specialized user agents disregarding whether they are defined in this specification or not.

Enrich, Enriching — the process of associating module and extension based metadata and resources with the *Extracted* XLIFF payload

Processing Requirements

- *Enriching may* happen at the time of *Extraction*.

Note

Extractor knowledge of the native format is not assumed while *Enriching*.

Enricher, Enricher Agent — any *Agent* that performs the *Enriching* process

Extract, Extraction — the process of encoding localizable content from a native content or User Interface format as XLIFF payload, so that localizable parts of the content in the source language are available for *Translation* into the target language along with the necessary context information

Extractor, Extractor Agent — any *Agent* that performs the *Extraction* process

Merge, Merging — the process of importing XLIFF payload back to the originating native format, based on the *full knowledge* of the *Extraction* mechanism, so that the localized content or User Interface strings replace the source language in the native format

Merger, Merger Agent

an *Agent* that performs the *Merge* process

> **Warning**
>
> Unless specified otherwise, any *Merger* is deemed to have the same knowledge of the native format as the *Extractor* throughout the specification.
>
> *Mergers* independent of *Extractors* can succeed, but it is out of scope of this specification to specify interoperability for *Merging* back without the full *Extractor* knowledge of the native format.

Modify, Modification

the process of changing core and module XLIFF structural and inline elements that were previously created by other *Writers*

Processing Requirements

- XLIFF elements *may* be *Modified* and *Enriched* at the same time.

> **Note**
>
> *Extractor* or *Enricher* knowledge of the native format is not assumed while *Modifying*.

Modifier, Modifier Agent

an *Agent* that performs the *Modification* process

Translation, Translate

a rendering of the meaning of the source text, expressed in the target language

Writer, Writer Agent

an *Agent* that creates, generates, or otherwise writes an *XLIFF Document* for whatever purpose, including but not limited to *Extractor*, *Modifier*, and *Enricher Agents*.

> **Note**
>
> Since XLIFF is intended as an exchange format rather than a processing format, many applications will need to generate *XLIFF Documents* from their internal processing formats, even in cases when they are processing *XLIFF Documents* created by another *Extractor*.

Key concepts

XLIFF Core

The core of XLIFF 2.2 consists of the minimum set of XML elements and attributes required to (a) prepare a document that contains text extracted from one or more files for localization, (b) allow it to be completed with the translation of the extracted text, and (c) allow the generation of *Translated* versions of the original document.

The XML namespace that corresponds to the core subset of XLIFF 2.2 is `"urn:oasis:names:tc:xliff:document:2.0"`.

XLIFF-defined (elements and attributes)	The following is the list of allowed schema URI prefixes for *XLIFF-defined* elements and attributes:
	
	`urn:oasis:names:tc:xliff:`
	`http://www.w3.org/2005/11/its`
	
	However, the following namespaces are NOT considered *XLIFF-defined* for the purposes of the XLIFF 2.2 specification:
	
	`urn:oasis:names:tc:xliff:document:1.0`
	`urn:oasis:names:tc:xliff:document:1.1`
	`urn:oasis:names:tc:xliff:document:1.2`
	`urn:oasis:names:tc:xliff:changetracking:2.0`
	
	Elements and attributes from other namespaces are not *XLIFF-defined*.

XLIFF Document	Any XML document that declares the namespace `"urn:oasis:names:tc:xliff:document:2.0"` as its main namespace, has `<xliff>` as the root element and complies with the XML Schemas and the declared Constraints that are part of this specification.
XLIFF Module	A module is an *optional* set of XML elements and attributes that stores information about a process applied to an *XLIFF Document* and the data incorporated into the document as result of that process. Each official module defined for XLIFF 2.2 has its grammar defined in an independent XML Schema with a separate namespace.

Normative References

[BCP 47] M. Davis, Tags for Identifying Languages, *http://tools.ietf.org/html/bcp47* IETF (Internet Engineering Task Force).

[HTML5] Ian Hickos, Robin Berjon, Steve Faulkner, Travis Leithead, Erika Doyle Navara, Edward O'Connor, Silvia Pfeiffer HTML5. A vocabulary and associated APIs for HTML and XHTML, *http://www.w3.org/TR/html5/* W3C Recommendation 28 October 2014.

[ITS] David Filip, Shaun McCance, Dave Lewis, Christian Lieske, Arle Lommel, Jirka Kosek, Felix Sasaki, Yves Savourel Internationalization Tag Set (ITS) Version 2.0, *http://www.w3.org/TR/its20/* W3C Recommendation 29 October 2013.

[NOTE-datetime] M. Wolf, C. Wicksteed, Date and Time Formats, *http://www.w3.org/TR/NOTE-datetime* W3C Note, 15th Setember 1997.

[NVDL] International Standards Organization, ISO/IEC 19757-4, Information Technology - Document Schema Definition Languages (DSDL) - Part 4: Namespace-based Validation Dispatching Language (NVDL), *http://standards.iso.org/ittf/PubliclyAvailableStandards/c038615_ISO_IEC_19757-4_2006(E).zip [http://standards.iso.org/ittf/PubliclyAvailableStandards/c038615_ISO_IEC_19757-4_2006(E).zip]* ISO, June 1, 2006.

[RFC 2119] S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, *https://www.ietf.org/rfc/rfc2119.txt* IETF (Internet Engineering Task Force) RFC 2119, March 1997.

[RFC 3987] M. Duerst and M. Suignard, Internationalized Resource Identifiers (IRIs), *https://www.ietf.org/rfc/rfc3987.txt* IETF (Internet Engineering Task Force) RFC 3987, January 2005.

[RFC 7303] H. Thompson and C. Lilley, XML Media Types, *https://www.tools.ietf.org/html/rfc7303 [https://www.tools.ietf.org/html/rfc7303]* IETF (Internet Engineering Task Force) RFC 7303, July 2014.

[Schematron] International Standards Organization, ISO/IEC 19757-3, Information Technology - Document Schema Definition Languages (DSDL) - Part 3: Rule-Based Validation — Schematron (Second Edition), *http://standards.iso.org/ittf/PubliclyAvailableStandards/c055982_ISO_IEC_19757-3_2016.zip [http://standards.iso.org/ittf/PubliclyAvailableStandards/c055982_ISO_IEC_19757-3_2016.zip]* ISO, January 15, 2016.

[UAX #9] M. Davis, A. Lanin, A. Glass, UNICODE BIDIRECTIONAL ALGORITHM, *http://www.unicode.org/reports/tr9/tr9-35.html* Unicode Bidirectional Algorithm, May 18, 2016.

[UAX #15] M. Davis, K. Whistler, UNICODE NORMALIZATION FORMS, *http://www.unicode.org/reports/tr15/tr15-44.html* Unicode Normalization Forms, February 24, 2016.

[Unicode] The Unicode Consortium, The Unicode Standard, *http://www.unicode.org/versions/Unicode9.0.0/* Mountain View, CA: The Unicode Consortium, June 21, 2016.

[XML] W3C, Extensible Markup Language (XML) 1.0, *http://www.w3.org/TR/xml/* (Fifth Edition) W3C Recommendation 26 November 2008.

[XML namespace] W3C, Schema document for namespace http://www.w3.org/XML/1998/namespace *http://www.w3.org/2001/xml.xsd* [http://www.w3.org/2009/01/xml.xsd]. at http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/informativeCopiesOf3rdPartySchemas/w3c/xml.xsd in this distribution

[XML Catalogs] Norman Walsh, XML Catalogs, *https://www.oasis-open.org/committees/download.php/14809/xml-catalogs.html* OASIS Standard V1.1, 07 October 2005.

[XML Schema] W3C, XML Schema, *refers to the two part standard comprising [XML Schema Structures] and [XML Schema Datatypes]* (Second Editions) W3C Recommendations 28 October 2004.

[XML Schema Datatypes] W3C, XML Schema Part 2: Datatypes, *http://www.w3.org/TR/xmlschema-2/* (Second Edition) W3C Recommendation 28 October 2004.

[XML Schema Structures] W3C, XML Schema Part 1: Structures, *https://www.w3.org/TR/xmlschema-1/* (Second Edition) W3C Recommendation 28 October 2004.

Non-Normative References

[LDML] Unicode Locale Data Markup Language *http://unicode.org/reports/tr35/*

[SRX] Segmentation Rules eXchange *http://www.unicode.org/uli/pas/srx/*

[UAX #29] M. Davis, UNICODE TEXT SEGMENTATION, *http://www.unicode.org/reports/tr29/* Unicode text Segmentation.

[XML I18N BP] Best Practices for XML Internationalization, 13 February 2008, *http://www.w3.org/TR/xml-i18n-bp/* W3C Working Group.

Conformance

1. ## Document Conformance

   a. XLIFF is an XML vocabulary, therefore conformant *XLIFF Documents must* be well formed and valid [XML] documents.

   b. Conformant *XLIFF Documents must* be valid instances of the official Core XML Schema (http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/xliff_core_2.0.xsd) that is a part of this multipart Work Product.

   c. As not all aspects of the XLIFF specification can be expressed in terms of XML Schemas, conformant *XLIFF Documents must* also comply with all relevant elements and attributes definitions, normative usage descriptions, and Constraints specified in this specification document.

   d. *XLIFF Documents may* contain custom extensions, as defined in the Extension Mechanisms section.

2. ## Application Conformance

   a. XLIFF *Writers must* create conformant *XLIFF Documents* to be considered XLIFF compliant.

   b. *Agents* processing conformant *XLIFF Documents* that contain custom extensions are not *required* to understand and process non-XLIFF elements or attributes. However, conformant applications *should* preserve existing custom extensions when processing conformant *XLIFF Documents*, provided that the elements that contain custom extensions are not removed according to XLIFF Processing Requirements or the extension's own processing requirements.

   c. All *Agents must* comply with Processing Requirements for otherwise unspecified *Agents* or without a specifically set target *Agent*.

   d. Specialized *Agents* defined in this specification - this is *Extractor*, *Merger*, *Writer*, *Modifier*, and *Enricher Agents* - *must* comply with the Processing Requirements targeting their specifically defined type of *Agent* on top of Processing Requirements targeting all *Agents* as per point c. above.

   e. XLIFF is a format explicitly designed for exchanging data among various *Agents*. Thus, a conformant XLIFF application *must* be able to accept *XLIFF Documents* it had written after those *XLIFF Documents* were *Modified* or *Enriched* by a different application, provided that:

      i. The processed files are conformant *XLIFF Documents*,

      ii. in a state compliant with all relevant Processing Requirements.

3. ## Backwards Compatibility

   a. Conformant applications are *required* to support XLIFF 2.0.

b. Conformant applications are NOT *required* to support XLIFF 1.2 or previous Versions.

Note

XLIFF Documents conformant to this specification are not and cannot be conformant to XLIFF 1.2 or earlier versions. If an application needs to support for whatever business reason both XLIFF 2 and XLIFF 1.2 or earlier, these will need to be supported as separate functionalities.

Fragment Identification

Because *XLIFF Documents* do not follow the usual behavior of XML documents when it comes to element identifiers, this specification defines how *Agents must* interpret the fragment identifiers in IRIs pointing to *XLIFF Documents*.

Note

Note that some identifiers may change during the localization process. For example `<data>` elements may be re-grouped or not depending on how tools treat identical original data.

Constraints

- A fragment identifier *must* match the following format:

```
<expression>      ::= "#" ["/"] <selector>
                      {<selectorSeparator> <selector>}
<selector>        ::= [<prefix> <prefixSeparator>] <id>
<prefix>          ::= NMTOKEN
<id>              ::= NMTOKEN
<prefixSeparator> ::= "="
<selectorSeparator>  ::= "/"
```

- There *must not* be two identical prefixes in the expression.
- When used, the following selectors *must* be declared in this order: file selector, group selector and unit selector.
- The selectors for modules or extensions, `<note>`, `<segment>` or `<ignorable>` or source inline elements, target inline elements and `<data>` have the following constraints:
  - Only one of them *may* be used in the expression.
  - The one used *must* be the last selector of the expression.

Warning

Please note that due to the above Constraints, referencing fragments using third party namespaces within *Modules* or extensions (including but not limited to *XLIFF Core* or the Metadata Module) is not possible. This is to restrict the complexity of the fragment identification mechanism, as it would otherwise have potentially unlimited depth.

Selectors for Core Elements

- The prefix `f` indicates a `<file>` id and the value of that id is unique among all `<file>` `id` attribute values within the enclosing `<xliff>` element.
- The prefix `g` indicates a `<group>` id and the value of that id is unique among all `<group>` `id` attribute values within the enclosing `<file>` element.
- The prefix `u` indicates a `<unit>` id and the value of that id is unique among all `<unit>` `id` attribute values within the enclosing `<file>` element.
- The prefix `n` indicates a `<note>` id and the value of that id is unique among all `<note>` `id` attribute values within the immediate enclosing `<file>`, `<group>`, or `<unit>` element.
- The prefix `d` indicates a `<data>` id and the value of that id is unique among all `<data>` `id` attribute values within the enclosing `<unit>` element.
- The prefix `t` indicates an id for an inline element in the `<target>` element and the value of that id is unique within the enclosing `<unit>` element (with the exception of the matching inline elements in the `<source>`).
- No prefix indicates an id for a `<segment>` or an `<ignorable>` or an inline element in the `<source>` element and the value of that id is unique within the enclosing `<unit>` element (with the exception of the matching inline elements in the `<target>`).

Selectors for Modules and Extensions

A selector for a module or an extension uses a registered prefix and the value of that id is unique within the immediate enclosing `<file>`, `<group>` or `<unit>` element.

Constraints

- The prefix of a module or an extension *must* be an NMTOKEN longer than 1 character and *must* be defined in the module or extension specification.
- The prefix of a module or an extension *must* be registered with the XLIFF TC.
- A given module or extension namespace URI *must* be associated with a single prefix.
- A prefix *may* be associated with more than one namespace URI (to allow for example different versions of a given module or extension to use the same prefix).

See also the constraints related to how IDs need to be specified in extensions (which applies for modules as well).

Relative References

Fragment identifiers that do not start with a character `/` (U+002F) are relative to their location in the document, or to the document being processed.

Any unit, group or file selector missing to resolve the relative reference is obtained from the immediate enclosing unit, group or file elements.

Examples

Given the following XLIFF document:

```
<xliff xmlns="urn:oasis:names:tc:xliff:document:2.0" version="2.0"
    srcLang="en" trgLang="fr">
  <file id="f1">
    <notes>
      <note id="n1">note for file.</note>
    </notes>
    <unit id="u1">
      <my:elem xmlns:my="myNamespaceURI" id="x1">data</my:elem>
      <notes>
        <note id="n1">note for unit</note>
      </notes>
      <segment id="s1">
        <source><pc id="1">Hello <mrk id="m1" type="term">World</mrk>!</pc>
            </source>
        <target><pc id="1">Bonjour le <mrk id="m1" type="term">Monde</mrk>
            ! </pc></target>
      </segment>
    </unit>
  </file>
</xliff>
```

You can have the following fragment identifiers:

- `#f=f1/u=u1/1` refers to the element `<pc id="1">` of the source content of the element `<unit id="u1">`.
- `#f=f1/u=u1/t=1` refers to the element `<pc id="1">` of the target content of the element `<unit id="u1">`.
- `#f=f1/n=n1` refers to the element `<note id="n1">` of the element `<file id="f1">`.
- `#f=f1/u=u1/n=n1` refers to the element `<note id="n1">` of the element `<unit id="u1">`.
- `#f=f1/u=u1/s1` refers to the element `<segment id="s1">` of the element `<unit id="u1">`.
- Assuming the extension defined by the namespace URI `myNamespaceURI` has registered the prefix `myprefix`, the expression `#f=f1/u=u1/myprefix=x1` refers to the element `<my:element id="x1">` of the element `<unit id="u1">`.

The Core Specification

XLIFF is a bilingual document format designed for containing text that needs *Translation*, its corresponding translations and auxiliary data that makes the *Translation* process possible.

At creation time, an *XLIFF Document may* contain only text in the source language. Translations expressed in the target language *may* be added at a later time.

The root element of an *XLIFF Document* is `<xliff>`. It contains a collection of `<file>` elements. Typically, each `<file>` element contains a set of `<unit>` elements that contain the text to be translated in the `<source>` child of one or more `<segment>` elements. Translations are stored in the `<target>` child of each `<segment>` element.

General Processing Requirements

- An *Agent* processing a valid *XLIFF Document* that contains *XLIFF-defined* elements and attributes that it cannot handle *must* preserve those elements and attributes.
- An *Agent* processing a valid *XLIFF Document* that contains custom elements and attributes that it cannot handle *should* preserve those elements and attributes.

Elements

This section contains a description of all elements used in *XLIFF Core*.

Tree Structure

Legend:

1 = one
\+ = one or more
? = zero or one
* = zero or more

```
<xliff>
|
+---<file> +
  |
  +---<skeleton> ?
  | |
  | +---<other> *
  |
  +---<other> *
  |
  +---<notes> ?
  | |
  | +---<note> +
  |
  +---At least one of (<unit> OR <group>)
      | |
      | +---<unit>
      |   |
      |   +---<other> *
      |   |
      |   +---<notes> ?
      |   | |
      |   | +---<note> +
      |   |
      |   +---<originalData> ?
      |   | |
      |   | +---<data> +
      |   |
      |   +---At least one of (<segment> OR <ignorable>)
      |       | |
```

```
|       | +---<segment>
|       |   |
|       |   +---<source> 1
|       |   |
|       |   +---<target> ?
|       |
|       +---<ignorable>
|           |
|           +---<source> 1
|           |
|           +---<target> ?
|
+---<group>
    |
    +---<other> *
    |
    +---<notes> ?
    | |
    | +---<note> +
    |
    +---At least one of (<unit> OR <group>)
```

Structural Elements

The structural elements used in *XLIFF Core* are: `<xliff>`, `<file>`, `<skeleton>`, `<group>`, `<unit>`, `<segment>`, `<ignorable>`, `<notes>`, `<note>`, `<originalData>`, `<data>`, `<source>` and `<target>`.

xliff

Root element for XLIFF documents.

Contains:

- One or more `<file>` elements

Attributes:

- `version`, *required*
- `srcLang`, *required*
- `trgLang`, *optional*
- `xml:space`, *optional*
- attributes from other namespaces, *optional*

Constraints

- The `trgLang` attribute is *required* if and only if the *XLIFF Document* contains `<target>` elements that are children of `<segment>` or `<ignorable>`.
- The following *XLIFF Module* attributes are explicitly allowed by the wildcard `other`:

  - the `its:version`attribute from the namespace `http://www.w3.org/2005/11/its`, *optional*.

file

Container for localization material extracted from an entire single document, or another high level self contained logical node in a content structure that cannot be described in the terms of documents.

Note

Sub-document artifacts such as particular sheets, pages, chapters and similar are better mapped onto the `<group>` element. The `<file>` element is intended for the highest logical level. For instance a collection of papers would map to a single *XLIFF Document*, each paper will be represented with one `<file>` element, whereas chapters and subsections will map onto nested `<group>` elements.

Contains:

- Zero or one `<skeleton>` element followed by
- elements from other namespaces, *optional*
- Zero or one `<notes>` element followed by
- One or more `<unit>` or `<group>` elements in any order.

Attributes:

- `id`, *required*
- `canResegment`, *optional*
- `original`, *optional*
- `translate`, *optional*
- `srcDir`, *optional*
- `trgDir`, *optional*
- `xml:space`, *optional*
- attributes from other namespaces, *optional*

Constraints

- The following *XLIFF Module* elements are explicitly allowed by the wildcard `other`:

  - Zero or one `<mda:metadata>` elements
  - Zero or one `<res:resourceData>` element
  - Zero or one `<slr:profiles>` elements
  - Zero or one `<slr:data>` elements
  - Zero or one `<val:validation>` elements
  - Zero, one, or more `<its:provenanceRecords>` elements

- *Module* and Extension elements *may* be used in any order.

- The following *XLIFF Module* attributes are explicitly allowed by the wildcard `other`:

  - attributes from the namespace `urn:oasis:names:tc:xliff:fs:2.0`, *optional*, provided that the Constraints specified in the Format Style Module are met.
  - attributes from the namespace `urn:oasis:names:tc:xliff:sizerestriction:2.0`, *optional*, provided that the Constraints specified in the Size and Length Restriction Module are met.
  - attributes from the namespace `http://www.w3.org/2005/11/its`, *optional*, provided that the Constraints specified in the ITS Module are met.
  - attributes from the namespace `urn:oasis:names:tc:xliff:itsm:2.1`, *optional*, provided that the Constraints specified in the ITS Module are met.

skeleton

Container for non-translatable material pertaining to the parent `<file>` element.

Contains:

Either

- Non-translatable text
- elements from other namespaces

or

- is empty.

Attributes:

- `href`, *optional*

Constraints

- The attribute `href` is *required* if and only if the `<skeleton>` element is empty.

Processing Requirements

- *Modifiers* and *Enrichers* processing an *XLIFF Document* that contains a `<skeleton>` element *must not* change that element, its attributes, or its content.
- *Extractors* creating an *XLIFF Document* with a `<skeleton>` element *must* leave the `<skeleton>` element empty if and only if they specify the attribute `href`.

group

Provides a way to organize units into a structured hierarchy.

Note that this is especially useful for mirroring a source format's hierarchical structure.

Contains:

- elements from other namespaces, *optional*
- Zero or one `<notes>` element followed by
- Zero, one or more `<unit>` or `<group>` elements in any order.

Attributes:

- `id`, *required*
- `name`, *optional*
- `canResegment`, *optional*
- `translate`, *optional*
- `srcDir`, *optional*
- `trgDir`, *optional*
- `type`, *optional*
- `xml:space`, *optional*
- attributes from other namespaces, *optional*

Constraints

- The following *XLIFF Module* elements are explicitly allowed by the wildcard `other`:

- Zero or one `<mda:metadata>` elements
- Zero or one `<slr:data>` elements
- Zero or one `<val:validation>` elements
- Zero, one, or more `<its:provenanceRecords>` elements

- *Module* and Extension elements *may* be used in any order.

- The following *XLIFF Module* attributes are explicitly allowed by the wildcard `other`:

  - attributes from the namespace `urn:oasis:names:tc:xliff:fs:2.0`, *optional*, provided that the Constraints specified in the Format Style Module are met.
  - attributes from the namespace `urn:oasis:names:tc:xliff:sizerestriction:2.0`, *optional*, provided that the Constraints specified in the Size and Length Restriction Module are met.
  - attributes from the namespace `http://www.w3.org/2005/11/its`, *optional*, provided that the Constraints specified in the ITS Module are met.
  - attributes from the namespace `urn:oasis:names:tc:xliff:itsm:2.1`, *optional*, provided that the Constraints specified in the ITS Module are met.

unit

Static container for a dynamic structure of elements holding the extracted translatable source text, aligned with the *Translated* text.

Contains:

- elements from other namespaces, *optional*
- Zero or one `<notes>` elements followed by
- Zero or one `<originalData>` element followed by
- One or more `<segment>` or `<ignorable>` elements in any order.

Attributes:

- `id`, *required*
- `name`, *optional*
- `canResegment`, *optional*
- `translate`, *optional*
- `srcDir`, *optional*
- `trgDir`, *optional*
- `xml:space`, *optional*
- `type`, *optional*
- attributes from other namespaces, *optional*

Constraints

- A `<unit>` *must* contain at least one `<segment>` element.

- The following *XLIFF Module* elements are explicitly allowed by the wildcard `other`:

  - Zero or one `<mtc:matches>` elements
  - Zero or one `<gls:glossary>` elements
  - Zero or one `<mda:metadata>` elements
  - Zero or one `<res:resourceData>` elements
  - Zero or one `<slr:data>` elements
  - Zero or one `<val:validation>` elements

- Zero, one, or more `<its:locQualityIssues>` elements
- Zero, one, or more `<its:provenanceRecords>` elements

- *Module* and Extension elements *may* be used in any order.
- The following *XLIFF Module* attributes are explicitly allowed by the wildcard `other`:

  - attributes from the namespace `urn:oasis:names:tc:xliff:fs:2.0`, *optional*, provided that the Constraints specified in the Format Style Module are met.
  - attributes from the namespace `urn:oasis:names:tc:xliff:sizerestriction:2.0`, *optional*, provided that the Constraints specified in the Size and Length Restriction Module are met.
  - attributes from the namespace `http://www.w3.org/2005/11/its`, *optional*, provided that the Constraints specified in the ITS Module are met.
  - attributes from the namespace `urn:oasis:names:tc:xliff:itsm:2.1`, *optional*, provided that the Constraints specified in the ITS Module are met.

segment

This element is a container to hold in its aligned pair of children elements the minimum portion of translatable source text and its *Translation* in the given Segmentation.

Contains:

- One `<source>` element followed by
- Zero or one `<target>` element

Attributes:

- `id`, *optional*
- `canResegment`, *optional*
- `state`, *optional*
- `subState`, *optional*

ignorable

Part of the extracted content that is not included in a segment (and therefore not translatable). For example tools can use `<ignorable>` to store the white space and/or codes that are between two segments.

Contains:

- One `<source>` element followed by
- Zero or one `<target>` element

Attributes:

- `id`, *optional*

notes

Collection of comments.

Contains:

- One or more `<note>` elements

note

This is an XLIFF specific way how to present end user readable comments and annotations. A note can contain information about `<source>`, `<target>`, `<unit>`, `<group>`, or `<file>` elements.

Contains:

- Text

Attributes:

- `id`, *optional*
- `appliesTo`, *optional*
- `category`, *optional*
- `priority`, *optional*
- attributes from other namespaces, *optional*

Constraints

- The following *XLIFF Module* attributes are explicitly allowed by the wildcard `other`:
  - `fs:fs`, *optional*
  - `fs:subFs`, *optional*

originalData

Unit-level collection of original data for the inline codes.

Contains:

- One or more `<data>` elements

data

Storage for the original data of an inline code.

Contains:

- Non-translatable text
- Zero, one or more `<cp>` elements.

Non-translatable text and `<cp>` elements *may* appear in any order.

Attributes:

- `id`, *required*
- `dir`, *optional*
- `xml:space`, *optional*, the value is restricted to `preserve` on this element

source

Portion of text to be translated.

Contains:

- Text
- Zero, one or more `<cp>` elements

- Zero, one or more <ph> elements
- Zero, one or more <pc> elements
- Zero, one or more <sc> elements
- Zero, one or more <ec> elements
- Zero, one or more <mrk> elements
- Zero, one or more <sm> elements
- Zero, one or more elements

Text and inline elements may appear in any order.

Attributes:

- xml:lang, *optional*
- xml:space, *optional*

Constraints

- When a <source> element is a child of or <ignorable>, the explicit or inherited value of the *optional* xml:lang attribute *must* be equal to the value of the srcLang attribute of the enclosing <xliff> element.

target

The translation of the sibling <source> element.

Contains:

- Text
- Zero, one or more <cp> elements
- Zero, one or more <ph> elements
- Zero, one or more <pc> elements
- Zero, one or more <sc> elements
- Zero, one or more <ec> elements
- Zero, one or more <mrk> elements
- Zero, one or more <sm> elements
- Zero, one or more elements

Text and inline elements may appear in any order.

Attributes:

- xml:lang, *optional*
- xml:space, *optional*
- order, *optional*

Constraints

- When a <target> element is a child of or <ignorable>, the explicit or inherited value of the *optional* xml:lang *must* be equal to the value of the trgLang attribute of the enclosing <xliff> element.

Inline Elements

The *XLIFF Core* inline elements at the <source> or <target> level are: <cp>, <ph>, <pc>, <sc>, <ec>, <mrk>, <sm> and .

The elements at the <unit> level directly related to inline elements are: <originalData> and <data>.

cp

Represents a Unicode character that is invalid in XML.

Contains:

This element is always empty.

Parents:

<data>, <mrk>, <source>, <target> and <pc>

Attributes:

- hex, *required*

Example:

```
<unit id="1">
  <segment>
    <source>Ctrl+C=<cp hex="0003"/></source>
  </segment>
</unit>
```

The example above shows a character U+0003 (Control C) as it has to be represented in XLIFF.

Processing Requirements

- *Writers must* encode all invalid XML characters of the content using <cp>.
- *Writers must not* encode valid XML characters of the content using <cp>.

ph

Represents a standalone code of the original format.

Contains:

This element is always empty.

Parents:

<source>, <target>, <pc> and <mrk>

Attributes:

- canCopy, *optional*
- canDelete, *optional*
- canReorder, *optional*
- copyOf, *optional*
- disp, *optional*
- equiv, *optional*
- id, *required.*
- dataRef, *optional*

- `subFlows`, *optional*
- `subType`, *optional*
- `type`, *optional*
- attributes from other namespaces, *optional*

Example:

```
<unit id="1">
  <originalData>
    <data id="d1">%d</data>
    <data id="d2">&lt;br/></data>
  </originalData>
  <segment>
    <source>Number of entries: <ph id="1" dataRef="d1" /><ph id="2"
        dataRef="d2"/>(These entries are only the ones matching the
        current filter settings)</source>
  </segment>
</unit>
```

Constraints

- The following *XLIFF Module* attributes are explicitly allowed by the wildcard `other`:

  - attributes from the namespace `urn:oasis:names:tc:xliff:fs:2.0`, *optional*, provided that the Constraints specified in the Format Style Module are met.
  - attributes from the namespace `urn:oasis:names:tc:xliff:sizerestriction:2.0`, *optional*, provided that the Constraints specified in the Size and Length Restriction Module are met.

- No other attributes *must* be used.

Processing Requirements

- *Extractors must not* use the `<ph>` element to represent spanning codes.

  Rationale: Using a standalone placeholder code for a spanning code does not allow for controlling the span (for instance tag order and data integrity) when *Modifying* inline content and is in *direct contradiction* to the business logic described in Representation of the codes and normative statements included in Usage of <pc> and <sc>/<ec>

> **Note**
>
> It is possible although not advised to use <ph> to mask non translatable inline content. The preferred way of protecting portions of inline content from translation is the *Core* Translate Annotation. See also discussion in the ITS Module section on representing translatability inline..

pc

Represents a well-formed spanning original code.

Contains:

- Text
- Zero, one or more `<cp>` elements
- Zero, one or more `<ph>` elements

- Zero, one or more `<pc>` elements
- Zero, one or more `<sc>` elements
- Zero, one or more `<ec>` elements
- Zero, one or more `<mrk>` elements
- Zero, one or more `<sm>` elements
- Zero, one or more `<em>` elements

Text and inline elements may appear in any order.

Parents:

- `<source>`
- `<target>`
- `<pc>`
-`<mrk>`

Attributes:

- `canCopy`, *optional*
- `canDelete`, *optional*
- `canOverlap`, *optional*
- `canReorder`, *optional*
- `copyOf`, *optional*
- `dispEnd`, *optional*
- `dispStart`, *optional*
- `equivEnd`, *optional*
- `equivStart`, *optional*
- `id`, *required*
- `dataRefEnd`, *optional*
- `dataRefStart`, *optional*
- `subFlowsEnd`, *optional*
- `subFlowsStart`, *optional*
- `subType`, *optional*
- `type`, *optional*
- `dir`, *optional*
- attributes from other namespaces, *optional*

Example:

```
<unit id="1">
  <originalData>
    <data id="1">&lt;B&gt;</data>
    <data id="2">&lt;/B&gt;</data>
  </originalData>
  <segment><pc id="1" dataRefStart="1" dataRefEnd="2">
      Important</pc> text</source></segment>
</unit>
```

Constraints

- The following *XLIFF Module* attributes are explicitly allowed by the wildcard `other`:

  - attributes from the namespace `urn:oasis:names:tc:xliff:fs:2.0`, *optional*, provided that the Constraints specified in the Format Style Module are met.

- attributes from the namespace `urn:oasis:names:tc:xliff:sizerestriction:2.0`, *optional*, provided that the Constraints specified in the Size and Length Restriction Module are met.

- No other attributes *must* be used.

Processing Requirements

- *Extractors must not* use the `<pc>` element to represent standalone codes.

  Rationale: Using a spanning code for a standalone code can easily result in having text inside a span where the original format does not allow it.

sc

Start of a spanning original code.

Contains:

This element is always empty.

Parents:

`<source>`, `<target>`, `<pc>` and `<mrk>`

Attributes:

- `canCopy`, *optional*
- `canDelete`, *optional*
- `canOverlap`, *optional*
- `canReorder`, *optional*
- `copyOf`, *optional*
- `dataRef`, *optional*
- `dir`, *optional*
- `disp`, *optional*
- `equiv`, *optional*
- `id`, *required*
- `isolated`, *optional*
- `subFlows`, *optional*
- `subType`, *optional*
- `type`, *optional*
- attributes from other namespaces, *optional*

Example:

```
<unit id="1">
  <segment>
    <source><sc id="1" type="fmt" subType="xlf:b"/>
        First sentence. </source>
  </segment>
  <segment>
    <source>Second sentence.<ec startRef="1" type="fmt"
        subType="xlf:b"/></source>
  </segment>
</unit>
```

Constraints

- The following *XLIFF Module* attributes are explicitly allowed by the wildcard `other`:

  - attributes from the namespace `urn:oasis:names:tc:xliff:fs:2.0`, *optional*, provided that the Constraints specified in the Format Style Module are met.
  - attributes from the namespace `urn:oasis:names:tc:xliff:sizerestriction:2.0`, *optional*, provided that the Constraints specified in the Size and Length Restriction Module are met.

- No other attributes *must* be used.
- The values of the attributes `canCopy`, `canDelete`, `canReorder` and `canOverlap` *must* be the same as the values the ones in the `<ec>` element corresponding to this start code.
- The attribute `isolated` *must* be set to `yes` if and only if the `<ec>` element corresponding to this start marker is not in the same `<unit>`, and set to `no` otherwise.

Processing Requirements

- *Extractors must not* use the `<sc>` / `<ec>` pair to represent standalone codes.

  Rationale: Using a spanning code for a standalone code can easily result in having text inside a span where the original format does not allow it.

ec

End of a spanning original code.

Contains:

This element is always empty.

Parents:

`<source>`, `<target>`, `<pc>` and `<mrk>`

Attributes:

- `canCopy`, *optional*
- `canDelete`, *optional*
- `canOverlap`, *optional*
- `canReorder`, *optional*
- `copyOf`, *optional*
- `dataRef`, *optional*
- `dir`, *optional*
- `disp`, *optional*
- `equiv`, *optional*
- `id`, *optional*
- `isolated`, *optional*
- `startRef`, *optional*
- `subFlows`, *optional*
- `subType`, *optional*
- `type`, *optional*
- attributes from other namespaces, *optional*

Example:

```
<unit id="1">
  <originalData>
    <data id="d1">\b </data>
    <data id="d2">\i </data>
    <data id="d3">\b0 </data>
    <data id="d4">\i0 </data>
  </originalData>
  <segment>
    <source>Text in <sc id="1" dataRef="d1"/>bold <sc id="2"
        dataRef="d2"/> and<ec startRef="1" dataRef="d3"/>
         italics<ec startRef="2" dataRef="d4"/>. </source>
  </segment>
</unit>
```

Constraints

- The following *XLIFF Module* attributes are explicitly allowed by the wildcard `other`:

  - attributes from the namespace `urn:oasis:names:tc:xliff:fs:2.0`, *optional*, provided that the Constraints specified in the Format Style Module are met.
  - attributes from the namespace `urn:oasis:names:tc:xliff:sizerestriction:2.0`, *optional*, provided that the Constraints specified in the Size and Length Restriction Module are met.

- No other attributes *must* be used.

- The values of the attributes `canCopy`, `canDelete` and `canOverlap` *must* be the same as the values the ones in the `<sc>` element corresponding to this end code.

- The value of the attribute `canReorder` *must* be `no` if the value of `canReorder` is `firstNo` in the `<sc>` element corresponding to this end code.

- The attribute `isolated` *must* be set to `yes` if and only if the `<sc>` element corresponding to this end code is not in the same `<unit>` and set to `no` otherwise.

- If and only if the attribute `isolated` is set to `yes`, the attribute `id` *must* be used instead of the attribute `startRef` that *must* be used otherwise.

- If and only if the attribute `isolated` is set to `yes`, the attribute `dir` *may* be used, otherwise the attribute `dir` *must not* be used on the `<ec>` element.

Processing Requirements

- *Extractors must not* use the `<sc>` / `<ec>` pair to represent standalone codes.

  Rationale: Using a spanning code for a standalone code can easily result in having text inside a span where the original format does not allow it.

mrk

Represents an annotation pertaining to the marked span.

Contains:

- Text

- Zero, one or more <cp> elements
- Zero, one or more <ph> elements
- Zero, one or more <pc> elements
- Zero, one or more <sc> elements
- Zero, one or more <ec> elements
- Zero, one or more <mrk> elements
- Zero, one or more <sm> elements
- Zero, one or more elements

Text and inline elements may appear in any order.

Parents:

<source>, <target>, <pc> and <mrk>

Attributes:

- id, *required*
- translate, *optional*
- type, *optional*
- ref, *optional*
- value, *optional*
- attributes from other namespaces, *optional*

Constraints

- The [XML namespace] *must not* be used at this extension point.
- The following *XLIFF Module* attributes are explicitly allowed by the wildcard other:

  - attributes from the namespace urn:oasis:names:tc:xliff:fs:2.0, *optional*, provided that the Constraints specified in the Format Style Module are met.
  - attributes from the namespace urn:oasis:names:tc:xliff:sizerestriction:2.0, *optional*, provided that the Constraints specified in the Size and Length Restriction Module are met.
  - attributes from the namespace http://www.w3.org/2005/11/its, *optional*, provided that the Constraints specified in the ITS Module are met.
  - attributes from the namespace urn:oasis:names:tc:xliff:itsm:2.1, *optional*, provided that the Constraints specified in the ITS Module are met.

See the Annotations section for more details and examples on how to use the <mrk> element.

sm

Start marker of an annotation where the spanning marker cannot be used for well-formedness reasons.

Contains:

This element is always empty.

Parents:

<source>, <target>, <pc> and <mrk>

Attributes:

- `id`, *required*
- `translate`, *optional*
- `type`, *optional*
- `ref`, *optional*
- `value`, *optional*
- attributes from other namespaces, *optional*

Constraints

- The [XML namespace] *must not* be used at this extension point.
- The following *XLIFF Module* attributes are explicitly allowed by the wildcard `other`:

  - attributes from the namespace `urn:oasis:names:tc:xliff:fs:2.0`, *optional*, provided that the Constraints specified in the Format Style Module are met.
  - attributes from the namespace `urn:oasis:names:tc:xliff:sizerestriction:2.0`, *optional*, provided that the Constraints specified in the Size and Length Restriction Module are met.
  - attributes from the namespace `http://www.w3.org/2005/11/its`, *optional*, provided that the Constraints specified in the ITS Module are met.
  - attributes from the namespace `urn:oasis:names:tc:xliff:itsm:2.1`, *optional*, provided that the Constraints specified in the ITS Module are met.

See the Annotations section for more details and examples on how to use the `<sm>` element.

em

End marker of an annotation where the spanning marker cannot be used for wellformedness reasons.

Contains:

This element is always empty.

Parents:

`<source>`, `<target>`, `<pc>` and `<mrk>`

Attributes:

- `startRef`, *required*

See the Annotations section for more details and examples on how to use the `<em>` element.

Attributes

This section lists all the various attributes used in XLIFF core elements.

XLIFF Attributes

The attributes defined in XLIFF 2.0 are: `appliesTo`, `canCopy`, `canDelete`, `canOverlap`, `canReorder`, `canResegment`, `category`, `copyOf`, `dataRef`, `dataRefEnd`, `dataRefStart`, `dir`, `disp`, `dispEnd`, `dispStart`, `equiv`, `equivEnd`, `equivStart`, `hex`, `href`, `id`, `isolated`, `name`, `order`, `original`, `priority`, `ref`, `srcDir`, `srcLang`, `startRef`, `state`,

`subFlows`, `subFlowsEnd`, `subFlowsStart`, `subState`, `subType`, `trgLang`, `translate`, `trgDir`, `type`, `value` and `version`.

appliesTo

Comment target - indicates the element to what the content of the note applies.

Value description: `source` or `target`.

Default value: undefined.

Used in: `<note>`.

canCopy

Replication editing hint - indicates whether or not the inline code can be copied.

Value description: `yes` if the code can be copied, `no` if the code is not intended to be copied.

Default value: `yes`.

Used in: `<pc>`, `<sc>`, `<ec>`, `<ph>`.

canDelete

Deletion editing hint - indicates whether or not the inline code can be deleted.

Value description: `yes` if the code can be deleted, `no` if the code is not allowed to be deleted.

Default value: `yes`.

Used in: `<pc>`, `<sc>`, `<ec>`, `<ph>`.

canOverlap

Code can overlap - indicates whether or not the spanning code where this attribute is used can enclose partial spanning codes (i.e. a start code without its corresponding end code, or an end code without its corresponding start code).

Value description: `yes` or `no`.

Default value: default values for this attribute depend on the element in which it is used:

- When used in `<pc>`: `no`.
- When used in `<sc>` or `<ec>`: `yes`.

Used in: `<pc>`, `<sc>` and `<ec>`

Example:

```
<unit id="1">
  <originalData>
    <data id="1">\i1 </data>
    <data id="2">\i0 </data>
    <data id="3">{\b </data>
```

```
    <data id="4">}</data>
  </originalData>
  <segment>
    <source><pc id="1" dataRefStart="3" dataRefEnd="4" canOverlap="no">
        Bold, <sc id="2" dataRef="1" canOverlap="yes"/>both</pc>,
         italics<ec startRef="2" dataRef="2"/></source>
  </segment>
</unit>
```

canReorder

Re-ordering editing hint - indicates whether or not the inline code can be re-ordered. See Editing Hints section for more details.

Value description: `yes` in case the code can be re-ordered, `firstNo` when the code is the first element of a sequence that cannot be re-ordered, `no` when it is another element of such a sequence.

Default value: `yes`.

Used in: `<pc>`, `<sc>`, `<ec>`, `<ph>`.

For the normative Usage Description see Constraints and Processing Requirements in the Editing Hints section.

canResegment

Can resegment - indicates whether or not the source text in the scope of the given `canResegment` flag can be reorganized into a different structure of `<segment>` elements within the same parent `<unit>`.

Value description: `yes` or `no`.

Default value: default values for this attribute depend on the element in which it is used:

- When used in `<file>`:

  The value `yes`.

- When used in any other element:

  The value of the `canResegment` attribute of its parent element.

Used in: `<file>` `<group>` `<unit>`, and `<segment>`.

category

Category - provides a way to categorize notes.

Value description: Text.

Default value: undefined

Used in: `<note>`.

copyOf

Reference to base code - holds the `id` of the base code of a copied code.

Value description: NMTOKEN. The `id` value of the base code of which this code is a copy.

Default value: undefined

Used in: `<ph>`, `<pc>`, `<sc>`, `<ec>`.

Example:

```
<unit id="1">
  <segment>
    <source>Äter <pc id="1">katter möss</pc>?</source>
    <target>Do <pc id="1">cats</pc> eat <pc id="2" copyOf="1">
        mice</pc>? </target>
  </segment>
</unit>
```

dataRef

Original data reference - holds the identifier of the `<data>` element that contains the original data for a given inline code.

Value description: An [XML Schema Datatypes] NMTOKEN that *must* be the value of the `id` attribute of one of the `<data>` element listed in the same `<unit>` element.

Default value: undefined.

Used in: `<ph>`, `<sc>`, `<ec>`.

Example:

```
<unit id="1">
  <originalData>
    <data id="d1">{0}</data>
  </originalData>
  <segment>
    <source>Error in '<ph id="1" dataRef="d1"/>'.</source>
    <target>Erreur dans '<ph id="1" dataRef="d1"/>'.</target>
  </segment>
</unit>
```

The example above shows a `<ph>` element that has its original data stored outside the content, in a `<data>` element.

dataRefEnd

Original data reference - holds the identifier of the `<data>` element that contains the original data for the end marker of a given inline code.

Value description: An [XML Schema Datatypes] NMTOKEN that *must* be the value of the `id` attribute of one of the `<data>` element listed in the same `<unit>` element.

Default value: undefined.

Used in: `<pc>`.

Example:

```
<unit id="1">
  <originalData>
    <data id="d1">&lt;EM></data>
    <data id="d2">&lt;/EM></data>
  </originalData>
  <segment>
    <source><pc id="1" dataRefStart="d1" dataRefEnd="d2">
        Efficiency</pc> is the operative word here.</source>
    <target><pc id="1" dataRefStart="d1" dataRefEnd="d2">
        Efficacité</pc> est le mot clé ici.</target>
  </segment>
</unit>
```

The example above shows two `<pc>` elements with their original data stored outside the content, in two `<data>` elements.

dataRefStart

Original data reference - holds the identifier of the `<data>` element that contains the original data for the start marker of a given inline code.

Value description: An [XML Schema Datatypes] NMTOKEN that *must* be the value of the `id` attribute of one of the `<data>` element listed in the same `<unit>` element.

Default value: undefined.

Used in: `<pc>`.

Example:

```
<unit id="1">
  <originalData>
    <data id="d1">&lt;EM></data>
    <data id="d2">&lt;/EM></data>
  </originalData>
  <segment>
    <source><pc id="1" dataRefStart="d1" dataRefEnd="d2">
        Efficiency</pc> is the operative word here.</source>
    <target><pc id="1" dataRefStart="d1" dataRefEnd="d2">
        Efficacité</pc> est le mot clé ici.</target>
  </segment>
</unit>
```

The example above shows two `<pc>` elements with their original data stored outside the content, in two `<data>` elements.

dir

Directionality - indicates the directionality of content.

Value description: `ltr` (Left-To-Right), `rtl` (Right-To-Left), or `auto` (determined heuristically, based on the first strong directional character in scope, see [UAX #9]).

Default value: default values for this attribute depend on the element in which it is used:

- When used in a <pc>, <sc>, or <ec> element that has a <source> element as its parent:

  The value of the srcDir attribute of the <unit> element, in which the elements are located.

- When used in a <pc>, <sc>, or <ec> element that has a <target> element as its parent:

  The value of the trgDir attribute of the <unit> element, in which the elements are located.

- When used in a <pc>, <sc>, or <ec> element that has a <pc> element as its parent:

  The value of the dir attribute of the parent <pc> element.

- When used in <data>:

  The value auto.

Used in: <data>, <pc>, <sc>, and <ec>.

disp

Display text - holds an alternative user-friendly display representation of the original data of the inline code.

Value description: Text.

Default value: undefined

Used in: <ph>, <sc>, <ec>.

Example:

```
<unit id="1">
  <originalData>
    <data id="d1">{1}</data>
  </originalData>
  <segment>
    <source>Welcome back <ph id="1" disp="[UserName]" dataRef="d1"/>!
        </source>
  </segment>
</unit>
```

Note

To provide a plain text equivalent of the code, use the equiv attribute.

dispEnd

Display text - holds an alternative user-friendly display representation of the original data of the end marker of an inline code.

Value description: Text.

Default value: undefined

Used in: <pc>.

Example:

```
<unit id="1">
  <originalData>
    <data id="d1">\cf1\ul\b\f1\fs24 </data>
    <data id="d2">\cf0\ulnone\b0\f0\fs22 </data>
  </originalData>
  <segment>
    <source>Example of <pc id="1" dataRefStart="d1" dataRefEnd="d2"
        dispStart="&lt;span>" dispEnd="&lt;/span>">
        formatted text</pc>.</source>
  </segment>
</unit>
```

In the example above, the `dispStart` and `dispEnd` attributes provide a more user-friendly representation of the original formatting codes.

Note

To provide a plain text equivalent of the code, use the `equivEnd` attribute.

dispStart

Display text - holds an alternative user-friendly display representation of the original data of the start marker of an inline code.

Value description: Text.

Default value: undefined

Used in: <pc>.

Example:

```
<unit id="1">
  <originalData>
    <data id="d1">\cf1\ul\b\f1\fs24 </data>
    <data id="d2">\cf0\ulnone\b0\f0\fs22 </data>
  </originalData>
  <segment>
    <source>Example of <pc id="1" dataRefStart="d1" dataRefEnd="d2"
        dispStart="&lt;span>" dispEnd="&lt;/span>">
        formatted text</pc>.</source>
  </segment>
</unit>
```

In the example above, the `dispStart` and `dispEnd` attributes provide a more user-friendly representation of the original formatting codes.

Note

To provide a plain text equivalent of the code, use the `equivStart` attribute.

equiv

Equivalent text - holds a plain text representation of the original data of the inline code that can be used when generating a plain text representation of the content.

Value description: Text.

Default value: an empty string.

Used in: `<ph>`, `<sc>`, `<ec>`.

Example:

```
<unit id="1">
  <originalData>
    <data id="d1">&amp;</data>
  </originalData>
  <segment>
    <source>Open <ph id="1" equiv="" dataRef="d1"/>File</source>
  </segment>
</unit>
```

In this example the `equiv` attribute of the `<ph>` element is used to indicate that the original data of the code can be ignored in the text representation of the string. This could, for instance, help a spell-checker tool to process the content as "Open File".

Note

To provide a user-friendly representation, use the `disp` attribute.

equivEnd

Equivalent text - holds a plain text representation of the original data of the end marker of an inline code that can be used when generating a plain text representation of the content.

Value description: Text.

Default value: an empty string

Used in: `<pc>`.

Example:

```
<unit id="1">
  <originalData>
    <data id="d1">&lt;span class="link" onclick="linkTo('dbId5345')">
        </data>
    <data id="d2">&lt;/span></data>
  </originalData>
  <segment>
    <source>The jam made of <pc id="1" dataRefStart="d1" equivStart=""
        dataRefEnd="d2" equivEnd="">lingonberries</pc> is quite
        tasty.</source>
  </segment>
</unit>
```

Note

To provide a user-friendly representation, use the `dispEnd` attribute.

equivStart

Equivalent text - holds a plain text representation of the original data of the start marker of an inline code that can be used when generating a plain text representation of the content.

Value description: Text.

Default value: an empty string

Used in: `<pc>`.

Example:

```
<unit id="1">
  <originalData>
    <data id="d1">&lt;span class="link" onclick="linkTo('dbId5345')">
        </data>
    <data id="d2">&lt;/span></data>
  </originalData>
  <segment>
    <source>The jam made of <pc id="1" dataRefStart="d1" equivStart=""
        dataRefEnd="d2" equivEnd="">lingonberries</pc> is quite
        tasty.</source>
  </segment>
</unit>
```

Note

To provide a user-friendly representation, use the `dispStart` attribute.

hex

Hexadecimal code point - holds the value of a Unicode code point that is invalid in XML.

Value description: A canonical representation of the hexBinary [XML Schema Datatypes] data type: Two hexadecimal digits to represent each octet of the Unicode code point. The allowed values are any of the values representing code points invalid in XML, between hexadecimal 0000 and 10FFFF (both included).

Default value: undefined

Used in: `<cp>`.

Example:

```
<cp hex="001A"/><cp hex="0003"/>
```

The example above shows a character U+001A and a character U+0003 as they have to be represented in XLIFF.

href

href - a pointer to the location of an external skeleton file pertaining to the enclosing `<file>` element..

Value description: IRI.

Default value: undefined

Used in: `<skeleton>`.

id

Identifier - a character string used to identify an element.

Value description: NMTOKEN. The scope of the values for this attribute depends on the element, in which it is used.

- When used in a `<file>` element:

  The value *must* be unique among all `<file>` `id` attribute values within the enclosing `<xliff>` element.

- When used in `<group>` elements:

  The value *must* be unique among all `<group>` `id` attribute values within the enclosing `<file>` element.

- When used in `<unit>` elements:

  The value *must* be unique among all `<unit>` `id` attribute values within the enclosing `<file>` element.

- When used in `<note>` elements:

  The value *must* be unique among all `<note>` `id` attribute values within the immediate enclosing `<file>`, `<group>`, or `<unit>` element.

- When used in `<data>` elements:

  The value *must* be unique among all `<data>` `id` attribute values within the enclosing `<unit>` element.

- When used in `<segment>`, `<ignorable>`, `<mrk>`, `<sm>`, `<pc>`, `<sc>`, `<ec>`, or `<ph>` elements:

  - The inline elements enclosed by a `<target>` element *must* use the duplicate `id` values of their corresponding inline elements enclosed within the sibling `<source>` element if and only if those corresponding elements exist.
  - Except for the above exception, the value *must* be unique among all of the above within the enclosing `<unit>` element.

Note

All of the above defined uniqueness scopes ignore *Module* and *Extension* data. It would be impossible to impose those uniqueness requirements onto *Module* or *Extension* data. As *Core* only *Modifiers* could inadvertently cause conflicts with *Modules*

or *Extensions* based data they cannot access. *Modules* and *Extensions* reusing *Core* need to specify their own uniqueness scopes for the xlf:id. In general, *Modules* and *Extensions* are advised to mimic the *Core* uniqueness requirement within their specific wrapper elements enclosing the reused *Core* elements or attributes, yet *Module* or *Extensions* are free to set wider uniqueness scopes if it makes business sense.

Default value: undefined

Used in: `<file>`, `<group>`, `<unit>`, `<note>`, `<segment>`, `<ignorable>`, `<data>`, `<sc>`, `<ec>`, `<ph>`, `<pc>`, `<mrk>` and `<sm>`.

isolated

Orphan code flag - indicates if the start or end marker of a spanning inline code is not in the same `<unit>` as its corresponding end or start code.

Value description: `yes` if this start or end code is not in the same `<unit>` as its corresponding end or start code, `no` if both codes are in the same `<unit>`.

Default value: `no`.

Used in: `<sc>`, `<ec>`.

Example:

```
<file id="f2" xmlns:abc="urn:abc">
  <unit id="1">
    <mtc:matches>
      <mtc:match id="tc01" ref="seg2">
        <source><sc id="1" isolated="yes"/>Warning:</source>
        <target><sc id="1" isolated="yes"/>Attention :</target>
      </mtc:match>
    </mtc:matches>
    <segment id="seg2">
      <source><pc id="1">Warning: File not found.</pc></source>
    </segment>
  </unit>
</file>
```

In the example above the `<sc>` elements have their `isolated` attribute set to `yes` because they do not have their corresponding `<ec>` elements.

name

Resource name - the original identifier of the resource corresponding to the *Extracted* `<unit>` or `<group>`.

For example: the key in the key/value pair in a Java properties file, the ID of a string in a Windows string table, the index value of an entry in a database table, etc.

Value description: Text.

Default value: undefined.

Used in: `<unit>` and `<group>`.

order

target order - indicates the order, in which to compose the target content parts.

Value description: A positive integer.

Default value: implicit, see below

When order is not explicitly set, the `<target>` `order` corresponds to its sibling `<source>`, i.e. it is not being moved anywhere when composing target content of the enclosing `<unit>` and the implicit `order` value is of that position within the `<unit>`.

Used in: `<target>`.

Constraints

- The value of the `order` attribute *must* be unique within the enclosing `<unit>` element.
- The value of each of the `order` attributes used within a `<unit>` element *must not* be higher than N, where N is the number of all current `<segment>` and `<ignorable>` children of the said `<unit>` element.

See the Segments Order section for the normative usage description.

original

Original file - a pointer to the location of the original document from which the content of the enclosing `<file>` element is extracted.

Value description: IRI.

Default value: undefined

Used in: `<file>`.

priority

Priority - provides a way to prioritize notes.

Value description: Integer 1-10.

Default value: `1`

Used in: `<note>`.

Note

Please note that 1 is the highest priority that can be interpreted as an alert, e.g. an [ITS] Localization Note [http://www.w3.org/TR/its20/#locNote-datacat] of the type alert. The best practice is to use only one alert per an annotated element, and the full scale of 2-10 can be used for prioritizing notes of lesser importance than the alert.

ref

Reference - holds a reference for the associated annotation.

Value description: A value of the [XML Schema Datatypes] type anyURI. The semantics of the value depends on the type of annotation:

- When used in a term annotation, the URI value is referring to a resource providing information about the term.
- When used in a translation candidates annotation, the URI value is referring to an external resource providing information about the translation candidate.
- When used in a comment annotation, the value is referring to a `<note>` element within the same enclosing `<unit>`.
- When used in a custom annotation, the value is defined by each custom annotation.

Default value: undefined

Used in: `<mrk>` or `<sm>`.

Example:

```
<unit id="1">
  <segment>
    <source>The <pc id="1">ref</pc> attribute of a term
        annotation holds a <mrk id="m1" type="term"
        ref="http://dbpedia.org/page/Uniform_Resource_Identifier">
        URI</mrk> pointing to more information about the given
        term.</source>
  </segment>
</unit>
```

srcDir

Source directionality - indicates the directionality of the source content.

Value description: `ltr` (Left-To-Right), `rtl` (Right-To-Left), , or `auto` (determined heuristically, based on the first strong directional character in scope, see [UAX #9]).

Default value: default values for this attribute depend on the element in which it is used:

- When used in `<file>`:

  The value `auto`.

- When used in any other element:

  The value of the `srcDir` attribute of its parent element.

Used in: `<file>`, `<group>`, and `<unit>`.

srcLang

Source language - the code of the language, in which the text to be *Translated* is expressed.

Value description: A language code as described in [BCP 47].

Default value: undefined

Used in: `<xliff>`.

startRef

Start code or marker reference - The `id` of the `<sc>` element or the `<sm>` element a given `<ec>` element or `<em>` element corresponds.

Value description: NMTOKEN.

Default value: undefined

Used in: `<ec>`, `<em>`.

Example:

```
<unit id="1">
  <segment>
    <source><sc id="1"/>Bold, <sc id="2"/>both
        <ec startRef="1"/>, italics<ec startRef="2"/></source>
  </segment>
</unit>
```

state

State - indicates the state of the translation of a segment.

Value description: The value *must* be set to one of the following values:

`initial` - indicates the segment is in its initial state.
`translated` - indicates the segment has been translated.
`reviewed` - indicates the segment has been reviewed.
`final` - indicates the segment is finalized and ready to be used.

The 4 defined states constitute a simple linear state machine that advances in the above given order. No particular workflow or process is prescribed, except that the three states more advanced than the default `initial` assume the existence of a *Translation* within the segment. One can further specify the state of the *Translation* using the `subState` attribute.

Default value: `initial`

Used in: `<segment>`

Processing Requirements

- *Writers must not* set the `state` attribute values to other than the default `initial` if and only if the `<segment>` element where the attribute is set doesn't have the `<target>` child.
- *Writers* updating the attribute `state` *must* also update or delete `subState`.

Note

`state` is an *optional* attribute of segments with a default value and segmentation can change as the XLIFF roundtrip progresses, hence implementers don't have to make explicit use of the attribute. However setting of the attribute is advantageous if a workflow needs to make use of Advanced Validation methods. For instance

missing non-removable codes will only be reported as an Error by the XLIFF Core Schematron Schema when the `state` is `final`.

subFlows

Sub-flows list - holds a list of `id` attributes corresponding to the `<unit>` elements that contain the sub-flows for a given inline code.

Value description: A list of NMTOKEN values separated by spaces. Each value corresponds to the `id` attribute of a `<unit>` element.

Default value: undefined

Used in: `<ph>`, `<sc>`, `<ec>`.

Example:

See the example in the Sub-Flows section.

subFlowsEnd

Sub-flows list - holds a list of `id` attributes corresponding to the `<unit>` elements that contain the sub-flows for the end marker of a given inline code.

Value description: A list of NMTOKEN values separated by spaces. Each value corresponds to the `id` attribute of a `<unit>` element.

Default value: undefined

Used in: `<pc>`.

Example:

See the example in the Sub-Flows section.

subFlowsStart

Sub-flows list - holds a list of `id` attributes corresponding to the `<unit>` elements that contain the sub-flows for the start marker of a given inline code.

Value description: A list of NMTOKEN values separated by spaces. Each value corresponds to the `id` attribute of a `<unit>` element.

Default value: undefined

Used in: `<pc>`.

Example:

See the example in the Sub-Flows section.

subState

subState - indicates a user-defined status for the `<segment>` element.

Value description:

The value is composed of a prefix and a sub-value separated by a character `:` (U+003A).

The prefix is a string uniquely identifying a collection of values for a specific authority. The sub-value is any string value defined by an authority.

The prefix `xlf` is reserved for this specification.

Other prefixes and sub-values *may* be defined by the users.

Default value: undefined

Used in: `<segment>`

Constraints

- If the attribute `subState` is used, the attribute `state` *must* be explicitly set.

Processing Requirements

- *Writers* updating the attribute `state` *must* also update or delete `subState`.

subType

subType - indicates the secondary level type of an inline code.

Value description:

The value is composed of a prefix and a sub-value separated by a character `:` (U+003A).

The prefix is a string uniquely identifying a collection of sub-values for a specific authority. The sub-value is any string value defined by the authority.

The prefix `xlf` is reserved for this specification, and the following sub-values are defined:

`xlf:lb` - Line break
`xlf:pb` - Page break
`xlf:b` - Bold
`xlf:i` - Italics
`xlf:u` - Underlined
`xlf:var` - Variable

Other prefixes and sub-values *may* be defined by the users.

Default value: undefined

Used in: `<pc>`, `<sc>`, `<ec>` and `<ph>`

Constraints

- If the attribute `subType` is used, the attribute `type` *must* be specified as well.
- The reserved `xlf:` prefixed values map onto the `type` attribute values as follows:

  For `xlf:b`, `xlf:i`, `xlf:u`, `xlf:lb`, and `xlf:pb`, the *required* value of the `type` attribute is `fmt`.
  For `xlf:var`, the *required* value of the `type` attribute is `ui`.

Processing Requirements

- *Modifiers* updating the attribute `type` *must* also update or delete `subType`.

trgLang

Target language - the code of the language, in which the *Translated* text is expressed.

Value description: A language code as described in [BCP 47].

Default value: undefined

Used in: `<xliff>`.

translate

Translate - indicates whether or not the source text in the scope of the given `translate` flag is intended for *Translation*.

Value description: `yes` or `no`.

Default value: default values for this attribute depend on the element in which it is used:

- When used in `<file>`:

  The value `yes`.

- When used in any other admissible structural element (`<group>` or `<unit>`):

  The value of the `translate` attribute of its parent element.

- When used in annotations markers `<mrk>` or `<sm>`:

  The value of the `translate` attribute of the innermost `<mrk>` or `<unit>` element, in which the marker in question is located.

Used in: `<file>` `<group>` `<unit>`, `<mrk>` and `<sm>`.

trgDir

Target directionality - indicates the directionality of the target content.

Value description: `ltr` (Left-To-Right), `rtl` (Right-To-Left), or `auto` (determined heuristically, based on the first strong directional character in scope, see [UAX #9]).

Default value: default values for this attribute depend on the element in which it is used:

- When used in `<file>`:

  The value `auto`.

- When used in any other element:

  The value of the `trgDir` attribute of its parent element.

Used in: `<file>`, `<group>`, and `<unit>`.

type

Type - indicates the type of an element.

Value description: allowed values for this attribute depend on the element in which it is used.

- When used in `<pc>`, `<sc>`, `<ec>` or `<ph>`:

  The value *must* be set to one of the following values:

  `fmt` - Formatting (e.g. a <b> element in HTML)
  `ui` - User interface element
  `quote` - Inline quotation (as opposed to a block citation)
  `link` - Link (e.g. an <a> element in HTML)
  `image` - Image or graphic
  `other` - Type of element not covered by any of the other top-level types.

  Example:

```
<segment>
  <source xml:lang="cs"><pc type="quote">Blázen,
      chce dobýt to nu v takovém po asi</pc>, dodal slovy svého
      oblíbeného imaginárního autora.</source>
  <target xml:lang="en"><pc type="quote">Madman, he wants to conquer the
      pole in this weather</pc>, offered he the words of his
      favourite imaginary playwright.</target>
</segment>
```

  One can further specify the type of a code using the `subType` attribute.

  Default value: Undefined

- When used in `<mrk>` or `<sm>`:

  One of the following values: `generic`, `comment`, `term`, or a user-defined value that is composed of a prefix and a sub-value separated by a character `:` (U+003A).

  The prefix is a string uniquely identifying a collection of sub-values for a specific authority. The sub-value is any string value defined by the authority.

  Default value: `generic`

- When used in `<group>` or `<unit>`:

  A value that is composed of a prefix and a sub-value separated by a character `:` (U+003A).

  The prefix is a string uniquely identifying a collection of sub-values for a specific authority. The sub-value is any string value defined by the authority. The prefix `xlf` is reserved.

  Default value: Undefined

Used in: `<group>`, `<unit>`, `<pc>`, `<sc>`, `<ec>`, `<mrk>`, `<ph>` and `<sm>`.

Processing Requirements

- *Modifiers* updating the attribute `type` on `<pc>`, `<sc>`, `<ec>`, or `<ph>` *must* also update or delete `subType`.

value

Value - holds a value for the associated annotation.

Value description: Text.

- When used in a term annotation, the value is a definition of the term.
- When used in a comment annotation, the value is the text of the comment.
- When used in a custom annotation, the value is defined by each custom annotation.

Default value: undefined

Used in: `<mrk>` and `<sm>`.

version

XLIFF Version - is used to specify the Version of the *XLIFF Document*. This corresponds to the Version number of the XLIFF specification that the *XLIFF Document* adheres to.

Value description: Text.

Default value: undefined

Used in: `<xliff>`.

XML namespace

The attributes from XML namespace used in XLIFF 2.0 are: xml:lang and xml:space.

xml:lang

Language - the xml:lang attribute specifies the language variant of the text of a given element. For example: `xml:lang="fr-FR"` indicates the French language as spoken in France.

Value description: A language code as described in [BCP 47].

Default value: default values for this attribute depend on the element in which it is used:

- When used in a `<source>` element:

  The value set in the `srcLang` attribute of the enclosing `<xliff>` element.

- When used in a `<target>` element:

  The value set in the `trgLang` attribute of the enclosing `<xliff>` element.

- When used in any other element:

  The value of the `xml:lang` attribute of its parent element.

Used in: `<source>`, `<target>` and where extension attributes are allowed.

xml:space

White spaces - the xml:space attribute specifies how white spaces (ASCII spaces, tabs and line-breaks) are to be treated.

Value description: `default` or `preserve`. The value `default` signals that an application's default white-space processing modes are acceptable for this element; the value `preserve` indicates the intent that applications preserve all the white space. This declared

intent 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:space attribute. For more information see the section on xml:space [http://www.w3.org/TR/REC-xml/#sec-white-space] in the [XML] specification.

Default value: default values for this attribute depend on the element in which it is used:

- When used in `<data>`:

  The value `preserve`.

- When used in `<xliff>`:

  The value `default`.

- When used in any other element:

  The value of the `xml:space` attribute of its parent element.

Used in: `<xliff>`, `<file>`, `<group>`, `<unit>`, `<source>`, `<target>`, and `<data>`.

CDATA sections

CDATA sections (`<![CDATA[...]]>`) are allowed in XLIFF content, but on output they *may* be changed into normal escaped content.

Note that avoiding CDATA sections is considered a best practice from the internationalization viewpoint [XML I18N BP].

Processing Requirements

- *Agents must* process CDATA sections.
- *Writers may* preserve the original CDATA sections.

XML Comments

XML comments (`<!--...--!>`) are allowed in XLIFF content, but they are ignored in the parsed content.

For example:

```
<source>Text content <!--IMPORTANT-->that is important</source>
```

and

```
<source>Text content that is important</source>
```

are identical after parsing and correspond to the same following parsed content:

```
Text content that is important
```

To annotate a section of the content with a comment that is recognized and preserved by XLIFF user agents, use the `<note>` element, or the `<mrk>` element.

Processing Requirements

- *Agents must* ignore XML comments. That is the XLIFF parsed content is the same whether or not there is an XML comment in the document.
- *Writers may* preserve XML comments on output.

XML Processing Instructions

XML Processing Instructions [XML] (see specifically http://www.w3.org/TR/REC-xml/#sec-pi) are an XML mechanism to "allow documents to contain instructions for applications." XML Processing Instructions are allowed in XLIFF content but they are ignored in the parsed content in the same sense as XML Comments.

Processing Requirements

- *Agents must not* use Processing Instructions as a means to implement a feature already specified in *XLIFF Core* or *Modules*.
- *Writers should* preserve XML Processing Instructions in an *XLIFF Document*.

Warning

Please note that *Agents* using Processing Instructions to implement *XLIFF Core* or *Module* features are not compliant XLIFF applications disregarding whether they are otherwise conformant.

Warning

Although this specification encourages XLIFF *Agents* to preserve XML Processing Instructions, it is not and cannot be, for valid processing reasons, an absolute protection and it is for instance highly unlikely that Processing Instructions could survive an XLIFF roundtrip at the `<segment>` level or lower. Hence implementers are discouraged from using XML Processing Instructions at the `<segment>` and lower levels.

Inline Content

The XLIFF inline content defines how to encode the content *Extracted* from the original source. The content includes the following types of data:

- Text – Textual content.
- Inline codes – Sequences of content that are not linguistic text, such as formatting codes, variable placeholders, etc.

  For example: the element `<b>` in HTML, or the placeholder `{0}` in a Java string.
- Annotations – Markers that delimit a span of the content and carry or point to information about the specified content.

  For example: a flag indicating that a given section of text is not intended for translation, or an element indicating that a given expression in the text is a term associated with a definition.

There are two elements that contain inline markup in XLIFF: `<source>` and `<target>`.

In some cases, data directly associated with inline elements *may* also be stored at the `<unit>` level in an `<originalData>` element.

Text

The XLIFF inline markup does not prescribe how to represent normal text, besides that it *must* be valid XML.

Characters invalid in XML

Because the content represented in XLIFF can be extracted from anywhere, including software resources and other material that can contain control characters, XLIFF needs to be able to represent all Unicode code points [Unicode].

However, XML does not have the capability to represent all Unicode code points [Unicode], and does not provide any official mechanism to escape the forbidden code points.

To remedy this, the inline markup provides the `<cp>` element.

The syntax and semantic of `<cp>` in XLIFF are similar to the ones of `<cp>` in the Unicode Locale Data Markup Language [LDML].

Inline Codes

The specification takes into account two types of codes:

Original code
: An *original code* is a code that exists in the original document being extracted into XLIFF.

Added code
: An *added code* is a code that does not exist in the original document, but has been added to the content at some point after extraction.

Any code (original or added) belongs to one of the two following categories:

Standalone
: A *standalone* code is a code that corresponds to a single position in the content. An example of such code is the `<br/>` element in HTML.

Spanning
: A *spanning* code is a code that encloses a section of the content using a start and an end marker. There are two kinds of spanning codes:

  - Codes that can overlap, that is: they can enclose a non-closing or a non-opening spanning code. Such codes do not have an XML-like behavior. For example the RTF code `\b1...\b0` is a spanning code that is allowed to overlap.
  - Codes that cannot overlap, that is: they cannot enclose a partial spanning code and have an XML-like behavior at the same time. An example of such code is the `<emphasis>...</emphasis>` element in DocBook.

  When the opening or closing marker of a spanning code does not have its corresponding closing or opening marker in the same unit, it is an *orphan code*.

Representation of the codes

Spanning codes present a set of challenges in XLIFF:

First, because the code format of the original data extracted to XLIFF does not need to be XML, spanning codes can overlap.

For example, in the following RTF content, the format markers are in a sequence: start bold, start italics, end bold, end italics. This does not translate into a well-formed mapping.

```
Text in \b bold \i and\b0  italics\i0
```

Another challenge is the possible effect of segmentation: A spanning code can start in one segment and end in another.

For example, in the following HTML content, the segmentation splits the text independently of the codes so the starting and ending tags of the `<B>...</B>` element end up in different parts of the `<unit>` element:

```
[Sentence <B>one. ][Sentence two.][ ][Sentence</B> three.]
```

Finally, a third potential cause of complication is that the start or the end markers of a spanning code can become orphans if their segment is used outside of its original `<unit>`.

For example, an entry with bold text can be broken down into two segments:

```
Segment 1 = "<b>Warning found: "
Segment 2 = "The file is read-only</b>"
```

And later, one of the segments can be re-used outside its original `<unit>`, for instance as a translation candidate:

```
New segment = "<b>Warning found - see log</b>"
Fuzzy match = "<b>Warning found: "
```

Because of these use cases, the representation of a spanning code cannot always be mapped to a similar spanning element in XLIFF.

When taking into account these issues, the possible use cases and their corresponding XLIFF representations are as follow:

Table 1. Inline code use cases

Use Case	Example of Representation
Standalone code	`<ph id='1'/>`
Well-formed spanning code	`<pc id='1'>text</pc>`
Start marker of spanning code	`<sc id='1'/>`
End marker of spanning code	`<ec startRef='1'/>`
Orphan start marker of spanning code	`<sc id='1' isolated='yes'/>`
Orphan end marker of spanning code	`<ec id='1' isolated='yes'/>`

Usage of <pc> and <sc>/<ec>

A spanning code *must* be represented using a `<sc>` element and a `<ec>` element if the code is not well-formed or orphan.

For example, the following RTF content has two spans of formatting:

```
Text in \b bold \i and\b0  italics\i0
```

They can only be represented using two pairs of <sc> and <ec> elements:

```
<unit id="1">
  <originalData>
    <data id="d1">\b </data>
    <data id="d2">\i </data>
    <data id="d3">\b0 </data>
    <data id="d4">\i0 </data>
  </originalData>
  <segment>
    <source>Text in <sc id="1" dataRef="d1"/>bold <sc id="2"
      dataRef="d2"/> and<ec startRef="1" dataRef="d3"/>
       italics<ec startRef="2" dataRef="d4"/>. </source>
  </segment>
</unit>
```

If the spanning code is well-formed it *may* be represented using either a single <pc> element or using a pair of <sc> and a <ec> elements.

For example, the following RTF content has a single span of formatting:

```
Text in \b bold\b0 .
```

It can be represented using either notations:

```
Text in <pc id="1" canOverlap="yes" dataRefStart="c1" dataRefEnd="c2">
bold</pc>.
```

```
Text in <sc id="1" dataRef="c1"/>bold<ec startRef="1" dataRef="c2"/>.
```

Processing Requirements

- When both the <pc> and the <sc>/<ec> representations are possible, *Extractors* and *Modifiers may* use either one as long as all the information of the inline code (e.g. original data, sub-flow indicators, etc.) are preserved.
- When converting representation between a pair of <sc> and <ec> elements and a <pc> element or vice-versa, *Modifiers must* map their attributes as shown in the following table:

Table 2. Mapping between attributes

<pc> attributes	<sc> attributes	<ec> attributes
id	id	startRef / id (see <ec>)
type	type	type
subType	subType	subType
dispStart	disp	
dispEnd		disp
equivStart	equiv	

equivEnd		equiv
subFlowsStart	subFlows	
subFlowsEnd		subFlows
dataRefStart	dataRef	
dataRefEnd		dataRef
	isolated	isolated
canCopy	canCopy	canCopy
canDelete	canDelete	canDelete
canReorder	canReorder	canReorder
copyOf	copyOf	copyOf
canOverlap	canOverlap	canOverlap
dir	dir	dir

- *Agents must* be able to handle any of the above two types of inline code representation.

Storage of the original data

Most of the time, inline codes correspond to an original construct in the format from which the content was extracted. This is the *original data*.

XLIFF tries to abstract and normalize as much as possible the extracted content because this allows a better re-use of the material across projects. Some tools require access to the original data in order to create the translated document back into its original format. Others do not.

No storage of the original data

In this option, the original data of the inline code is not preserved inside the XLIFF document.

The tool that created the initial XLIFF document is responsible for providing a way to re-create the original format properly when merging back the content.

For example, for the following HTML content:

```
This <B>naked mole rat</B> is <B>pretty ugly</B>.
```

one possible XLIFF representation is the following:

```
<unit id="1">
  <segment>
    <source>This <pc id="1">naked mole rat</pc> is
        <pc id="2">pretty ugly</pc>.</source>
    <target>Cet <pc id="1">hétérocéphale</pc> est
        <pc id="2">plutôt laid</pc>.</target>
  </segment>
</unit>
```

Storage of the original data

In this option, the original data of the inline code is stored in a structure that resides outside the content (i.e. outside `<source>` or `<target>`) but still inside the `<unit>` element.

The structure is an element `<originalData>` that contains a list of `<data>` entries uniquely identified within the `<unit>` by an `id` attribute. In the content, each inline code using this mechanism includes a `dataRef` attribute that points to a `<data>` element where its corresponding original data is stored.

For example, for the following HTML content:

```
This <B>naked mole rat</B> is <B>pretty ugly</B>.
```

The following XLIFF representation stores the original data:

```
<unit id="1">
  <originalData>
    <data id="d1">&lt;B></data>
    <data id="d2">&lt;/B></data>
  </originalData>
  <segment>
    <source>This <pc id="1" dataRefStart="d1" dataRefEnd="d2"> naked
        mole rat</pc> is <pc id="2" dataRefStart="d1"
        dataRefEnd="d2"> pretty ugly</pc>.</source>
    <target>Cet <pc id="1" dataRefStart="d1" dataRefEnd="d2">
        hétérocéphale</pc> est <pc id="2" dataRefStart="d1"
        dataRefEnd="d2"> plutôt laid</pc>.</target>
  </segment>
</unit>
```

Note

This mechanism allows to re-use identical original data by pointing to the same `<data>` element.

Adding Codes

When processing content, there are possible cases when new inline codes need to be added.

For example, in the following HTML help content, the text has the name of a button in bold:

```
Press the <b>Emergency Stop</b> button
to interrupt the count-down sequence.
```

In the translated version, the original label needs to remain in English because the user interface, unlike the help, is not translated. However, for convenience, a translation is also provided and emphasized using another style. That new formatting needs to be added:

```
Appuyez sur le bouton <b>Emergency Stop</b> (<i>Arrêt d'urgence</i>)
pour interrompre le compte à rebours.
```

Having to split a single formatted span of text into several separate parts during translation, can serve as another example. For instance, the following sentence in Swedish uses bold on the names of two animals:

```
Äter <b>katter möss</b>?
```

But the English translation separates the two names and therefore needs to duplicate the bold codes.

```
Do <b>cats</b> eat <b>mice</b>?
```

Processing Requirements

- *Modifiers may* add inline codes.
- The `id` value of the added code *must* be different from all `id` values in both source and target content of the unit where the new code is added.
- *Mergers may* ignore added inline codes when *Merging* the *Translated* content back into the original format.

There are several ways to add codes:

Duplicating an existing code

One way to create a new code is to duplicate an existing one (called the *base code*).

If the base code is associated with some original data: the new code simply uses the same data.

For example, the translation in the following unit, the second inline code is a duplicate of the first one:

```
<unit id="1">
  <originalData>
    <data id="d1">&lt;b></data>
    <data id="d2">&lt;/b></data>
  </originalData>
  <segment>
    <source>Äter <pc id="1" dataRefStart="d1" dataRefEnd="d2">katter
        möss</pc>?</source>
    <target>Do <pc id="1" dataRefStart="d1" dataRefEnd="d2">
        cats</pc> eat <pc id="2" dataRefStart="d1"
        dataRefEnd="d2">mice</pc>?</target>
  </segment>
</unit>
```

If the base code has no associated data, the new code *must* use the `copyOf` attribute to indicate the `id` of the base code. This allows the merging tool to know what original data to re-use.

For example, the translation in the following unit, the second inline code is a duplicate of the first one:

```
<unit id="1">
  <segment>
```

```
    <source>Esznek <pc id="1">a magyarok svéd húsgombócot
        </pc>?</source>
    <target>Do <pc id="1">Hungarians</pc> eat <pc id="2"
        copyOf="1">Swedish meatballs</pc>?</target>
  </segment>
</unit>
```

Processing Requirements

- *Modifiers must not* clone a code that has its `canCopy` attribute is set to `no`.
- The `copyOf` attribute *must* be used when, and only when, the base code has no associated original data.

Creating a brand-new code

Another way to add a code is to create it from scratch. For example, this can happen when the translated text requires additional formatting.

For example, in the following unit, the UI text needs to stay in English, and is also translated into French as a hint for the French user. The French translation for the UI text is formatted in italics:

```
<unit id="1">
  <originalData>
    <data id="d1">&lt;b></data>
    <data id="d2">&lt;/b></data>
    <data id="n1">&lt;i></data>
    <data id="n2">&lt;/i></data>
  </originalData>
  <segment>
    <source>Press the <pc id="1" dataRefStart="d1" dataRefEnd="d2">
        Emergency Stop</pc> button to interrupt the count-down
        sequence. </source>
    <target>Appuyez sur le bouton <pc id="1" dataRefStart="d1"
        dataRefEnd="d2">Emergency Stop</pc> (<pc id="2"
        dataRefStart="n1" dataRefEnd="n2">Arrêt d'urgence
        </pc>) pour interrompre le compte à rebours. </target>
  </segment>
</unit>
```

Converting text into a code

Another way to add a code is to convert part of the extracted text into code. In some cases the inline code can be created after extraction, using part of the text content. This can be done, for instance, to get better matches from an existing TM, or better candidates from an MT system.

For example, it can happen that a tool extracting a Java properties file to XLIFF is not sophisticated enough to treat HTML or XML snippets inside the extracted text as inline code:

```
# text property for the widget 'next'
nextText: Click <ui>Next</ui>
```

Resulting XLIFF content:

```
<unit id="1">
  <segment>
    <source>Click &lt;ui>Next&lt;/ui></source>
  </segment>
</unit>
```

But another tool, later in the process, can be used to process the initial XLIFF document and detect additional inline codes. For instance here the XML elements such as `<ui>`.

The original data of the new code is the part of the text content that is converted as inline code.

```
<unit id="1">
  <originalData>
    <data id="d1">&lt;ui></data>
    <data id="d2">&lt;/ui></data>
  </originalData>
  <segment>
    <source>Click <pc id="1" dataRefStart="d1" dataRefEnd="d2">
        Next</pc></source>
  </segment>
</unit>
```

Warning

Converting XLIFF text content into original data for inline code might need a tool-specific process as the tool which did the initial extraction could have applied some conversion to the original content to create the XLIFF content (e.g. un-escape special characters).

Removing Codes

When processing content, there are some possible cases when existing inline codes need to be removed.

For an example the translation of a sentence can result in grouping of several formatted parts into a single one. For instance, the following sentence in English uses bold on the names of two animals:

```
Do <b>cats</b> eat <b>mice</b>?
```

But the Swedish translation group the two names and therefore needs only a single bolded part.

```
Äter <b>katter möss</b>?
```

Processing Requirements

- User agents *may* remove a given inline code only if its `canDelete` attribute is set to `yes`.
- When removing a given inline code, the user agents *must* remove its associated original data, except if the original data is shared with another inline code that remains in the unit.

Note that having to delete the original data is unlikely because such original data is likely to be associated to an inline code in the source content.

There are several ways to remove codes:

Deleting a code

One way to remove a code is to delete it from the extracted content. For example, in the following unit, the translated text does not use the italics formatting. It is removed from the target content, but the original data are preserved because they are still used in the source content.

```
<unit id="1">
  <originalData>
    <data id="d1">&lt;i></data>
    <data id="d2">&lt;/i></data>
  </originalData>
  <segment>
    <source>I read <pc id="1" dataRefStart="d1" dataRefEnd="d2">Little
        House on the Prairie</pc> to my children.</source>
    <target>                    </target>
  </segment>
</unit>
```

Converting a code into text

Another way to remove an inline code is to convert it into text content. This is likely to be a rare use case. It is equivalent to deleting the code, with the addition to place the original data for the given code into the content, as text. This can be done, for example, to get better matches from an existing TM, or better candidates from an MT system.

For instance, the following unit has an inline code corresponding to a variable placeholder. A tool can temporarily treat this variable as text to get better matches from an existing TM.

```
<unit id="1">
  <originalData>
    <data id="d1">%s</data>
  </originalData>
  <segment>
    <source>Cannot find '<ph id="1" dataRef="d1"/>'.</source>
  </segment>
</unit>
```

The modified unit would end up like as shown below. Note that because the original data was not associated with other inline code it has been removed from the unit:

```
<unit id="1">
  <segment>
    <source>Cannot find '%s'.</source>
  </segment>
</unit>
```

Warning

Converting the original data of an inline code into text content might need a tool-specific process as the tool which did the initial extraction could have applied some conversion to the original content.

Editing Hints

XLIFF provides some information about what editing operations are applicable to inline codes:

- A code can be deleted: That is, the code element as well as its original data (if any are attached) are removed from the document. This hint is represented with the `canDelete` attribute. The default value is `yes`: deletion is allowed.

  For example, the following extracted C string has the code `<ph id='1'/>` set to be not deletable because removing the original data (the variable placeholder `%s`) from the string would result in an error when running the application:

- A code can be copied: That is, the code is used as a *base code* for adding another inline code. See the section called "Duplicating an existing code" for more details. This hint is represented with the `canCopy` attribute. The default value is `yes`: copy is allowed.

- A code can be re-ordered: That is, a given code can be moved before or after another inline code. This hint is represented with the `canReorder` attribute. The default value is `yes`: re-ordering is allowed.

Note

Please note that often those properties are related and appear together. For example, the code in the first unit shown below is a variable placeholder that has to be preserved and cannot be duplicated, and when several of such variables are present, as in the second unit, they cannot be re-ordered:

```
<unit id="1">
  <originalData>
    <data id="d1">%s</data>
  </originalData>
  <segment>
    <source>Can't open '<ph id="1" dataRef="d1" canCopy="no"
        canDelete="no"/>'.</source>
  </segment>
</unit>
<unit id="2">
  <originalData>
    <data id="d1">%s</data>
    <data id="d2">%d</data>
  </originalData>
  <segment>
    <source>Number of <ph id="1" dataRef="d1" canCopy="no"
        canDelete="no" canReorder="firstNo"/>: <ph id="2" dataRef="d2"
        canCopy="no" canDelete="no" canReorder="no"/>. </source>
  </segment>
</unit>
```

See the Target Content Modification section for additional details on editing.

Constraints

- When the attribute `canReorder` is set to `no` or `firstNo`, the attributes `canCopy` and `canDelete` *must* also be set to `no`.
- Inline codes re-ordering within a source or target content *may* be limited by defining non-reorderable sequences. Such sequence is made of a first inline code with the attribute `canReorder` set to `firstNo` and zero or more following codes with `canReorder` set to `no`.
- A non-reorderable sequence of codes *must not* start with a code with the attribute `canReorder` set to `No` and zero or more following codes with `canReorder` set to `no`

Note

A non-reorderable sequence made of a single code with `canReorder` set to `firstNo` are allowed just for *Extraction* convenience and are equivalent to a code with the attribute `canReorder` set to `yes`.

Processing Requirements

- *Extractors should* set the `canDelete`, `canCopy` and `canReorder` attributes for the codes that need to be treated differently than with the default settings.
- *Modifiers must not* change the number and order of the inline codes making up a non-reorderable sequence.
- *Modifiers may* move a whole non-reorderable sequence before or after another non-reorderable sequence.
- When a non-reorderable sequence is made of a single non-reorderable code, *Modifiers MAY* remove the `canReorder` attribute of that code or change its value to `yes`.
- *Modifiers must not* delete inline codes that have their attribute `canDelete` set to `no`.
- *Modifiers must not* replicate inline codes that have their attribute `canCopy` set to `no`.

Note

Conformance of codes to Editing Hints Processing Requirements within *Translations* can only be checked on existing `<target>` elements, i.e. non-conformance is not reported on `<segment>` or `<ignorable>` elements without `<target>` children.

The XLIFF Core Schematron Schema will throw *Warnings* for all existing `<target>` elements where codes don't conform to the Editing Hints Processing Requirements, except for `<target>` children of `<segment>` elements with the `state` attribute set to `final`, where it will throw *Errors*.

Annotations

An annotation is an element that associates a section of the content with some metadata information.

Annotations *may* be created by an *Extractor* that generated the initial *XLIFF Document*, or by any other *Modifier* or *Enricher* later in the process. For example, after an *Extractor*

creates the document, an *Enricher* can annotate the source content with terminological information.

Annotations are represented using either the `<mrk>` element, or the pair of `<sm>` and `<em>` elements.

Type of Annotations

There are several pre-defined types of annotation and definition of custom types is also allowed.

Translate Annotation

This annotation is used to indicate whether a span of content is translatable or not.

Usage:

- The `id` attribute is *required*
- The `translate` attribute is *required* and set to `yes` or `no`
- The `type` attribute is *optional* and set to `generic` (this is the default value)

For example:

```
He saw his <mrk id="m1" translate="no">doppelgänger</mrk>.
```

Note

This annotation overrides the `translate` attribute set or inherited at the `<unit>` level.

Note

The `translate` attribute can also be used at the same time as another type of annotation. For example:

```
He saw his <mrk id="m1" translate="no" type="term">doppelgänger
</mrk>.
```

Term Annotation

This annotation is used to mark up a term in the content, and possibly associate information to it.

Usage:

- The `id` attribute is *required*
- The `type` attribute is *required* and set to `term`
- The `value` attribute is *optional* and contains a short definition of the term
- The `ref` attribute is *optional* and contains a URI pointing to information on the term
- The `translate` attribute is *optional* and set to `yes` or `no`

For example:

```
<file id="f-t_a">
  <unit id="1">
```

```
    <segment>
      <source>He is my <mrk id="m1" type="term"
          ref="http://dbpedia.org/page/Doppelgänger">
          doppelgänger</mrk>. </source>
    </segment>
  </unit>
</file>
```

Comment Annotation

This annotation is used to associate a span of content with a comment.

Usage:

- The `id` attribute is *required*
- The `type` attribute is *required* and set to `comment`
- If the `value` attribute is present it contains the text of the comment. If and only if the `value` attribute is not present, the `ref` attribute *must* be present and contain the URI of a `<note>` element within the same enclosing `<unit>` element that holds the comment.
- The `translate` attribute is *optional* and set to `yes` or `no`

For example, here with the `value` attribute:

```
The <mrk id="m1" type="comment"
 value="Possible values: Printer or Stacker"><ph id="1" dataRef="d1"/>
</mrk>
has been enabled.
```

And here using the `ref` attribute:

```
<unit id="1">
  <notes>
    <note id="n1" appliesTo="target">Please check the translation for
        'namespace'. One also can use 'espace de nom', but I think most
        technical manuals use the English term.</note>
  </notes>
  <segment>
    <source>You use your own namespace.</source>
    <target>Vous pouvez utiliser votre propre <mrk id="m1"
        type="comment" ref="#n=n1">namespace</mrk>.</target>
  </segment>
</unit>
```

Custom Annotation

The `<mrk>` element can be used to implement custom annotations.

A custom annotation *must not* provide the same functionality as a pre-defined annotation.

Usage:

- The `id` attribute is *required*
- The `type` attribute is *required* and set to a unique user-defined value.

- The `translate` attribute is *optional* and set to `yes` or `no`
- The use and semantics of the `value` and `ref` attributes are user-defined.

For example:

```
One of the earliest surviving works of literature is
<mrk id="m1" type="myCorp:isbn" value="978-0-14-44919-8">The
Epic of Gilgamesh</mrk>.
```

Splitting Annotations

Annotations can overlap spanning inline codes or other annotations. They also can be split by segmentation. Because of this, a single annotation span can be represented using a pair of `<sm>` and `<em>` elements instead of a single `<mrk>` element.

For example, one can have the following content:

```
<unit id="1">
  <segment>
    <source>Sentence A. <mrk id="m1" type="comment" value="Comment for B
        and C">Sentence B. Sentence C.</mrk></source>
  </segment>
</unit>
```

After a user agent performs segmentation, the annotation element `<mrk>` is changed to a pair of `<sm>` and `<em>` elements:

```
<unit id="1">
  <segment>
    <source>Sentence A. </source>
  </segment>
  <segment>
    <source><sm id="m1" type="comment" value="Comment for B and C"/>
        Sentence B. </source>
  </segment>
  <segment>
    <source>Sentence C.<em startRef="m1"/></source>
  </segment>
</unit>
```

Sub-Flows

A sub-flow is a section of text embedded inside an inline code, or inside another section of text.

For example, the following HTML content includes two sub-flows: The first one is the value of the `title` attribute ("`Start button`"), and the second one is the value of the `alt` attribute ("`Click here to start!`"):

```
Click to start: <img title="Start button"
 src="btnStart.png" alt="Click here to start!"/>
```

Another example is the following DITA content where the footnote "`A Palouse horse is the same as an Appaloosa.`" is defined at the middle of a sentence:

```
Palouse horses<fn>A Palouse horse is the same as
 an Appaloosa.</fn> have spotted coats.
```

In XLIFF, each sub-flow is stored in its own `<unit>` element, and the `subFlows` attribute is used to indicate the location of the embedded content.

Therefore the HTML content of the example above can be represented like below:

```
<unit id="1">
  <segment>
    <source>Start button</source>
  </segment>
</unit>
<unit id="2">
  <segment>
    <source>Click here to start!</source>
  </segment>
</unit>
<unit id="3">
  <segment>
    <source>Click to start: <ph id="1" subFlows="1 2"/></source>
  </segment>
</unit>
```

Constraints

- An inline code containing or delimiting one or more sub-flows *must* have an attribute `subFlows` that holds a list of the identifiers of the `<unit>` elements where the sub-flows are stored.
- Sub-flows *must* be in the same `<file>` element as the `<unit>` element from which they are referenced.

Processing Requirements

- *Extractors should* store each sub-flow in its own `<unit>` element.
- *Extractors may* order the `<unit>` elements of the sub-flows and the `<unit>` element, from where the sub-flows are referenced, as they see fit.

Note

Please note that the static structure encoded by `<file>`, `<group>`, and `<unit>` elements is principally immutable in *XLIFF Documents* and hence the unit order initially set by the *Extractor* will be preserved throughout the roundtrip even in the special case of sub-flows.

White Spaces

While white spaces can be significant or insignificant in the original format, they are always treated as significant when stored as original data in XLIFF. See the definition of the `<data>` element.

Processing Requirements

- For the inline content and all non empty inline elements: The white spaces *must* be preserved if the value for `xml:space` set or inherited at the enclosing `<unit>` level is `preserve`, and they *may* be preserved if the value is `default`.

Bidirectional Text

Text directionality in XLIFF content is defined by inheritance. Source and target content can have different directionality.

The initial directionality for both the source and the target content is defined in the `<file>` element, using the *optional* attributes `srcDir` for the source and `trgDir` for the target. The default value for both attributes is `auto`.

The `<group>` and `<unit>` elements also have the two *optional* attributes `srcDir` and `trgDir`. The default value of the `srcDir` is inherited from the value of the `srcDir` attribute of the respective parent element. The default value of the `trgDir` attribute is inherited from the value of the `trgDir` attribute of the respective parent element.

The `<pc>`, `<sc>`, and isolated `<ec>` elements have an *optional* attribute `dir` with a value `ltr`, `rtl`, or `auto`. The default value is inherited from the parent `<pc>` element. In case the inline element is a child of a `<source>` element, the default value is inherited from the `srcDir` value of the enclosing `<unit>` element. In case the inline element is a child of a `<target>` element, the default value is inherited from the `trgDir` value of the enclosing `<unit>` element.

Warning

While processing isolated `<ec>` elements with explicitly set directionality, please beware that unlike directionality set on the `<pc>` and `<sc>` , this method decreases the stack level as per [UAX #9].

In addition, the `<data>` element has an *optional* attribute `dir` with a value `ltr`, `rtl`, or `auto` that is not inherited. The default value is `auto`.

Directionality of source and target text contained in the `<source>` and `<target>` elements is fully governed by [UAX #9], whereas explicit *XLIFF-defined* structural and directionality markup is a higher-level protocol in the sense of [UAX #9]. The *XLIFF-defined* value `auto` determines the directionality based on the first strong directional character in its scope and *XLIFF-defined* inline directionality markup behaves exactly as Explicit Directional Isolate Characters, see [UAX #9], http://www.unicode.org/reports/tr9/#Directional_Formatting_Characters.

Note

Please note that this specification does not define explicit markup for inline directional Overrides or Embeddings; in case those are needed. *Extractors* and *Modifiers* will need to use [UAX #9] defined Directional Formatting Characters.

For instance, HTML elements `<bdi>` and `<bdo>` need both *Extracted* as a `<pc>` or `<sc>` / `<ec/>` pair with the `dir` attribute set respectively.

All XLIFF defined inline directionality markup isolates and `<sc>` / `<ec/>` isolated spans can reach over segment (but not unit) boundaries. This needs to be taken into account when splitting or joining segments (see Segmentation Modification) that

contain inline directionality markup. Albeit It is not advisable to split segments, so that corresponding inline directionality markup start and end would fall into different segments, such a situation is not too confusing. If this happens, the "watertight" BiDi box will simply span two or more segments. This is not too confusing because no XLIFF defined directionality markup is allowed on `<source>`, <target>, or , so all higher level protocol inheritance of directionality in such cases is from <unit> or higher.

Target Content Modification

This section defines the rules *Writers* need to follow when working with the target content of a given segment in order to provide interoperability throughout the whole process.

The *Extractor may* create the initial target content as it sees fit.

The *Merger* is assumed to have the same level of processing and native format knowledge as the *Extractor*. Providing an interoperable way to convert native documents into XLIFF with one tool and back to the native format with another tool without the same level of knowledge is outside the scope of this specification.

The *Writers Modifying* the target content of an *XLIFF Document* between the *Extractor* and the *Merger* ensure interoperability by applying specific rules. These rules are separated into two cases: When there is an existing target and when there is no existing target.

Without an Existing Target

When there is no existing target, the processing requirements for a given segment are the following:

Processing Requirements

- *Writers may* leave the segment without a target.
- *Modifiers may* create a new target as follows:
  - *Modifiers may* add translation of the source text.
  - *Modifiers must* put all non-removable inline codes in the target.
  - *Modifiers must* preserve the order of all the non-reorderable inline codes.
  - *Modifiers may* put any removable inline code in the target.
  - *Modifiers may* add inline codes.
  - *Modifiers may* add or remove annotations.
  - *Modifiers may* convert any `<pc>` element into a pair of `<sc>` and `<ec>` elements.
  - *Modifiers may* convert, if it is possible, any pair of `<sc>` and `<ec>` elements into a `<pc>` element.

With an Existing Target

When working with a segment with content already in the target, *Writers must* choose one of the three behaviors described below:

Processing Requirements

- *Writers may* leave the existing target unchanged.
- *Modifiers may* modify the existing target as follow:
  - *Modifiers may* add or *Modify* translatable text.
  - *Writers must* preserve all non-removable inline codes, regardless whether or not they exist in the source.
  - *Writers must* preserve any non-reorderable inline codes in the existing target.
  - *Writers must not* add any non-reorderable inline codes to the target.
  - *Modifiers may* remove any removable inline codes in the target.
  - *Modifiers may* add inline codes (including copying any cloneable inline codes of the existing target).
  - *Modifiers may* add or remove annotations.
  - *Modifiers may* convert any `<pc>` element into a pair of `<sc>` and `<ec>` elements.
  - *Modifiers may* convert, if it is possible, any pair of `<sc>` and `<ec>` elements into a `<pc>` element.
- *Modifiers may* delete the existing target and start over as if working without an existing target.

Content Comparison

This specification defines two types of content equality:

- Equality type A: Two contents are equal if their normalized forms are equal.
- Equality type B: Two contents are equal if, in their normalized forms and with all inline code markers replaced by the value of their `equiv` attributes, the resulting strings are equal.

A content is normalized when:

- The text nodes are in Unicode Normalized Form C defined in the Unicode Annex #15: Unicode Normalization Forms [UAX #15].
- All annotation markers are removed.
- All pairs of `<sc>` and `<ec>` elements that can be converted into a `<pc>` element, are converted.
- All adjacent text nodes are merged into a single text node.
- For all the text nodes with the white space property set to `default`, all adjacent white spaces are collapsed into a single space.

Segmentation

In the context of XLIFF, a segment is content which is either a unit of extracted text, or has been created from a unit of extracted text by means of a segmentation mechanism

such as sentence boundary detection. For example, a segment can be a title, the text of a menu item, a paragraph or a sentence in a paragraph.

In the context of XLIFF, other types representations sometimes called "segmentation" can be represented using annotations. For example: the terms in a segment can be identified and marked up using the term annotation.

XLIFF does not specify how segmentation is carried out, only how to represent its result. Material provisions regarding segmentation can be found for instance in the Segmentation Rules eXchange standard [SRX] or [UAX #29].

Segments Representation

In XLIFF each segment of processed content is represented by a `<segment>` element.

A `<unit>` can comprise a single `<segment>`.

Each `<segment>` element has one `<source>` element that contains the source content and one *optional* `<target>` element that can be empty or contain the translation of the source content at a given state.

Content parts between segments are represented with the `<ignorable>` element, which has the same content model as `<segment>`.

For example:

```
<unit id="1">
  <segment>
    <source>First sentence.</source>
    <target>Première phrase.</target>
  </segment>
  <ignorable>
    <source> </source>
  </ignorable>
  <segment>
    <source>Second sentence.</source>
  </segment>
</unit>
```

Segments Order

Some *Agents* (e.g. aligner tools) can segment content, so that the target segments are not in the same order as the source segments.

To be able to map order differences, the `<target>` element has an *optional* `order` attribute that indicates its position in the sequence of segments (and inter-segments). Its value is an integer from 1 to N, where N is the sum of the numbers of the `<segment>` and `<ignorable>` elements within the given enclosing `<unit>` element.

Warning

When *Writers* set explicit `order` on `<target>` elements, they have to check for conflicts with implicit `order`, as `<target>` elements without explicit `order` correspond to their sibling `<source>` elements. Beware that moving one `<target>` element is likely to cause a renumbering domino effect throughout the enclosing `<unit>` element.

For example, the following HTML documents have the same paragraph with three sentences in different order:

```
<p lang='en'>Sentence A. Sentence B. Sentence C.</p>
```

```
<p lang='fr'>Phrase B. Phrase C. Phrase A.</p>
```

The XLIFF representation of the content, after segmentation and alignment, would be:

```
<unit id="1">
  <segment id="1">
    <source>Sentence A.</source>
    <target order="5">Phrase A.</target>
  </segment>
  <ignorable>
    <source> </source>
  </ignorable>
  <segment id="2">
    <source>Sentence B.</source>
    <target order="1">Phrase B.</target>
  </segment>
  <ignorable>
    <source> </source>
  </ignorable>
  <segment id="3">
    <source>Sentence C.</source>
    <target order="3">Phrase C.</target>
  </segment>
</unit>
```

Segmentation Modification

When *Modifying* segmentation of a `<unit>`, *Modifiers must* meet the Constraints and follow the Processing Requirements defined below:

Constraints

- Integrity of the inline codes *must* be preserved. See the section on Inline Codes and on Annotations for details.
- The entire source content of any one `<unit>` element *must* remain logically unchanged: `<segment>` elements or their data *must not* be moved or joined across units.

Warning

Note that when splitting or joining segments that have both source and target content it is advisable to keep the resulting segments linguistically aligned, which is likely to require human linguistic expertise and hence manual re-segmentation. If the linguistically correct alignment cannot be guaranteed, discarding the target content and retranslating the resulting source segments is worth considering.

Processing Requirements

- When the *Modifiers* perform a split operation:

- Only or <ignorable> elements that have their canResegment value resolved to yes *may* be split.
- All new or <ignorable> elements created and their <source> and <target> children *must* have the same attribute values as the original elements they were created from, as applicable, except for the id attributes and, possibly, for the order, state and subState attributes.
- Any new id attributes *must* follow the or <ignorable> id constraints.
- If there was a target content in the original segment and if the state attribute of the original segment was not initial, the state attributes of the segments resulting from the split (and possibly their corresponding subState attributes) *may* be changed to reflect the fact that the target content *may* need to be verified as the new segmentation *may* have desynchronized the alignment between the source and target contents.

- When the *Modifiers* perform a join operation:
  - Only or <ignorable> elements that have their canResegment value resolved to yes *may* be join with other elements.
- When the *Modifiers* or *Mergers* perform a join operation:
  - Two elements (or <ignorable>) *must not* be joined if their <target> have resolved order values that are not consecutive.
  - The attributes of the elements to be joined (or <ignorable>) and the attributes of their <source> and <target> *must* be carried over in the resulting joined elements.
  - If attributes of elements to be joined (or <ignorable>) differ, or if the attributes of their <source> or <target> differ, the resulting joined elements *must* comply with following rules:
    - If the state attributes of the elements differ: the state attribute of the joined *must* be set to the "earliest" of the values specified in the original elements. The sequence of state values are defined in the following order: 1: initial, 2: translated, 3: reviewed, and 4: final.
    - The subState attribute *must* be the one associated with the state attribute selected to be used in the joined . If no subState attribute is associated with that state, the joined *must not* have a subState.
    - If the xml:space attributes differ: The <source> and <target> of the joined element *must* be set to xml:space="preserve".
- When the *Modifiers* or *Mergers* perform a join or a split operation:
  - If any or <ignorable> element of the <unit> had a <target> child with an order attribute prior to the segmentation modification, the <target> child of all and <ignorable> elements in the <unit> *must* be examined and if necessary their order attributes updated to preserve the ordering of the target content prior the segmentation modification.

Best Practice for *Mergers*

Since a typical simple corporate implementation of XLIFF 2 is a localization tool that is at the same time an *Extractor* and a *Merger* with the full knowledge of the *Extraction* mechanism, the community requested a non-normative best practice for *Merging* after an XLIFF Round-trip.

First of all, it needs to be noted that *Mergers* are not advised to rely on their knowledge of the *Extraction* mechanism in terms of segmentation. *Modifiers* are free to change segmentation during the roundtrip and even to change order of target content held in different segments of the same unit. Therefore, it can be advised as a best practice before *Merging* to look for all segments within each unit, even and especially when the *Extractor* had created only one segment per unit.

When joining segments, *Mergers* need to observe all *Processing Requirements* for joining segments and joining or splitting segments

When joining segments it can happen that not all `<segment>` or `<ignorable>` elements actually have their `<target>` element children. This situation can be legal depending on a specific workflow set up. The `<target>` child within an `<ignorable>` element is always optional, but at the same can be created any time by simply copying the content of the sibling `<source>`, see Content Modification Without Target. The presence of `<target>` children can be better governed in `<segment>` elements that have the `state` attribute. The `state` attribute is strictly optional with the default `initial`, yet it is advisable for a corporate localization operation to request that their service providers progress that attribute through `translated` and `reviewed` to `final`. This attribute cannot be progressed from the `initial` state without a `<target>` child and all violations of Editing Hints will become validation errors only in the `final` state. Usage of `state` also allows for fine-tuning of a specific workflow *State Machine* with the dependent `subState` attribute. With the attribute `subState`, implementers can create an arbitrary number of private state machine under their prefix authorities. It is advisable to register such authority prefixes with the XLIFF TC and publish their documentation.

When *Mergers* need to perform the *Merge* in a non-final state, when the presence of targets cannot be guaranteed, they are free to create preliminary targets again following the Processing Requirements for Content Modification Without Target

Extension Mechanisms

XLIFF 2.0 offers two mechanisms for storing custom data in an XLIFF document:

1. Using the Metadata module for storing custom data in elements defined by the official XLIFF specification.

2. Using the standard XML namespace mechanism for storing data in elements or attributes defined in a custom XML Schema.

Both mechanisms can be used simultaneously.

Extension Points

The following *XLIFF Core* elements allow storing custom data in `<mda:metadata>` elements or in elements from a custom XML namespace:

- `<file>`
- `<group>`

- `<unit>`

The following *XLIFF Core* elements accept custom attributes:

- `<xliff>`
- `<file>`
- `<group>`
- `<unit>`
- `<note>`
- `<mrk>`
- `<sm>`

Extensibility of *XLIFF Modules*

For extensibility of *XLIFF Modules* please refer to the relevant Module Sections.

Constraints

- When using identifiers, an extension *must* use either an attribute named `id` or the attribute `xml:id` to specify them.
- Extensions identifiers *must* be unique within their immediate `<file>`, `<group>` or `<unit>` enclosing element.
- Identifier values used in extensions *must* be of type `xs:NMTOKEN` or compatible with `xs:NMTOKEN` (e.g. `xs:NAME` and `xs:ID` are compatible).

These constraints are needed for the fragment identification mechanism.

Processing Requirements

- A user extension, whether implemented using `<mda:metadata>` or using a custom namespace, *must not* provide the same functionality as an existing XLIFF core or module feature, however it *may* complement an extensible XLIFF core feature or module feature or provide a new functionality at the provided extension points.
- *Mergers must not* rely on custom namespace extensions, other than the ones possibly defined in `<skeleton>`, to create the *Translated* version of the original document.
- *Writers* that do not support a given custom namespace based user extension *should* preserve that extension without *Modification*.

The Modules Specifications

This section specifies the *optional Modules* that *may* be used along with *Core* for advanced functionality.

Translation Candidates Module

Introduction

The source text of a document can be pre-processed against various translation resources (TM, MT, etc.) to provide translation candidates. This module provides an XLIFF capability to store lists of possible translations along with information about the similarity of the match, the quality of the translation, its provenance, etc.

Module Namespace and Validation Artifacts

The namespace for the Translation Candidates module is: `urn:oasis:names:tc:xliff:matches:2.0`

Schema and Schematron for this module are available at http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/matches.xsd and http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/matches.sch.

Module Fragment Identification Prefix

The fragment identification prefix for the Translation Candidates module is: `mtc`

XLIFF Core Reuse

The Translation Candidates Module reuses several *XLIFF Core* elements, most of them have mandatory `xlf:id`. The uniqueness scopes for the reused `xlf:id` attributes are separate from the *XLIFF Core*. The following states the exact normative Constraints for the validation purposes:

Constraints

- When the `xlf:id` attribute is used on `<xlf:mrk>`, `<xlf:sm>`, `<xlf:pc>`, `<xlf:sc>`, `<xlf:ec>`, or `<xlf:ph>` elements reused within the Translation Candidates Module:
  - The inline elements enclosed by a `<xlf:target>` element *must* use the duplicate `xlf:id` values of their corresponding inline elements enclosed within the sibling `<xlf:source>` element if and only if those corresponding elements exist.
  - Except for the above exception, the value *must* be unique among all of the above within the enclosing `<match>` element.
- When used on `<xlf:data>` elements reused within the Translation Candidates Module:

  The value *must* be unique among all `<xlf:data>` `xlf:id` attribute values within the enclosing `<match>` element.
- When the `xlf:dataRef`, `xlf:datarefstart`, and `xlf:dataRefEnd` attributes are used on `<xlf:pc>`, `<xlf:sc>`, `<xlf:ec>`, or `<xlf:ph>` elements reused within the Translation Candidates Module, their `NMTOKEN` values *must* identify `<data>` elements within the enclosing `<match>` element. Those attributes *must not* be used without corresponding `<data>` elements within the enclosing `<match>` element.

Translation Candidate Annotation

This annotation can be used to mark up the scope of a translation candidate within the content of a unit. This module can reference any source or even target spans of content that are referencable via the XLIFF Fragment Identification mechanism, however in case the corresponding fragment is not suitably delimited, the best way how to mark the relevant span is to use the following annotation.

Usage:

- The `id` attribute is *required*
- The `type` attribute is *required* and set to `mtc:match`

- The `ref` attribute is not used.
- The `translate` attribute is *optional*

For example:

```
<unit id="1">
  <mtc:matches>
    <mtc:match ref="#m1">
      <source>He is my friend.</source>
      <target>Il est mon ami.</target>
    </mtc:match>
    <mtc:match ref="#m1">
      <source>He is my best friend.</source>
      <target>Il est mon meilleur ami.</target>
    </mtc:match>
  </mtc:matches>
  <segment>
    <source><mrk id="m1" type="mtc:match">He is my friend.</mrk></source>
  </segment>
  <segment>
    <source>Yet, I barely see him.</source>
  </segment>
</unit>
```

Module Elements

The elements defined in the Translation Candidates module are: `<matches>` and `<match>`.

Tree Structure

Legend:

1 = one
+ = one or more
? = zero or one
* = zero or more

```
<matches>
|
+---<match> +
  |
  +---<mda:metadata> ?
  |
  +---<xlf:originalData> ?
  |
  +---<xlf:source> 1
  |
  +---<xlf:target> 1
  |
  +---<other> *
```

matches

Collection of matches retrieved from any leveraging system (MT, TM, etc.)

Contains:

- One or more `<match>` elements

match

A potential translation suggested for a part of the source content of the enclosing `<unit>` element.

Contains:

- Zero or one `<mda:metadata>` element followed by.
- Zero or one `<originalData>` element followed by
- One `<source>` element followed by
- One `<target>` element followed by
- elements from other namespaces, *optional*

Attributes:

- `id`, *optional*
- `matchQuality`, *optional*
- `matchSuitability`, *optional*
- `origin`, *optional*
- `ref`, *required*
- `reference`, *optional*
- `similarity`, *optional*
- `subType`, *optional*
- `type`, *optional*
- attributes from other namespaces, *optional*

Constraints

- When a `<target>` element is a child of `<match>` and the `reference` attribute is set to `yes`, the *optional* `xml:lang` attribute's value is not *required* to be equal to the value of the `trgLang` attribute of the enclosing `<xliff>` element.
- The following *XLIFF Module* attributes are explicitly allowed by the wildcard `other`:

  - attributes from the namespace `http://www.w3.org/2005/11/its`, *optional*, provided that the Constraints specified in the ITS Module are met.
  - attributes from the namespace `urn:oasis:names:tc:xliff:itsm:2.1`, *optional*, provided that the Constraints specified in the ITS Module are met.

Module Attributes

The attributes defined in the Translation Candidates module are: `id`, `matchQuality`, `matchSuitability`, `origin`, `ref`, `reference`, `similarity`, `subType`, and `type`.

id

Identifier - a character string used to identify a `<match>` element.

Value description: NMTOKEN.

Default value: undefined

Used in: `<match>`.

Constraints

- The `id` value *must* be unique within the enclosing `<matches>` element.

matchQuality

Match quality - indicates the quality of the `<target>` child of a `<match>` element based on an external benchmark or metric.

Value description: a decimal number between 0.0 and 100.0.

Default value: undefined

Used in: `<match>`.

Note

This attribute can carry a human review based metrics score, a Machine Translation self-reported confidence score etc.

matchSuitability

Match suitability - indicates the general suitability and relevance of its `<match>` element based on various external benchmarks or metrics pertaining to both the `<source>` and the `<target>` children of the `<match>`.

This attribute is intended to carry a value that can be combined from values provided in `similarity` and `matchQuality` attributes based on an externally provided algorithm.

Value description: a decimal number between 0.0 and 100.0.

Default value: undefined

Used in: `<match>`.

Note

This attribute is also useful for mapping match-quality as specified in XLIFF 1.2 because 1.2 is not capable of discerning between the source similarity and the target quality.

Processing Requirements

- *Agents* processing this module *must* make use of `matchSuitability` for match ordering purposes if the attribute is specified.

origin

Match origin - indicates the tool, system or repository that generated a `<match>` element. This is a free text short informative description. For example, 'Microsoft Translator Hub' or 'tm-client123-v456', or 'MSTH (52217d25-d9e7-54a2-af44-3d4e4341d112_healthc).'

Value description: Text.

Default value: undefined

Used in: `<match>`.

ref

Reference - points to a span of text within the same unit, to which the translation candidate is relevant.

Value description: IRI

Default value: undefined

Used in: `<match>`.

Constraints

- The value of the `ref` attribute *must* point to a span of text within the same `<unit>` element where the `<match>` is located.

reference

Reference - indicates that the `<target>` child of the `<match>` element contains a *Translation* into a reference language rather than into the target language. For example, a German translation can be used as reference by a Luxembourgish translator.

Value description: `yes` or `no`.

Default value: `no`.

Used in: `<match>`

similarity

Similarity - indicates the similarity level between the content of the `<source>` child of a `<match>` element and the translatable text being matched.

Value description: a decimal number between 0.0 and 100.0.

Default value: undefined

Used in: `<match>`.

subType

Sub-type - indicates the sub-type, i.e. a secondary level type, of a `<match>` element.

Value description:

The value is composed of a prefix and a sub-value separated by a character : (U+003A). The prefix is a string uniquely identifying a collection of values for a specific authority. The sub-value is any string value defined by an authority.

The prefix `xlf` is reserved for this specification, but no sub-values are defined for it at this time. Other prefixes and sub-values *may* be defined by the users.

- *Default value:* undefined

Used in: `<match>`

Constraints

- If the attribute `subType` is used, the attribute `type` *must* be explicitly set.

Processing Requirements

- *Writers* updating the attribute `type` *must* also update or delete `subType` .

type

Type - indicates the type of a `<match>` element, it gives the value providing additional information on how the match was generated or qualifying further the relevance of the match. The list of pre-defined values is general and user-specific information can be added using the `subType` attribute.

Value description:

Table 3. Values

Value	Description
am	Assembled Match: candidate generated by assembling parts of different translations. For example: constructing a candidate by using the known translations of various spans of content of the source.
mt	Machine Translation: candidate generated by a machine translation system.
icm	In Context Match: candidate for which the content context of the translation was the same as the one of the current source. For example: the source text for both contents is also preceded and/or followed by an identical source segment, or both appear as e.g. level 2 headings.
idm	Identifier-based Match: candidate that has an identifier identical to the one of the source content. For example: the previous translation of a given UI component with the same ID. match that has an identifier identical to the source content.
tb	Term Base: candidate obtained from a terminological database, i.e. the whole source segment matches with a source term base entry.
tm	Translation Memory: candidate based on a simple match of the source content.
other	Candidate of a top level type not covered by any of the above definitions.

- *Default value:* tm

Used in: `<match>`

Processing Requirements

- *Writers* updating the attribute `type` *must* also update or delete `subType` .

Example

```
<mtc:matches>
  <mtc:match id="[NMTOKEN]">
    <xlf:source><!-- text data --></xlf:source>
    <xlf:target><!-- text data --></xlf:target>
    <xlf:originalData>
      <xlf:data id="[NMTOKEN]">
        <xlf:cp hex="[required]"><!-- text data --></xlf:cp>
      </xlf:data>
    </xlf:originalData>
    <mda:metadata>
      <mda:metagroup><!-- One or more of mda:metagroup or mda:meta -->
          </mda:metagroup>
    </mda:metadata>
    <!-- Zero, one or more elements from any namespace -->
  </mtc:match>
</mtc:matches>
```

Glossary Module

Introduction

Simple glossaries, consisting of a list of terms with a definition or translation, can be optionally embedded in an XLIFF document using the namespace mechanism to include elements from the Glossary module.

Module Namespace and Validation Artifacts

The namespace for the Glossary module is: `urn:oasis:names:tc:xliff:glossary:2.0`

Schema and Schematron for this module are available at http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/glossary.xsd and http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/glossary.sch

Module Fragment Identification Prefix

The fragment identification prefix for the Glossary module is: `gls`

Module Elements

The elements defined in the Glossary module are: `<glossary>`, `<glossEntry>`, `<term>`, `<translation>` and `<definition>`.

Tree Structure

Legend:

1 = one

+ = one or more
? = zero or one
* = zero, one or more

```
<glossary>
|
+---<glossEntry> +
  |
  +---<term> 1
  |
  +---<translation> *
  |
  +---<definition> ?
  |
  +---<other> *
```

glossary

Container for a list of glossary terms.

Contains:

- One or more `<glossEntry>` elements.

glossEntry

Glossary entry.

Contains:

- One `<term>` element followed by
- Zero, one or more `<translation>` elements followed by
- Zero or one `<definition>` element followed by
- elements from other namespaces, *optional*

Attributes:

- `id` , *optional*
- `ref` , *optional*
- attributes from other namespaces, *optional*

Constraints

- A `<glossEntry>` element *must* contain a `<translation>` or a `<definition>` element to be valid.
- The following *XLIFF Module* elements are explicitly allowed by the wildcard `other`:

  - Zero or one `<mda:metadata>` elements

term

A term in the glossary, expressed in the source language of the enclosing `<xliff>` element.

Contains:

- Text

Attributes:

- `source` , *optional*
- attributes from other namespaces, *optional*

translation

A translation of the sibling `<term>` element expressed in the target language of the enclosing `<xliff>` element. Multiple translations can be specified as synonyms.

Contains:

- Text

Attributes:

- `id`, *optional*
- `ref` , *optional*
- `source`, *optional*
- attributes from other namespaces, *optional*

definition

Optional definition in plain text for the term stored in the sibling `<term>` element.

Contains:

- Text

Attributes:

- `source` , *optional*
- attributes from other namespaces, *optional*

Module Attributes

The attributes defined in the Glossary module are: `id`, `ref`, and `source`

id

Identifier - a character string used to identify a `<glossEntry>` or `<translation>` element.

Value description: NMTOKEN

Default value: undefined

Used in:`<glossEntry>` and `<translation>`

Constraints

- The values of `id` attributes *must* be unique among all `<glossEntry>` and `<translation>` elements within the given enclosing `<glossary>` element.

ref

Reference - points to a span of source or target text within the same unit, to which the glossary entry is relevant.

Value description: IRI

Default value: undefined

Used in: `<glossEntry>` and `<translation>`.

Constraints

- The value of the `ref` attribute *MUST* point to a span of text within the same `<unit>` element, where the enclosing `<glossary>` element is located.

source

Source - indicates the origin of the content of the element where the attribute is defined.

Value description: Text.

Default value: undefined

Used in:`<term>`, `<translation>`, and `<definition>`.

Example

```
<unit id="1">
  <gls:glossary>
    <gls:glossEntry ref="#m1">
      <gls:term source="publicTermbase">TAB key</gls:term>
      <gls:translation id="1" source="myTermbase">Tabstopptaste
          </gls:translation>
      <gls:translation ref="#t=m1" source="myTermbase">TAB-TASTE
          </gls:translation>
      <gls:definition source="publicTermbase">A keyboard key that is
          traditionally used to insert tab characters into a document.
          </gls:definition>
    </gls:glossEntry>
  </gls:glossary>
  <segment>
    <source>Press the <mrk id="m1" type="term">TAB key</mrk>.</source>
    <target>Drücken Sie die <mrk id="m1" type="term">TAB-TASTE</mrk>. </target>
  </segment>
</unit>
```

Format Style Module

Introduction

This is intended as a namespace mechanism to carry inside an XLIFF document information needed for generating a quick at a glance HTML preview of XLIFF content using a predefined set of simple HTML formatting elements.

Module Namespace and Validation Artifacts

The namespace for the Format style module is: `urn:oasis:names:tc:xliff:fs:2.0`

Schema and Schematron for this module are available at http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/fs.xsd and http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/fs.sch.

Module Fragment Identification Prefix

Format Style module does not have a fragment identification prefix. Prefix `fs` is reserved in case it became needed in the future developments of this module.

Module Specification

Format Style module consists of just two attributes: `fs` and `subFs`. It does not specify any elements.

Format Style allows most structural and inline XLIFF core elements to convey basic formatting information using a predefined subset of HTML formatting elements. It primarily enables the generation of HTML pages or snippets for preview and review purposes. It *must not* be used to prescribe a roundtrip to a source document format.

The `fs` attribute holds the name of an HTML formatting element. If additional style information is needed, the *optional* `subFs` attribute is provided.

Constraints

- The Format Style attributes *must* be configured in such a way that the HTML [HTML5] snippet resulting at the `<file>` level is valid.

Processing Requirements

- *Extractors* and *Enrichers should* use the following method to validate their HTML snippets:

  1. Parse the snippet with the [HTML5] fragment parsing algorithm, see http://www.w3.org/TR/html5/syntax.html#parsing-html-fragments.
  2. the result *must* be a valid DOM tree as per [HTML5], see http://www.w3.org/TR/html5/infrastructure.html#tree-order.

> **Note**
>
> The above constraint and validation method will make sure that the snippets are renderable by standard HTML browsers.

Module Attributes

The attributes defined in the Format Style module are: `fs`, `subFs`.

fs

Format style attribute, fs - allows most structural and inline XLIFF core elements to convey basic formatting information using a predefined subset of HTML formatting elements (for example, HTML elements names like <script> are not included). It enables

the generation of HTML pages or snippets for preview and review purposes. If additional style information is needed, the *optional* `subFs` attribute is provided.

Value description:

Table 4. Values

a	anchor
b	bold text style
bdo	I18N BiDi over-ride
big	large text style
blockquote	long quotation
body	document body
br	forced line break
button	push button
caption	table caption
center	shorthand for DIV align=center
cite	citation
code	computer code fragment
col	table column
colgroup	table column group
dd	definition description
del	deleted text
div	generic language/style container
dl	definition list
dt	definition term
em	emphasis
h1	heading
h2	heading
h3	heading
h4	heading
h5	heading
h6	heading
head	document head
hr	horizontal rule
html	document root element
i	italic text style
img	image
label	form field label text
legend	fieldset legend
li	list item
ol	ordered list
p	paragraph
pre	preformatted text

q	short inline quotation
s	strike-through text style
samp	sample program output, scripts, etc.
select	option selector
small	small text style
span	generic language/style container
strike	strike-through text
strong	strong emphasis
sub	subscript
sup	superscript
table	
tbody	table body
td	table data cell
tfoot	table footer
th	table header cell
thead	table header
title	document title
tr	table row
tt	teletype or monospaced text style
u	underlined text style
ul	unordered list

Default value: undefined.

Used in: `<file>`, `<unit>`, `<note>`, `<sc>`, `<ec>`, `<ph>`, `<pc>`, `<mrk>`, and `<sm>`.

Warning

The `fs` attribute is not intended to facilitate *Merging* back into the original format.

Constraints

- The `fs` *must* only be used with `<ec>` in cases where the `isolated` attribute is set to 'yes'.

Processing Requirements

- *Writers* updating the attribute `fs` *must* also update or delete `subFs`.

Example: To facilitate HTML preview, fs can be applied to XLIFF like this like:

```
<xliff xmlns:fs="urn:oasis:names:tc:xliff:fs:2.0">
  <file fs:fs="html">
    <unit id="1" fs:fs="p">
      <segment>
        <source>Mick Jones renewed his interest in the Vintage <pc id="1"
            fs:fs="strong">'72 Telecaster Thinline </pc> guitar.
            <ph id="ph2" fs:fs="br" />He says <pc fs:fs="q">I love 'em
            </pc><ph id="ph1" fs:fs="img"
```

```
            fs:subFs="src,smileface.png" /></source>
      </segment>
    </unit>
  </file>
</xliff>
```

With an XSL stylesheet like this:

```
<xsl:template match="*" priority="2"
    xmlns:fs="urn:oasis:names:tc:xliff:fs:2.0">
  <xsl:choose>
    <xsl:when test="@fs:fs">
      <xsl:element name="{@fs:fs}">
        <xsl:if test="@fs:subFs">
          <xsl:variable name="att_name"
              select="substring-before(@fs:subFs,',')" />
          <xsl:variable name="att_val"
              select="substring-after(@fs:subFs,',')" />
          <xsl:attribute name="{$att_name}">
            <xsl:value-of select="$att_val" />
          </xsl:attribute>
        </xsl:if>
        <xsl:apply-templates />
      </xsl:element>
    </xsl:when>
    <xsl:otherwise>
      <xsl:apply-templates />
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

You can generate a an HTML page like this:

```
<html>
  <p>Mick Jones renewed his interest in the Vintage <strong>'72
      Telecaster Thinline </strong> guitar. <br/>He says <q>I love 'em
      </q><img src="smileface.png"/></p>
</html>
```

subFs

Sub-format style, subFs - allows extra metadata, like URL for example, to be added in concert with the `fs` attribute.

Value description: The subFs attribute is used to specify the HTML attributes to use along with the HTML element declared in the `fs` attribute. It is a list of name/value pairs. Each pair is separated from the next with a backslash (\). The name and the value of a pair are separated with a comma (,). Both literal backslash and comma characters are escaped with a backslash prefix.

Default value: undefined.

Used in: `<file>`, `<unit>`, `<note>`, `<source>`, `<target>`, `<sc>`, `<ec>`, `<ph>`, `<pc>`, `<mrk>`, and `<sm>`.

Warning

The `subFs` attribute is not intended to facilitate *Merging* back into the original format.

Constraints

- Commas (`,`) and backslashes (\) in the value parts of the `subFs` *must* be escaped with a backslash (\).
- If the attribute `subFs` is used, the attribute `fs` *must* be specified as well.
- The `subFs` *must* only be used with `<ec>` in cases where the `isolated` attribute is set to 'yes'.

Processing Requirements

- *Writers* updating the attribute `fs` *must* also update or delete `subFs`.

Example: For complex HTML previews that require more than one attribute on an HTML preview element, attribute pairs are separated by backslashes (). Any literal comma or backslash in an attribute value *must* be escaped with a backslash.

For example, we would use this convention:

```
<ph id="p1" fs="img" subFs="src,c:\\docs\\images\\smile.png\alt,
    My Happy Smile\title,Smiling faces\, are nice" />
```

To produce this HTML preview:

```
<img src="c:\docs\images\smile.png" alt="My Happy Smile" title="Smiling
    faces, are nice" />
```

Metadata Module

Introduction

The Metadata module provides a mechanism for storing custom metadata using elements that are part of the official XLIFF specification.

Module Namespace and Validation Artifacts

The namespace for the Metadata module is: `urn:oasis:names:tc:xliff:metadata:2.0`

Schema and Schematron for this module are available at http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/metadata.xsd and http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/metadata.sch.

Module Fragment Identification Prefix

The fragment identification prefix for the Metadata module is: `mda`

Module Elements

The elements defined in the Metadata module are: `<metadata>`, `<metaGroup>`, and `<meta>`.

Tree Structure

Legend:

+ = one or more

```
<metadata>
|
+---<metaGroup> +
  |
  +---At least one of (<metaGroup> OR <meta>)
      |
      +---<meta>
```

metadata

Container for metadata associated with the enclosing element.

Contains:

- One or more `<metaGroup>` elements

Attributes:

- `id`, *optional*

Example: Metadata can be used to store XML attribute names and values for *XLIFF Documents* that do not use a skeleton. The following XML sample contains attributes on the `<document>` and `<row>` elements.

```
<document version="3" phase="draft">
  <table>
    <row style="head">
      <cell>Name</cell>
      <cell>Position</cell>
    </row>
    <row>
      <cell>Patrick K.</cell>
      <cell>Right Wing</cell>
    </row>
    <row>
      <cell>Bryan B.</cell>
      <cell>Left Wing</cell>
    </row>
  </table>
</document>
```

The Metadata module can be used to preserve these attributes for a round trip without using a skeleton:

```
<?xml version="1.0" encoding="utf-8"?>
<xliff xmlns="urn:oasis:names:tc:xliff:document:2.0"
    xmlns:fs="urn:oasis:names:tc:xliff:fs:2.0"
    xmlns:mda="urn:oasis:names:tc:xliff:metadata:2.0" version="2.0"
    srcLang="en">
  <file id="f1">
    <group id="g1" name="document">
      <mda:metadata>
        <mda:metaGroup category="document_xml_attribute">
          <mda:meta type="version">3</mda:meta>
          <mda:meta type="phase">draft</mda:meta>
        </mda:metaGroup>
      </mda:metadata>
      <group id="g2" name="table">
        <group id="g3" name="row">
          <mda:metadata>
            <mda:metaGroup category="row_xml_attribute">
              <mda:meta type="style">head</mda:meta>
            </mda:metaGroup>
          </mda:metadata>
          <unit id="u1" name="cell">
            <segment>
              <source>Name</source>
            </segment>
          </unit>
          <unit id="u2" name="cell">
            <segment>
              <source>Position</source>
            </segment>
          </unit>
        </group>
        <group id="g4" name="row">
          <unit id="u3" name="cell">
            <segment>
              <source>Patrick K.</source>
            </segment>
          </unit>
          <unit id="u4" name="cell">
            <segment>
              <source>Right Wing</source>
            </segment>
          </unit>
        </group>
        <group id="g5" name="row">
          <unit id="u5" name="cell">
            <segment>
              <source>Bryan B.</source>
            </segment>
          </unit>
          <unit id="u6" name="cell">
            <segment>
              <source>Left Wing</source>
            </segment>
          </unit>
        </group>
      </group>
    </group>
```

```
  </file>
</xliff>
```

metaGroup

Provides a way to organize metadata into a structured hierarchy.

Contains:

- One or more `<metaGroup>` or `<meta>` elements in any order.

Attributes:

- `id`, *optional*
- `category`, *optional*
- `appliesTo`, *optional*

meta

Container for a single metadata component.

Contains:

- Non-translatable text

Attributes:

- `type`, *required*

Module Attributes

The attributes defined in the Metadata module are: `appliesTo`, `category`, `id`, and `type`.

appliesTo

Indicates the element to which the content of the metagroup applies.

Value description: `source`, `target`, or `ignorable`.

Default value: undefined.

Used in: `<metaGroup>`.

category

category - indicates a category for metadata contained in the enclosing `<metaGroup>` element.

Value description: Text.

Default value: undefined.

Used in: `<metaGroup>`.

id

Identifier - a character string used to identify a `<metadata>` or `<metaGroup>` element.

Value description: NMTOKEN

Default value: undefined

Used in:<metadata> and <metaGroup>

Constraints

- The values of id attributes *must* be unique among all <metaGroup> and <metadata> elements within the given enclosing <metadata> element.

type

type - indicates the type of metadata contained by the enclosing element.

Value description: Text.

Default value: undefined.

Used in: <meta>.

Resource Data Module

Introduction

The Resource Data module provides a mechanism for referencing external resource data that *may* need to be modified or used as contextual reference during translation.

Module Namespace and Validation Artifacts

The namespace for the Resource Data module is: urn:oasis:names:tc:xliff:resourcedata:2.0

Schema and Schematron for this module are available at http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/resource_data.xsd and http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/resource_data.sch.

Module Fragment Identification Prefix

The fragment identification prefix for the Resource Data module is: res

Module Elements

The elements defined in the Resource Data module are: <resourceData>, <resourceItemRef>, <resourceItem>, <source>, <target>, and <reference>.

Tree Structure

Legend:

? = zero or one
* = zero, one or more

```
<resourceData>
|
+---<resourceItemRef> *
```

```
|
+---<resourceItem> *
  |
  +---<source> ?
  | |
  | +---<other> *
  |
  +---<target> ?
  | |
  | +---<other> *
  |
  +---<reference> *
```

resourceData

Parent container for resource data associated with the enclosing element.

Contains:

At least one of the following

- Zero, one or more `<resourceItemRef>` elements.
- Zero, one or more `<resourceItem>` elements.

resourceItemRef

Specifies a reference to an associated `<resourceItem>` element located at the `<file>` level.

Contains:

This element is always empty.

Attributes:

- `id`, *optional*
- `ref`, *required*
- attributes from other namespaces, *optional*

Constraints

- The value of the *optional* id attribute *must* be unique among all `<resourceItem>` and `<resourceItemRef>` elements of the enclosing `<resourceData>` element.

Processing Requirements

- *Modifiers must* remove `<resourceItemRef>` when removing the referenced `<resourceItem>`.

resourceItem

Container for specific resource data that is either intended for *Modification*, or to be used as contextual reference during *Translation*.

Contains:

At least one of the following

- Zero or one `<source>` element followed by
- Zero or one `<target>` element followed by
- Zero, one or more `<reference>` elements

Attributes:

- `mimeType`, *optional*
- `id`, *optional*
- `context`, *optional*
- attributes from other namespaces, *optional*

Constraints

- The `mimeType` attribute is *required* if `<target>` and `<source>` child elements are empty, otherwise it is *optional*.
- The value of the *optional* id attribute *must* be unique among all `<resourceItem>` and `<resourceItemRef>` elements of the enclosing `<resourceData>` element.

Processing Requirements

- If a *Modifier* does not understand how to process the `mimeType` attribute, or the file it references, the `<resourceItem>` element *may* be ignored, but still *must* be preserved.
- The `mimeType` attribute *should* only be modified or removed if the referenced files are modified or removed.
- For each instance of `<resourceItem>` containing only `<source>`:
  - *Modifiers may* leave `<resourceItem>` unchanged, i.e. they are not *required* to create `<target>` or `<reference>`.
  - *Modifiers may* create `<target>` or `<reference>` as a siblings of `<source>`.

source

References the actual resource data that is either intended for *Modification*, or to be used as contextual reference during *Translation*.

Contains:

Either

- elements from other namespaces

or

- is empty.

Attributes:

- `href`, *optional*
- `xml:lang`, *optional*
- attributes from other namespaces, *optional*

Constraints

- The attribute `href` is *required* if and only if `<source>` is empty.
- When the *optional* `xml:lang` attribute is present, its value *must* be equal to the value of the `srcLang` attribute of the enclosing `<xliff>` element.

Processing Requirements

- When the `context` attribute of `<resourceItem>` is set to `yes`:
  - *Modifiers may* create `<source>` if not already present.
  - *Modifiers should not* change `<source>`.
  - *Modifiers may* remove `<source>`.
- When the `context` attribute of `<resourceItem>` is set to `no`:
  - `<source>` *must* be present.
  - *Modifiers must not* change `<source>`.
  - *Modifiers must not* remove `<source>`.

target

References the localized counterpart of the sibling `<source>` element.

Contains:

Either

- elements from other namespaces

or

- is empty.

Attributes:

- `href`, *optional*
- `xml:lang`, *optional*
- attributes from other namespaces, *optional*

Constraints

- The attribute `href` is *required* if and only if `<target>` is empty.
- When the *optional* `xml:lang` attribute is present, its value *must* be equal to the value of the `trgLang` attribute of the enclosing `<xliff>` element.

Processing Requirements

- When the `context` attribute of `<resourceItem>` is set to `yes`:
  - *Modifiers may* create `<target>` if not already present.

- *Modifiers should not* change `<target>`.
- *Modifiers may* remove `<target>`.

- When the `context` attribute of `<resourceItem>` is set to `no`:
  - *Modifiers may* create `<target>` if not already present.
  - *Modifiers may* leave `<target>` unchanged.
  - *Modifiers may* change `<target>`.
  - *Modifiers may* replace an existing `<target>`, i.e. the previously populated `<target>` *must not* be left blank.

reference

References contextual data relating to the sibling `<source>` and `<target>` elements, such as a German screenshot for a Luxembourgish translator.

Contains:

- This element is always empty.

Attributes:

- `href`, *required*
- `xml:lang`, *optional*
- attributes from other namespaces, *optional*

Constraints

- When the *optional* `xml:lang` attribute is present, its value does not need to be equal to the value of the `srcLang` or `trgLang` attribute of the enclosing `<xliff>` element.

Processing Requirements

- *Writers may* create `<reference>` if not already present.
- *Modifiers should not* change `<reference>`.
- *Modifiers may* remove `<reference>`.

Module Attributes

The attributes defined in the Resource Data module are: `id`, `xml:lang`, `mimeType`, `context`, `href`, and `ref`.

id

Identifier - A character string used to identify a `<resourceData>` element.

Value description: NMTOKEN

Default value: undefined

Used in:`<resourceItem>` and `<resourceItemRef>`

xml:lang

Language - The xml:lang attribute specifies the language variant of the text of a given element. For example: `xml:lang="fr-FR"` indicates the French language as spoken in France.

Value description: A language code as described in [BCP 47].

Default value: undefined

Used in: `<source>`, `<target>`, and `<reference>`.

mimeType

MIME type, mimeType - indicates the type of a resource object. This generally corresponds to the content type of [RFC 2045] [http://tools.ietf.org/rfc/rfc2045.txt], the MIME specification; e.g. mimeType="text/xml" indicates the resource data is a text file of XML format.

Value description: A MIME type. An existing MIME type *must* be used from a list of standard values [http://www.iana.org/assignments/media-types].

Default value: undefined

Used in:`<resourceItem>`

Note

If you cannot use any of the standard MIME type values as specified above, a new MIME type can be registered according to [RFC 2048] [http://tools.ietf.org/rfc/rfc2048.txt].

context

Contextual Information - Indicates whether an external resource is to be used for context only and not modified.

Value description: `yes` or `no`

Default value: `yes`

Used in:`<resourceItem>`

href

Hypertext Reference, href - IRI referencing an external resource.

Value description: IRI.

Default value: undefined

Used in:`<source>`, `<target>`, and `<reference>`

ref

Resource Item Reference - holds a reference to an associated `<resourceItem>` element located at the `<file>` level.

Value description: An [XML Schema Datatypes] NMTOKEN

Default value: undefined

Used in:<resourceItemRef>

Constraints

- The ref attribute value *must* be the value of the id attribute of the <resourceItem> element being referenced.

Examples

In this example, the <resourceData> module at <file> level references external XML that contains resource data for a user interface control. The control is the container for the text "Load Registry Config" and needs to be resized to accommodate the increased length of the string due to translation. The <resourceItemRef> element contained in the <resourceData> module at <unit> level provides the reference between them. The name attribute of the <unit> element could serve as the key for an editor to associate <source> and <target> text with the resource data contained in the referenced XML and display it for modification.

```
<file id="f1">
  <res:resourceData>
    <res:resourceItem id="r1" mimeType="text/xml" context="no">
      <res:source href="resources\en\registryconfig.resources.xml" />
      <res:target href="resources\de\registryconfig.resources.xml" />
    </res:resourceItem>
  </res:resourceData>
  <unit id="1" name="130;WIN_DLG_CTRL_">
    <res:resourceData>
      <res:resourceItemRef ref="r1" />
    </res:resourceData>
    <segment id="1" state="translated">
      <source>Load Registry Config</source>
      <target>Registrierungskonfiguration laden</target>
    </segment>
  </unit>
</file>
```

In this example, the <resourceData> module at the <unit> level contains elements from another namespace (abc), which could be displayed for modification in an editor that understands how to process the namespace.

```
<file id="f2" xmlns:abc="urn:abc">
  <unit id="1">
    <res:resourceData>
      <res:resourceItem id="r1" context="no">
        <res:source>
          <abc:resourceType>button</abc:resourceType>
          <abc:resourceHeight>40</abc:resourceHeight>
          <abc:resourceWidth>75</abc:resourceWidth>
        </res:source>
        <res:target>
```

```
          <abc:resourceType>button</abc:resourceType>
          <abc:resourceHeight>40</abc:resourceHeight>
          <abc:resourceWidth>150</abc:resourceWidth>
        </res:target>
      </res:resourceItem>
    </res:resourceData>
    <segment id="1" state="translated">
      <source>Load Registry Config</source>
      <target>Registrierungskonfiguration laden</target>
    </segment>
  </unit>
</file>
```

In this example, the <resourceData> module references multiple static images that an editor can make use of as context while translating or reviewing.

```
<file id="f3">
  <res:resourceData>
    <res:resourceItem id="r1" mimeType="image/jpeg" context="yes">
      <res:source xml:lang="en-us"
          href="resources\en\registryconfig1.resources.jpg" />
      <res:target xml:lang="lb-lu"
          href="resources\lb\registryconfig1.resources.jpg" />
      <res:reference xml:lang="de-de"
          href="resources\de\registryconfig1.resources.jpg" />
    </res:resourceItem>
    <res:resourceItem id="r2" mimeType="image/jpeg" context="yes">
      <res:source xml:lang="en-us"
          href="resources\en\registryconfig2.resources.jpg" />
      <res:target xml:lang="lb-lu"
          href="resources\lb\registryconfig2.resources.jpg" />
    </res:resourceItem>
  </res:resourceData>
  <unit id="1">
    <res:resourceData>
      <res:resourceItemRef ref="r1" />
      <res:resourceItemRef ref="r2" />
    </res:resourceData>
    <segment id="1" state="translated">
      <source>Remove Registry Config</source>
      <target>Registrierungskonfiguration entfernen</target>
    </segment>
  </unit>
</file>
```

Change Tracking Extension (Informative)

Introduction

The Change Tracking extension is used to store revision information for XLIFF elements and attributes. The Change Tracking extension is in place as informative material until the TC will be able to replace it with a revised Change Tracking Module hopefully for XLIFF Version 2.2.

Module Namespace and Validation Artifacts

The namespace for the Change Tracking extension is: `urn:oasis:names:tc:xliff:changetracking:2.0`

Schema for this extension is available at http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/informativeCopiesOf3rdPartySchemas/extensions/change_tracking.xsd.

Module Fragment Identification Prefix

The fragment identification prefix for the Change Tracking module or extension is: `ctr`

Module Elements

The elements defined in the Change Tracking extension are: `<changeTrack>`, `<revisions>`, `<revision>`, and `<item>`.

Tree Structure

Legend:

\+ = one or more

```
<changeTrack>
|
+---<revisions> +
  |
  +---<revision> +
    |
    +---<item> +
```

changeTrack

Parent container for change tracking information associated with a sibling element, or a child of a sibling element, to the change track extension within the scope of the enclosing element.

Contains:

- One or more `<revisions>` elements.

Parents:

- `<file>`
- `<group>`
- `<unit>`

revisions

Container for logical groups of revisions associated with a sibling element, or a child of a sibling element, to the change track extension within the scope of the enclosing element.

Contains:

- One or more `<revision>` elements.

Attributes:

- `appliesTo`, *required*
- `ref`, *optional*
- `currentVersion`, *optional*
- attributes from other namespaces, *optional*

Processing Requirements

- Modifying agents *may* create `<revisions>` elements with attributes.
- Modifying agents *should not* modify `<revisions>` and its attributes defined in this extension, except in the case where the `currentVersion` attribute is used. This attribute *should* be updated when a new revision becomes the most current.
- Modifying agents *should not* remove `<revisions>` and its attributes defined in this extension.
- When the `appliesTo` attribute refers to an element that is a multiple instance within the enclosing element, the `ref` attribute *must* be used to reference an individual instance if and only if the referenced instance has an id. Using `<notes>` as an example:

```
<notes>
  <note id="n1">new note</note>
  <note id="n2">another note</note>
</notes>
<ctr:changeTrack>
  <ctr:revisions appliesTo="note" ref="n1">
    <ctr:revision>
      <ctr:item property="content">old note</ctr:item>
    </ctr:revision>
  </ctr:revisions>
</ctr:changeTrack>
```

revision

Container for specific revisions associated with a sibling element, or a child of a sibling element, to the change track extension within the scope of the enclosing element.

Contains:

- One or more `<item>` elements.

Attributes:

- `author`, *optional*
- `datetime`, *optional*
- `version`, *optional*
- attributes from other namespaces, *optional*

Processing Requirements

- Modifying agents *may* create `<revision>` elements with attributes.
- Modifying agents *should not* modify `<revision>` and its attributes defined in this extension.
- Modifying agents *may* remove `<revision>` and its attributes defined in this extension if and only if there is more than one instance of `<revision>` present. For example, a user agent can decide to preserve only the most current revision.

item

Container for a specific revision associated with a sibling element, or a child of a sibling element, to the change track extension within the scope of the enclosing element.

Contains:

- Text

Attributes:

- `property`, *required*
- attributes from other namespaces, *optional*

Processing Requirements

- Modifying agents *may* create `<item>` elements with attributes.
- Modifying agents *should not* modify `<item>` and its attribute defined in this extension.
- Modifying agents *should not* remove `<item>` and its attribute defined in this extension, unless they are removed as part of a `<revision>` element removed according to its own processing requirements.

Module Attributes

The attributes defined in the Change Tracking extension are: `appliesTo`, `author`, `currentVersion`, `datetime`, `ref`, `property`, and `version`.

appliesTo

appliesTo – Indicates a specific XLIFF element which is a sibling, or a child of a sibling element, to the change track extension within the scope of the enclosing element.

Value description: NMTOKEN name of any valid XLIFF element which is a sibling, or a child of a sibling element, to the change track extension within the scope of the enclosing element.

Default value: undefined

Used in:`<revisions>`

author

author - Indicates the user or agent that created or modified the referenced element or its attributes.

Value description: Text.

Default value: undefined

Used in:<revision>.

currentVersion

currentVersion - holds a reference to the most current version of a revision.

Value description: An [XML Schema Datatypes] NMTOKEN

Default value: none

Used in:<revisions>.

Constraints

- The value of the currentVersion attribute *must* be the value of the version attribute of one of the <revision> elements listed in the same <revisions> element.

datetime

Date and Time, datetime - Indicates the date and time the referenced element or its attributes were created or modified.

Value description: Date in one of the formats defined in [NOTE-datetime].

Default value: undefined

Used in:<revision>.

ref

Reference - Holds a reference to a single instance of an element that has multiple instances within the enclosing element.

Value description: An [XML Schema Datatypes] NMTOKEN

Default value: undefined

Used in:<revisions>

Constraints

- The value of the ref attribute *must* be the value of the id attribute of a single instance of an element that is a multiple instance within the enclosing element.

property

property – Indicates the type of revision data.

Value description: The value *must* be either content to signify the content of an element, or the name of the attribute relating to the revision data.

Default value: none

Used in:<item>.

version

version - Indicates the version of the referenced element or its attributes that were created or modified.

Value description: NMTOKEN.

Default value: undefined

Used in:<revision>.

Example

The following example shows change tracking for <source>, <target>, and <notes>. Current and previous versions are both stored in the Change Tracking extension.

```
<unit id="1">
<ctr:changeTrack>
  <ctr:revisions appliesTo="source" currentVersion="r1">
    <ctr:revision author="system" datetime="2013-07-15T10:00:00+8:00"
        version="r1">
      <ctr:item property="content">Hello World</ctr:item>
    </ctr:revision>
    <ctr:revision author="system" datetime="2013-06-15T10:00:00+8:00"
        version="r2">
      <ctr:item property="content">Hello</ctr:item>
    </ctr:revision>
    <ctr:revision author="system" datetime="2013-05-15T10:00:00+8:00"
        version="r3">
      <ctr:item property="content">Hi</ctr:item>
    </ctr:revision>
  </ctr:revisions>
  <ctr:revisions appliesTo="target" currentVersion="r1">
    <ctr:revision author="Frank" datetime="2013-07-17T11:00:00+8:00"
        version="r1">
      <ctr:item property="content">Guten Tag Welt</ctr:item>
    </ctr:revision>
    <ctr:revision author="Frank" datetime="2013-06-17T11:00:00+8:00"
        version="r2">
      <ctr:item property="content">Hallo</ctr:item>
    </ctr:revision>
    <ctr:revision author="Frank" datetime="2013-05-17T11:00:00+8:00"
        version="r3">
      <ctr:item property="content">Grüsse</ctr:item>
    </ctr:revision>
  </ctr:revisions>
  <ctr:revisions appliesTo="note" ref="n1" currentVersion="r1">
    <ctr:revision author="Bob" datetime="2013-07-16T10:30:00+8:00"
        version="r1">
      <ctr:item property="content">The translation should be formal
      </ctr:item>
      <ctr:item property="category">instruction</ctr:item>
    </ctr:revision>
    <ctr:revision author="Bob" datetime="2013-05-16T10:30:00+8:00"
        version="r2">
      <ctr:item property="content">The translation should be informal
```

```
          </ctr:item>
          <ctr:item property="category">comment</ctr:item>
        </ctr:revision>
      </ctr:revisions>
      <ctr:revisions appliesTo="note" ref="n2" currentVersion="r1">
        <ctr:revision author="Bob" datetime="2013-07-18T10:30:00+8:00"
            version="r1">
          <ctr:item property="content">Please Review my translation
          </ctr:item>
        </ctr:revision>
      </ctr:revisions>
    </ctr:changeTrack>
    <notes>
      <note category="instruction" id="n1">The translation should be
      formal</note>
      <note category="comment" id="n2">Please Review my translation</note>
    </notes>
    <segment>
      <source>Hello World</source>
      <target>Guten Tag Welt</target>
    </segment>
  </unit>
```

Size and Length Restriction Module

Introduction

The Size and Length Restriction module provides a mechanism to annotate the XLIFF content with information on storage and general size restrictions.

The restriction framework has support for two distinct types of restrictions; storage size restrictions and general size restriction. The reason for this is that it is often common to have separate restrictions between storage and display / physical representation of data. Since it would be impossible to define all restrictions here a concept of restriction profile is introduced. The profiles for storage size and general size are independent. The information related to restriction profiles are stored in the processing invariant part of the XLIFF file like the `<xlf:file>`, `<xlf:group>` and `<xlf:unit>` elements and contained within elements defined in this module. The information regarding the specific restrictions are stored on the processing invariant parts and on the inline elements as attributes or attributes referencing data in the elements defined in this module. To avoid issues with segmentation no information regarding size restrictions is present on `<xlf:segment>`, `<xlf:source>` and `<xlf:target>` elements. The module defines a namespace for all the elements and attributes it introduces, in the rest of the module specification elements and attributes are in this namespace unless stated otherwise. In other parts of the XLIFF specification the prefix "slr" is used to refer to this module's namespace. For clarity the prefix "xlf" will be used for *XLIFF Core* elements and attributes. Profile names use the same namespace-like naming convention as user defined values in the *XLIFF Core* specification. The names *should* be composed of two components separated by a colon. <authority>:<name>. The authority "xliff" is reserved for profiles defined by the OASIS XLIFF Technical Committee.

Module Namespace and Validation Artifacts

The namespace for the Size and Length restriction module is: `urn:oasis:names:tc:xliff:sizerestriction:2.0`

Schema and Schematron for this module are available at http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/size_restriction.xsd and http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/size_restriction.sch.

Module Fragment Identification Prefix

The fragment identification prefix for the Size and Length restriction module is: `slr`

Module Elements

The elements defined in the Size and Length restriction module are: `<profiles>`, `<normalization>` and `<data>`.

Tree Structure

Legend:

? = zero or one

```
<profiles>
|
+---<normalization> ?
|
+---<other> *
```

profiles

This element selects the restriction profiles to use in the document. If no storage or general profile is specified the default values (empty) of those elements will disable restriction checking in the file.

Contains:

- Zero or one `<normalization>` element followed by
- elements from other namespaces, *optional*

Attributes:

- `generalProfile`, *optional*
- `storageProfile`, *optional*

Processing Requirements

- Any overall configuration or settings related to the selected profile *must* be placed in child elements of this element.
- Data not related to the configuration of the selected profiles *must not* be placed in this element.

normalization

This element is used to hold the attributes specifying the normalization form to apply to storage and size restrictions defined in the standard profiles.

Contains:

This element is always empty.

Attributes:

- `general`, *optional*
- `storage`, *optional*

Processing Requirements

- If this element is not present no normalization *should* be performed for the standard profiles.
- Other profiles *may* use this element in its specified form but *must not* add new extensions to it.

data

This elements act as a container for data needed by the specified profile to check the part of the XLIFF document that is a sibling or descendant of a sibling of this element. It is not used by the default profiles.

Contains:

- elements from other namespaces, *optional*

Attributes:

- `profile`, *required*
- attributes from other namespaces, *optional*

Processing Requirements

- Third party profiles *must* place all data in this element instead of using other extension points if the data serves no other purpose in the processing of the document.
- Data not used by the specified profile *must not* be placed in this element.

Module Attributes

The attributes defined in the Size and Length restriction module are: `storageProfile`, `generalProfile`, `storage`, `general`, `profile`, `storageRestriction`, `sizeRestriction`, `equivStorage` , `sizeInfo` and `sizeInfoRef`.

storageProfile

This attribute specifies, which profile to use while checking storage size restrictions. Empty string means that no restrictions are applied.

Value description: Name of restriction profile to use for storage size restrictions.

Default value: empty string

Used in:`<profiles>`.

generalProfile

This attribute specifies, which profile to use while checking the general size restrictions. Empty string means that no restrictions apply.

Value description: Name of restriction profile to use for general size restrictions.

Default value: empty string

Used in:`<profiles>`.

storage

This attribute specifies the normalization form to apply for storage size restrictions. Only the normalization forms C and D as specified by the Unicode Consortium are supported, see Unicode Standard Annex #15 [http://unicode.org/reports/tr15/].

Value description: normalization to apply.

Table 5. Values

Value	Description
none	No additional normalization *should* be done, content *should* be used as represented in the document. It is possible that other *Agents* have already done some type of normalization when *Modifying* content. This means that this setting could give different results depending on what *Agents* are used to perform a specific action on the *XLIFF Document*.
nfc	Normalization Form C *must* be used
nfd	Normalization Form D *must* be used

Default value: `none`

Used in: `<normalization>`.

general

This attribute specifies the normalization to apply for general size restrictions. Only the normalization forms C and D as specified by the Unicode Consortium are supported, see Unicode Standard Annex #15 [http://unicode.org/reports/tr15/].

Value description: normalization to apply.

Table 6. Values

Value	Description
none	No additional normalization *should* be done, content *should* be used as represented in the document. It is possible that other *Agents* have already done some type of normalization when *Modifying* content. This means that this setting could give different results depending on what *Agents*

Value	Description
	are used to perform a specific action on the *XLIFF Document*.
nfc	Normalization Form C *must* be used
nfd	Normalization Form D *must* be used

Default value: `none`

Used in:`<normalization>`.

profile

This attribute is used on the `<data>` element to indicate what profile the contents of that element apply to.

Value description: Name of a restriction profile

Default value: undefined

Used in:`<data>`.

storageRestriction

This attribute specifies the storage restriction to apply to the collection descendants of the element it is defined on.

Value description: Interpretation of the value is dependent on selected `storageProfile`. It *must* represent the restriction to apply to the indicated sub part of the document.

Default value: undefined

Used in: `<file>`, `<group>`, `<unit>`, `<mrk>`, `<sm>`, `<pc>` and `<sc>`.

sizeRestriction

This attribute specifies the size restriction to apply to the collection descendants of the element it is defined on.

Value description: Interpretation of the value is dependent on selected `generalProfile`. It *must* represent the restriction to apply to the indicated sub part of the document.

Default value: undefined

Used in: `<file>`, `<group>`, `<unit>`, `<mrk>`, `<sm>`, `<pc>` and `<sc>`.

equivStorage

This attribute provides a means to specify how much storage space an inline element will use in the native format. This size contribution is then added to the size contributed by the textual parts. This attribute is only allowed on the `<ec>` element if that element has the `isolated` attribute set to `yes`. Otherwise the attribute on the paired `<sc>` element also cover its partner `<ec>` element.

Value description: Interpretation of the value is dependent on selected `storageProfile`. It *must* represent the equivalent storage size represented by the inline element.

Default value: undefined

Used in: <pc>, <sc>, <ec>, <ph> and

sizeInfo

This attribute is used to associate profile specific information to inline elements so that size information can be decoupled from the native format or represented when the native data is not available in the XLIFF document. It can be used on both inline elements and structural elements to provide information on things like GUI dialog or control sizes, expected padding or margins to consider for size, what font is used for contained text and so on. This attribute is only allowed on the <ec> element if that element has the isolated attribute set to yes. Otherwise the attribute on the paired <sc> element also cover its partner <ec> element.

Value description: Interpretation of the value is dependent on selected generalProfile. It *must* represent information related to how the element it is attached to contributes to the size of the text or entity in which it occurs or represents.

Default value: undefined

Used in: <file>, <group>, <unit>, <pc>, <sc>, <ec>, and <ph>.

Constraints

- This attribute *must not* be specified if and only if sizeInfoRef is used. They *must not* be specified at the same time.

sizeInfoRef

This attribute is used to point to data that provide the same function as the sizeInfo attribute does, but with the data stored outside the inline content of the XLIFF segment. This attribute is only allowed on the <ec> element if that element has the isolated attribute set to yes. Otherwise the attribute on the paired <sc> element also cover its partner <ec> element.

Value description: a reference to data that provide the same information that could be otherwise put in a sizeInfo attribute. The reference *must* point to an element in a <data> element that is a sibling to the element this attribute is attached to or a sibling to one of its ancestors.

Default value: undefined

Used in: <file>, <group>, <unit>, <pc>, <sc>, <ec>, and <ph>,

Constraints

- This attribute *must not* be specified if and only if sizeInfo is used. They *must not* be specified at the same time.

Standard profiles

General restriction profile "xliff:codepoints"

This profile implements a simple string length restriction based on the number of Unicode code points. It is *optional* to specify if normalization is to be applied using the <normalization> element and the general attribute. This profile makes use of the following attributes from this module:

sizeRestriction

The value of this attribute holds the "maximum" or "minimum and maximum" size of the string. Either size *must* be an integer. The maximum size *may* also be '*' to denote that there is no maximum restriction. If only a maximum is specified it is implied that the minimum is 0 (empty string). The format of the value is the *optional* minimum size and a coma followed by a maximum size ("[minsize,]maxsize"). The default value is '*' which evaluates to a string with unbounded size.

sizeInfo

The value of this attribute is an integer representing how many code points the element it is set on is considered to contribute to the total size. If empty, the default for all elements is 0.

Storage restriction profiles "xliff:utf8", "xliff:utf16" and "xliff:utf32"

These three profiles define the standard size restriction profiles for the common Unicode character encoding schemes. It is *optional* to specify if normalization is to be applied using the `<normalization>`element and the `storage`. All sizes are represented in 8bit bytes. The size of text for these profiles is the size of the text converted to the selected encoding without any byte order marks attached. The encodings are specified by the Unicode Consortium in chapter 2.5 of the Unicode Standard [http://www.unicode.org/versions/Unicode6.2.0/ch02.pdf] [Unicode].

Table 7. Profiles

Name	Description
xliff:utf8	The number of 8bit bytes needed to represent the string encoded as UTF-8 as specified by the Unicode consortium.
xliff:utf16	The number of 8bit bytes needed to represent the string encoded as UTF-16 as specified by the Unicode consortium.
xliff:utf32	The number of 8bit bytes needed to represent the string encoded as UTF-32 as specified by the Unicode consortium.

These profiles make use of the following attributes from this module:

storageRestriction

The value of this attribute holds the "maximum" or "minimum and maximum" size of the string. Either size *must* be an integer. The maximum size *may* also be '*' to denote that there is no maximum restriction. If only a maximum is specified it is implied that the minimum is 0 (empty string). The format of the value is the *optional* minimum size and a coma followed by a maximum size ("[minsize,]maxsize"). The default value is '*' which evaluates to a string with unbounded size.

equivStorage

The value of this attribute is an integer representing how many bytes the element it is set on is considered to contribute to the total size. If empty the default is 0. The `<cp>` is always converted to its representation in the profiles encoding and the size of that representation is used as the size contributed by the `<cp>`.

Third party profiles

The general structure of this module together with the extensibility mechanisms provided has been designed with the goal to cater for all practically thinkable size restriction schemes. For example, to represent two dimensional data, a profile can adopt a coordinate style for the values of the general restriction attributes. For instance `{x,y}` to represent width and height, or `{{x1,y1},{x2,y2}}` to represent a bounding box. It is also possible to embed information necessary to drive for instance a display simulator and attach that data to text in order to be able to perform device specific checking. Providing font information and checking glyph based general size are other feasible options.

Conformance

To claim conformance to the XLIFF size and length restriction module an *Agent must* meet the following criteria:

- *must* be compliant with the schema of the *XLIFF Core* specification and its extensions provided in this module.
- *must* follow all processing requirements set forth in this module specification regarding the general use of elements and attributes.
- *must* support all standard profiles with normalization set to `none`.
- *should* support all standard profiles with all modes of normalization.
- *may* support additional third party profiles for storage or general restrictions.
- *must* provide at least one of the following:
  - add size and length restriction information to an *XLIFF Document*
  - if it supports the profile(s) specified in the *XLIFF Document* it *must* provide a way to check if the size and length restrictions in the document are met according to the profile(s) requirements.

Example

A short example on how this module can be used is provided here with inline XML comments explaining the usage of the module features.

```
<xliff version="2.0" srcLang="en-us"
    xmlns="urn:oasis:names:tc:xliff:document:2.0"
    xmlns:slr="urn:oasis:names:tc:xliff:sizerestriction:2.0">
  <file id="f1">
    <slr:profiles generalProfile="xliff:codepoints"
        storageProfile="xliff:utf8">
      <!-- Select standard UTF-8 storage encoding and standard codepoint
          size restriction both with NFC normalization-->
      <slr:normalization general="nfc" storage="nfc" />
    </slr:profiles>
    <!-- The group should not require more than 255 bytes of storage And
        have at most 90 codepoints. Note that the sum of the unit sizes
        are larger than this the total content of the group must still
        be at most 90 codepoints. -->
    <group id="g1" slr:storageRestriction="255" slr:sizeRestriction="90">
      <!-- This unit must not contain more than 60 code points -->
      <unit id="u1" slr:sizeRestriction="60">
        <segment>
          <!-- The spanning <pc> element require 7 bytes of storage in the
              native format. Its content must not have more than 25
              codepoints -->
```

```
            <source>This is a small <pc equivStorage="7"
                slr:sizeRestriction="25">size restriction</pc>
                example. </source>
          </segment>
        </unit>
        <!-- This unit must not have more than 35 codepoints -->
        <unit id="u2" slr:sizeRestriction="35">
          <segment>
            <source>With a group structure.</source>
          </segment>
        </unit>
      </group>
    </file>
</xliff>
```

Validation Module

Introduction

This module defines a specific set of validation rules that can be applied to target text both globally and locally. Further constraints can be defined that allow rules to be applied to target text based on conditions in the source text or disabled to override a global scope.

Module Namespace and Validation Artifacts

The namespace for the Validation module is: `urn:oasis:names:tc:xliff:validation:2.0`

Schema and Schematron for this module are available at http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/validation.xsd and http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/validation.sch.

Module Fragment Identification Prefix

The fragment identification prefix for the Validation module is: `val`

Module Elements

The elements defined in the Validation module are: `<validation>` and `<rule>`.

Tree Structure

Legend:

\+ = one or more

```
<validation>
|
+---<rule> +
```

validation

Parent container for a list of rules and constraints to apply to the target text of the enclosing element.

Contains:

- One or more `<rule>` elements.

Attributes:

- attributes from other namespaces, *optional*

Processing Requirements

- When the `<validation>` element occurs at the `<file>` level, rules *must* be applied to all `<target>` elements within the scope of that `<file>` element, except where overrides are specified at the `<group>` or `<unit>` level.
- When `<validation>` occurs at the `<group>` level, rules *must* be applied to all `<target>` elements within the scope of that `<group>`, except where overrides are specified in a nested `<group>` element, or at the `<unit>` level.
- When `<validation>` occurs at the `<unit>` level, rules *must* be applied to all `<target>` elements within the scope of that `<unit>`.

rule

A specific rule and constraint to apply to the target text of the enclosing element.

Contains:

- This element is always empty.

Attributes:

- `isPresent`, *optional*
- `occurs`, *optional*
- `isNotPresent`, *optional*
- `startsWith`, *optional*
- `endsWith`, *optional*
- `existsInSource`, *optional*
- `caseSensitive`, *optional*
- `normalization`, *optional*
- `disabled`, *optional*
- attributes from other namespaces, *optional*

Constraints

- Exactly one of the following attributes:
  - `isPresent`
  - `isNotPresent`
  - `startsWith`

- `endsWith`

- a custom rule defined by attributes from any namespace
is *required* in any one `<rule>` element.

Processing Requirements

- *Writers may* create and add new `<rule>` elements, provided that the new rules do not contradict rules already present.
- *Modifiers must not* change attributes defined in this module that are already present in any `<rule>` element.
- *Modifiers must not* remove either `<rule>` elements or their attributes defined in this module.

Module Attributes

The attributes defined in the Validation module are: `isPresent`, `occurs`, `isNotPresent`, `startsWith`, `endsWith`, `existsInSource`, `caseSensitive`, `normalization`, and `disabled`.

isPresent

This rule attribute specifies that a string *must* be present in the target text *at least once*.

For example, the following is valid:

```
<unit id="1">
  <val:validation>
    <val:rule isPresent="online" />
  </val:validation>
  <segment id="1">
    <source>Choose an option in the online store.</source>
    <target>Escolha uma opção na loja online.</target>
  </segment>
</unit>
```

Whereas the following is invalid:

```
<unit id="1">
  <val:validation>
    <val:rule isPresent="loja" />
  </val:validation>
  <segment id="1">
    <source>Choose an option in the online store.</source>
    <target>Escolha uma opção na online store.</target>
  </segment>
</unit>
```

Other rule attributes can be combined with `isPresent` to produce the following results:

isPresent=”loja” - loja is found in the target text at least once.
isPresent=”loja” occurs=”1” - loja is found in the target text exactly once.

isPresent=”loja” existsInSource=”yes” - loja is found in both source and target text the same number of times.
isPresent=”loja” existsInSource=”yes” occurs=”1” - loja is found in both source and target text and occurs in target text exactly once.

Value description: Text.

Default value: none

Used in: `<rule>`

occurs

This rule attribute is used with the `isPresent` rule attribute to specify the exact number of times a string *must* be present in the target text. When this rule attribute is not used, then the string *must* be present in the target text *at least once*.

For example, the following is valid:

```
<unit id="1">
  <val:validation>
    <val:rule isPresent="loja" occurs="2" />
  </val:validation>
  <segment id="1">
    <source>Choose a store option in the online store.</source>
    <target>Escolha uma opção de loja na loja online.</target>
  </segment>
</unit>
```

Whereas the following is invalid:

```
<unit id="1">
  <val:validation>
    <val:rule isPresent="loja" occurs="2" />
  </val:validation>
  <segment id="1">
    <source>Choose a store option in the online store.</source>
    <target>Escolha uma opção de loja na online store.</target>
  </segment>
</unit>
```

Value description: A number of 1 or greater.

Default value: none

Used in: `<rule>`

isNotPresent

This rule attribute specifies that a string *must not* be present in the target text.

For example, the following is valid:

```
<unit id="1">
  <val:validation>
```

```
    <val:rule isNotPresent="store" />
  </val:validation>
  <segment id="1">
    <source>Choose an option in the online store.</source>
    <target>Escolha uma opção na loja online.</target>
  </segment>
</unit>
```

Whereas the following is invalid:

```
<unit id="1">
  <val:validation>
    <val:rule isNotPresent="store" />
  </val:validation>
  <segment id="1">
    <source>Choose an option in the online store.</source>
    <target>Escolha uma opção na online store.</target>
  </segment>
</unit>
```

Value description: Text.

Default value: none

Used in: `<rule>`

startsWith

This rule attribute specifies that a string *must* start with a specific value.

For example, the following is valid:

```
<unit id="1">
  <val:validation>
    <val:rule startsWith="*" />
  </val:validation>
  <segment id="1">
    <source>*Choose an option in the online store.</source>
    <target>*Escolha uma opção na loja online.</target>
  </segment>
</unit>
```

Whereas the following is invalid:

```
<unit id="1">
  <val:validation>
    <val:rule startsWith="*" />
  </val:validation>
  <segment id="1">
    <source>*Choose an option in the online store.</source>
    <target>Escolha uma opção na loja online.</target>
  </segment>
```

```
</unit>
```

Value description: Text.

Default value: none

Used in: `<rule>`

endsWith

This rule attribute specifies that a string *must* end with a specific value.

For example, the following is valid:

```
<unit id="1">
  <val:validation>
    <val:rule endsWith=":" />
  </val:validation>
  <segment id="1">
    <source>Choose an option in the online store:</source>
    <target>Escolha uma opção na loja online:</target>
  </segment>
</unit>
```

Whereas the following is invalid:

```
<unit id="1">
  <val:validation>
    <val:rule endsWith=":" />
  </val:validation>
  <segment id="1">
    <source>Choose an option in the online store:</source>
    <target>Escolha uma opção na online store.</target>
  </segment>
</unit>
```

Value description: Text.

Default value: none

Used in: `<rule>`

existsInSource

When this rule attribute is used with another rule attribute and is set to `yes`, it specifies that for the rule to succeed, the condition *must* be satisfied in both source and target text. This rule attribute is valid only when used with one of the following rule attributes: `isPresent`, `startsWith`, or `endsWith`.

When `existsInSource` is set to `no`, it will have no impact on execution of rules, except for overriding rules where `existsInSource` is set to `yes` on a higher level.

For example, the following are valid:

```
<unit id="1">
  <val:validation>
    <val:rule endsWith=":" existsInSource="yes" />
  </val:validation>
  <segment id="1">
    <source>Choose an option in the online store:</source>
    <target>Escolha uma opção na loja online:</target>
  </segment>
</unit>
...
<unit id="2">
  <val:validation>
    <val:rule endsWith=":" existsInSource="no" />
  </val:validation>
  <segment id="1">
    <source>Choose an option in the online store.</source>
    <target>Escolha uma opção na loja online:</target>
  </segment>
</unit>
```

Whereas the following is invalid:

```
<unit id="1">
  <val:validation>
    <val:rule endsWith=":" existsInSource="yes" />
  </val:validation>
  <segment id="1">
    <source>Choose an option in the online store.</source>
    <target>Escolha uma opção na loja online:</target>
  </segment>
</unit>
```

Value description: `yes` or `no`

Default value: `no`

Used in: `<rule>`

Constraints

- When `existsInSource` is specified, exactly one of
  - `isPresent`
  - `startsWith`
  - `endsWith`

  is *required* in the same `<val:rule>` element.

caseSensitive

This rule attribute specifies whether the test defined within that rule is case sensitive or not.

Value description: `yes` if the test is case sensitive, `no` if the test is case insensitive.

Default value: `yes`.

Used in: `<rule>`

normalization

This rule attribute specifies the normalization type to apply when validating a rule. Only the normalization forms C and D as specified in [UAX #15].

Value description: The allowed values are listed in the table below along with their corresponding types of normalization to be applied.

Table 8. Values

Value	Description
none	No normalization *should* be done.
nfc	Normalization Form C *must* be used.
nfd	Normalization Form D *must* be used.

Default value: `nfc`

Used in: `<rule>`

disabled

This rule attribute determines whether a rule *must* or *must not* be applied within the scope of its enclosing element. For example, a rule defined at the `<file>` level can be disabled at the `<unit>` level.

This attribute is provided to allow for overriding execution of rules set at higher levels, see `<val:validation>`.

In the following example, the isNotPresent rule is applied in its entirety to the first unit, but not to the second.

```
<file id="f1">
  <val:validation>
    <val:rule isPresent="store" />
  </val:validation>
  <unit id="1">
    <segment id="1">
      <source>Choose an option in the online store:</source>
      <target>Escolha uma opção na loja online:</target>
    </segment>
  </unit>
  <unit id="2">
    <val:validation>
      <val:rule isPresent="store" disabled="yes" />
    </val:validation>
    <segment id="1">
      <source>Choose an option in the application store:</source>
      <target>Escolha uma opção na application store:</target>
    </segment>
  </unit>
```

```
</file>
```

Value description: `yes` or `no`

Default value: `no`

Used in: `<rule>`

ITS Module

Introduction

This module defines Inline Annotations (normative usage descriptions for attributes on inline annotation markers), attributes and elements that are needed to map [ITS] data categories using only *XLIFF-defined* elements and attributes. The module also defines an external rules file to be used by generic ITS processors working with *XLIFF Documents*. This module only defines attributes and annotations that are not available through *XLIFF Core* or other *Modules*. This module specification also contains normative provisions for mapping of [ITS] data categories and features that are available via XLIFF *Core* and other modules (ITS data categories available through XLIFF Core and other Modules and ITS data categories that have a partial overlap with XLIFF features) or other *Modules* outside of the ITS Module (ITS data categories that have a partial overlap with XLIFF features). Finally an overview of data categories is provided where the information is or can be fully expressed by *Extraction* behavior and therefore those categories or their parts (sub-categories) cannot be represented as metadata within *XLIFF Documents* (ITS data categories that do not represent metadata after Extraction of content into XLIFF).

Note

This module specification chiefly describes how the [ITS] data categories need to be expressed within *XLIFF Documents*. Some data categories are typically *Extracted* from native source formats, others would be first injected into *XLIFF Documents* by *Enriching Agents* and might be useful or not in the target content after *Merging* back to the native format in the target natural language. For all ITS data categories that can be encoded within *XLIFF Documents*, there is an important XLIFF specific distinction between structural and inline elements in XLIFF. Some categories can only be expressed inline in *XLIFF Documents*. Others can be also expressed on structural markup levels; in such a case, inheritance (or not) from the XLIFF structural levels is important. Nevertheless, even the inline only ITS data categories can be in scope of ITS Tools Referencing that can be set on structural levels and possible inheritance of relevant `its:annotatorsRef` values needs to be always checked by implementers.

Warning

There is an important scope difference between attributes from the namespace `http://www.w3.org/2005/11/its` as implemented in *XLIFF Documents* and as defined in the [ITS] specification itself. This affects all ITS attributes that can be used on inline spans. In XLIFF, spans delimited by the well-formed `<mrk>` are always equivalent and interchangeable with pseudo-spans delimited by `<sm/>` / `<em/>` pairs. In many cases delimiting the needed spans by `<mrk>` is impossible due to overlap with other well-formed spans, while delimiting of inline spans with `<sm/>` / `<em/>` pairs is always possible and often preferable as it allows the spans to persist even through a change of segmentation.

However, [ITS] doesn't define a pseudo-span mechanism and thus generic ITS Processors cannot parse pseudo-spans. ITS processors will generally identify ITS attributes or their mappings from XLIFF specific namespaces on the `<sm/>` markers, but they will consider their scope to be the empty marker itself, whereas the true scope of all attributes on such markers within *XLIFF Documents* is between the start marker `<sm id="1"/>` and it's corresponding end marker `<em startRef="1"/>`.

Implementers who wish to better access [ITS] data categories information within *XLIFF Documents* can implement an additional capability in their ITS Processors to detect spans like this one `<sm id="1"/>span of text<em startRef="1"/>` without going into any more XLIFF specific features and becoming full fledged XLIFF *Agents*.

Module Namespace and Validation Artifacts

The namespaces for the ITS module are: `http://www.w3.org/2005/11/its` and `urn:oasis:names:tc:xliff:itsm:2.1`.

Schemas and Schematron for this module are available at http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/its.xsd, http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/itsm.xsd, and http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/its.sch.

Note

Although setting and usage of prefixes for namespaces in XML is arbitrary, we are using `its:` for the `http://www.w3.org/2005/11/its` namespace and `itsm:` for the `urn:oasis:names:tc:xliff:itsm:2.1` namespace throughout this specification.

Module Fragment Identification Prefix

The fragment identification prefix for the ITS module is: `its`.

Note

Although this module has to use two different XML namespace prefixes it uses only one fragment identification and authority prefix which is `its`.

Conformance to the ITS Module Specification

Note

Some ITS data categories like Translate [https://www.w3.org/TR/its20/#trans-datacat] are supported natively by XLIFF. Other data categories are not supported by XLIFF because they are focusing on source content and not XLIFF content. The below conformance statement is only relevant for data categories for which the usage in XLIFF 2.1 is normatively defined in this XLIFF 2.1 ITS Module. Like in the [ITS] 2.0 specification, there is no interrelation between data categories.

Processing Requirements

- Conformant ITS Processors *must* be able to use the external global rules included in the module's Schematron file http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/its.sch and compute ITS data categories encoded in *XLIFF Documents* as per [ITS] Conformance clauses

[https://www.w3.org/TR/its20/#conformance-product-processing-expectations] 2-1, 2-2, 2-3, and 2-4.

- In cases where any of the above specified global rules identifies an empty `<sm>` element, conformant ITS Processors *should* be able identify the closing `<em>` marker and process the relevant ITS data category as applied on the span delimited by those corresponding empty markers.

- Conformant *Agents must* be XLIFF Conformant in the sense of XLIFF Application Conformance and also implement at least one [ITS] data category defined in the section ITS data categories defined in the ITS Module of this ITS Module or provide full support for at least one of the [ITS] custom annotations (ITS Mapping Annotations) specified in the Section ITS data categories that have a partial overlap with XLIFF features.

  In particular:

  - Conformant *Extractors must* be capable of *Extracting* at least one of the above specified ITS data categories from a source format and encode it in a resulting conformant *XLIFF Document* with ITS Module based metadata.

  - Conformant *Enrichers must* be capable of *Enriching XLIFF Documents* with at least one of the above specified ITS data categories.

  - Conformant *Modifiers must* be capable of updating at least one of the above specified ITS data categories according to its own Constraints and Processing Requirements as specified in the ITS Module.

  - Conformant *Mergers must* be capable of *Merging* metadata of at least one of the above specified ITS data categories back to the respective native format (with full knowledge of the *Extraction* mechanism) in the target natural language.

ITS Tools Referencing

[ITS] Tools Annotation [http://www.w3.org/TR/2013/REC-its20-20131029/#its-tool-annotation] mechanism provides a way to record tools that produced [ITS] metadata.

Warning

This mechanism is reserved for recording producers of ITS metadata. General provenance information can be recorded using the Provenance data category mapping defined in this Module. Provenance metadata.

Note

The ITS Tools Referencing mechanism has to be always used with the MT Confidence data category. The Terminology and Text Analysis data categories have to use ITS Tools Referencing conditionally, i.e. whenever they specify `its:termConfidence` or `its:taConfidence` respectively.

With all other [ITS] data categories, there is no express need to use the ITS Tools Referencing mechanism. It is nevertheless advised that the relevant ITS Tooling metadata is *Extracted* where available and *Modified* when the relevant ITS data category information changes during the *XLIFF Document* processing. Finally, all conformant *Agents* and ITS Processors need to be able to compute the ITS Tools Referencing information in case this has been provided by other conformant *Agents* earlier in the workflow as per the ITS Module Conformance section.

Processing Requirements

- *Writers must* use the attribute `its:annotatorsRef` to express the information provided through the [ITS] Tools Annotation [http://www.w3.org/TR/2013/REC-its20-20131029/#its-tool-annotation] mechanism in *XLIFF Documents*.

ITS Tools Annotation

This is used to express the [ITS] Tools Annotation [http://www.w3.org/TR/2013/REC-its20-20131029/#its-tool-annotation] mechanism on inline markers.

Usage:

- The `id` attribute is *required*.
- The `its:annotatorsRef` attribute is *required*.
- The `type` attribute is *optional* and set to `its:generic`.
- The `translate` attribute is *optional*.

ITS data categories defined in the ITS Module

The following [ITS] data categories are fully specified within this module:

- Allowed Characters
- Domain
- Locale Filter
- Localization Quality Issue
- Localization Quality Rating
- Provenance,
- Text Analysis

Allowed Characters

Used to specify the characters that are permitted in a given piece of content. See [ITS] Allowed Characters [https://www.w3.org/TR/its20/#allowedchars] for further details.

Processing Requirements

- *Writers must* use the ITS Allowed Characters Annotation to express the [ITS] Allowed Characters [http://www.w3.org/TR/its20/#allowedchars] data category in *XLIFF Documents*.

  For both structural and inline elements, use `<mrk>` or an `<sm/>` / `<em/>` pair with the following attribute: `its:allowedCharacters`.

  See the ITS Allowed Characters Annotation for the normative usage description of this attribute and the following sections for further details on structural and inline elements.

Structural Elements

If a structural element of the original document has a Allowed Characters annotation, it is recommended to represent that annotation using a `<mrk>` element that encloses the whole content of the `<source>` element.

Example 1. Extraction of Allowed Characters at structural levels

Original:

```
...
<p its-allowed-characters="[a-ZA-Z]">Text</p>
...
```

Extraction:

```
...
<unit id="1">
  <segment>
    <source><mrk id="m1" type="its:generic"
        its:allowedCharacters="[a-ZA-Z]">Text</mrk></source>
  </segment>
</unit>
...
```

Inline Elements

Use the ITS attribute on the `<mrk>` element:

Original:

```
...
<p>user name: <span its-allowed-characters='[a-ZA-Z]'>johnDoe</span></p>
...
```

Extraction:

```
...
<unit id="1">
  <segment>
    <source>user name: <mrk id="m1" type="its:generic"
        its:allowedCharacters="[a-ZA-Z]">johnDoe</mrk>.</source>
  </segment>
</unit>
...
```

ITS Allowed Characters Annotation

This is used to fully map to and from the [ITS] Alllowed Characters [https://www.w3.org/TR/its20/#allowedchars] data category.

Usage:

- The [ITS] defined `its:allowedCharacters` attribute is *required*.
- The `type` attribute is *optional* and set to `its:generic`.

- The `translate` attribute is *optional*.

Domain

Identifies the topic, theme, or subject of the content in scope. See [ITS] Domain [http://www.w3.org/TR/its20/#domain] for details.

Processing Requirements

- *Writers must* use the attribute `itsm:domains` to express the [ITS] Domain [http://www.w3.org/TR/its20/#domain] data category in *XLIFF Documents*.

Warning

Please note that the Domain data category uses the `itsm:domains` attribute that belongs to the `urn:oasis:names:tc:xliff:itsm:2.1` namespace (prefixed with `itsm:`) and not to the `http://www.w3.org/2005/11/its` (prefixed with `its`) as most of the other attributes described in this module.

Structural Elements

Example 2. Extraction of Domain at structural levels

Original:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Data Category: Domain</title>
    <script type="application/its+xml">
      <its:rules xmlns:its="http://www.w3.org/2005/11/its" version="2.0"
          xmlns:h="http://www.w3.org/1999/xhtml">
        <its:domainRule selector="//h:*[@class='dom1']"
            domainPointer="./@class" domainMapping="dom1 domain1" />
      </its:rules>
    </script>
  </head>
  <body>
    <p class="dom1">Text in the domain domain1</p>
  </body>
</html>
```

Extraction:

```
...
<unit id='2' itsm:domains="domain1">
  <segment>
    <source>Text in the domain domain1</source>
  </segment>
</unit>
...
```

Inline Elements

Use `<mrk>` or an `<sm/>` / `<em/>` pair with the `itsm:domains` attribute set.

See the ITS Domain Annotation for the normative usage description on inline markers.

ITS Domain Annotation

This is used to express inline the [ITS] Domain [http://www.w3.org/TR/its20/#domain] data category.

Usage:

- The `id` attribute is *required*.
- The `itsm:domains` attribute is *required*.
- The `type` attribute is *optional* and set to `its:generic`.
- The `translate` attribute is *optional*.

Example 3. Extraction of Domain metadata on inline elements

Original:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Data Category: Domain</title>
    <script type="application/its+xml">
      <its:rules xmlns:its="http://www.w3.org/2005/11/its" version="2.0"
          xmlns:h="http://www.w3.org/1999/xhtml">
        <its:domainRule selector="//h:*[@class='dom1']"
            domainPointer="./@class" domainMapping="dom1 domain1" />
      </its:rules>
    </script>
  </head>
  <body>
    <p>Span of text <span class="dom1">in the domain domain1</span></p>
  </body>
</html>
```

Extraction:

```
...
<unit id="1">
  <segment>
    <source>Span of text <pc id="1"><mrk id="m1" type="its:generic"
        itsm:domains="domain1" >in the domain domain1</mrk></pc></source>
  </segment>
</unit>
...
```

Locale Filter

Expresses that a node is only applicable to certain locales. See [ITS] Locale Filter [https://www.w3.org/TR/its20/#LocaleFilter] for further details.

This section describes how the Locale Filter [https://www.w3.org/TR/its20/#LocaleFilter] information can be represented inline in *XLIFF Documents* if necessary. However, it is preferable that this data category is fully consumed by *Extraction*/*Merge* behavior as *recommended* in the section on ITS data categories that are not explicitly represented in *XLIFF Documents*.

Processing Requirements

- *Writers must* use the ITS Locale Filter Annotation to express the [ITS] Locale Filter [http://www.w3.org/TR/its20/#LocaleFilter] data category in *XLIFF Documents* that don't have set the target locale.
- *Writers must* use the *XLIFF Core* Translate Annotation to express the [ITS] Locale Filter [http://www.w3.org/TR/its20/#LocaleFilter] data category in *XLIFF Documents* with the target locale set.
- *Modifiers must* remove the ITS Locale Filter Annotation and replace it with the *XLIFF Core* Translate Annotation when setting the `trgLang` or when receiving an *XLIFF Documents* with `trgLang` set.

Warning

Core only *Modifiers* might have invalidated the ITS Locale Filter Annotation by setting the `trgLang`. Although, this is addressed by the above PR, [ITS] Locale Filter [https://www.w3.org/TR/its20/#LocaleFilter] capable *Modifiers* are strongly advised to better set the `trgLang` as soon as known and perform the above specified annotations' transformation rather than to assume that other tools downstream will be capable of interpreting the [ITS] Locale Filter [https://www.w3.org/TR/its20/#LocaleFilter] metadata when setting the target locale.

For both structural and inline elements, use `<mrk>` or an `<sm/>` / `<em/>` pair with the following attributes: `its:localeFilterList` and `its:localeFilterType`.

See the ITS Locale Filter Annotation for the normative usage description of those attributes and the following sections for further details on structural and inline elements.

Structural Elements

When the target locale in XLIFF is undefined, the locale filter data category *may* be *Extracted* using the ITS Locale Filter Annotation.

Example 4. Extraction of Locale Filter at structural levels

Original:

```
<p its-locale-filter-list='fr'>Text A</p>
<p its-locale-filter-list='ja'>Text B</p>
```

Extraction:

```
<xliff srcLang="en" version="2.1">
  ...
  <unit id="1">
    <segment>
      <source><sm id=1 its:localeFilterList="fr"/>Text A<em
```

```
          startRef="1"/></source>
    </segment>
  </unit>
  <unit id="2">
    <segment>
      <source><sm id="1" its:localeFilterList="ja"/>Text B<em
          startRef="1"/></source>
    </segment>
  </unit>
  ...
```

When the target locale in XLIFF is defined, use the `translate` attribute. (`yes` if the target locale applies, `no` if it does not).

Original:

```
<p its-locale-filter-list='fr'>Text A</p>
<p its-locale-filter-list='ja'>Text B</p>
```

Extraction:

```
<xliff srcLang="en" trgLang="fr" version="2.1">
  ...
  <unit id="1" translate="yes">
    <segment>
      <source>Text A</source>
    </segment>
  </unit>
  <unit id='2' translate="no">
    <segment>
      <source>Text B</source>
    </segment>
  </unit>
```

Inline Elements

When the target locale in XLIFF is undefined, use the `<mrk>` or an `<sm/>` / `<em/>` pair with the original ITS attributes.

Original:

```
<p>Text <span its-locale-filter-list='fr'
    its-locale-filter-type='exclude'>text</span></p>
```

Extraction:

```
<xliff srcLang="en" version="2.1">
  ...
  <unit id="1">
    <segment>
      <source>Text <pc id="1"><mrk id="m1" type="its:generic"
          its:localeFilterList="fr" its:localeFilterType="exclude">text
```

```
          </mrk></pc></source>
    </segment>
  </unit>
```

When the target locale in XLIFF is defined, use the `<mrk>` or an `<sm/>` / `<em/>` pair with `translate="yes"` if the target locale does apply, or `translate="no"` if it does not.

Original:

```
<p>Text <span its-locale-filter-list='fr' its-locale-filter-type='exclude'>
    text</span></p>
```

Extraction:

```
<xliff srcLang="en" trgLang="fr" version="2.1"...>
  ...
  <unit id="1">
    <segment>
      <source>Text <pc id="1"><mrk id="m1" type="generic" translate="no">
          text</mrk></pc></source>
    </segment>
  </unit>
```

ITS Locale Filter Annotation

This is used to fully map to and from the [ITS] Locale Filter [http://www.w3.org/TR/its20/#LocaleFilter] data category.

Usage:

- The `localeFilterList` attribute is *required* and used to map to and from the [ITS] defined `localeFilterList` attribute.
- The `type` attribute is *optional* and set to `its:generic`.
- The `its:localeFilterType` attribute is *optional* and used to map to and from the [ITS] defined `localeFilterList` attribute.
- The `translate` attribute is *optional*.

Localization Quality Issue

Expresses information related to localization quality assessment tasks in the form of highlighted issues. See [ITS] Localization Quality Issue [http://www.w3.org/TR/its20/#lqissue] for more details.

Processing Requirements

- *Writers must* use the ITS Localization Quality Issue Annotation to express the [ITS] Localization Quality Issue [http://www.w3.org/TR/its20/#lqissue] data category in *XLIFF Documents*.

Structural Elements

Localization Quality Issue is not to be used at structural levels. If a structural element of the original document has [ITS] Localization Quality Issue

[http://www.w3.org/TR/its20/#lqissue] information associated, it *must* be anyway *Extracted* using the ITS Localization Quality Issue Annotation.

Note

If human reviewers or other QA agents (*Enriching Agents* from the XLIFF specification point of view), need to insert general comments pertaining to whole structural elements such as paragraphs, sections, or files rather than to specific inline portions of source or target content, the Localization Note data category is more suitable.

Inline Elements

Use `<mrk>` or an `<sm/>` / `<em/>` pair with the attributes: `its:locQualityIssueComment`, `its:locQualityIssueEnabled`, `its:locQualityIssueProfileRef`, `its:locQualityIssuesRef`, `its:locQualityIssueSeverity`, and `its:locQualityIssueType`.

See the ITS Localization Quality Issue Annotation for the normative usage description of those attributes.

Because the same or overlapping spans of source or target text can be associated with more than one quality issue, this category provides its own elements that are to be used at the unit level as an alternative to the inline only annotations, especially in cases the inline only annotations would not be expressive enough to capture the issues to be reported. If more than one quality issue applies to the same content the particulars of those issues need to be stored in standoff annotations.

For specifics of the standoff annotation, see the `<locQualityIssue>` and `<locQualityIssues>` elements and the attributes `its:locQualityIssuesRef` and `id`.

ITS Localization Quality Issue Annotation

This is used to fully map to and from the [ITS] Localization Quality Issue [http://www.w3.org/TR/its20/#lqissue] data category.

Usage:

- The `id` attribute is *required*.
- The `type` attribute is *optional* and set to `its:generic`.
- Exactly one of the following *must* be set:
  - `its:locQualityIssuesRef`.
  - At least one of the following *must* be set:
    - `its:locQualityIssueType`,
    - `its:locQualityIssueComment`.
- The `translate` attribute is *optional*.
- The following attributes *must not* be set if and only if `its:locQualityIssuesRef` is declared, otherwise all of the following are *optional*:
  - `its:locQualityIssueSeverity`,
  - `its:locQualityIssueProfileRef`, and
  - `its:locQualityIssueEnabled`.

Warning

Usage of the `its:locQualityIssuesRef` attribute implies usage of Localization Quality Issue standoff elements. See `<locQualityIssues>` and `<locQualityIssue>` for related Constraints and Processing Requirements.

Example 5. Enriching XLIFF Documents with Localization Quality Issue Annotations

Simple (i.e. without stand off):

```
<unit id="1">
  <segment>
    <source>This is the content</source>
    <target><mrk id="m1" type="its:generic"
        its:locQualityIssueType="misspelling"
        its:locQualityIssueComment="'c'es' is unknown. Could be 'c'est'"
        its:locQualityIssueSeverity="50">c'es</mrk> le contenu</target>
  </segment>
</unit>
```

Stand off:

```
<unit id="1">
  <its:locQualityIssues xml:id="lqi1">
    <its:locQualityIssue locQualityIssueType="misspelling"
        locQualityIssueComment="'c'es' is unknown. Could be 'c'est'"
        locQualityIssueSeverity="50" />
    <its:locQualityIssue locQualityIssueType="grammar"
        locQualityIssueComment="Sentence is not capitalized"
        locQualityIssueSeverity="30" />
  </its:locQualityIssues>
  <segment>
    <source>This is the content</source>
    <target><mrk id="m1" type="its:generic"
        its:locQualityIssuesRef="lqi1">c'es le contenu</mrk></target>
  </segment>
</unit>
```

The `annotatorsRef` attribute inherits information in the document tree. The attribute `annotatorsRef` does not relate to standoff information. This is exemplified below. The `<mrk id="m1">` element has the `annotatorsRef` information - via `tool2` - expressed at the `target` element. The `tool1` `annotatorsRef` expressed at the `unit` element does not influence that interpretation and the standoff information in `<locQualityIssues>`.

```
<unit id="1" its:annotatorsRef="localization-quality-issue|tool1">
  <its:locQualityIssues xml:id="lqi1">
    <its:locQualityIssue locQualityIssueType="misspelling"
        locQualityIssueComment="'c'es' is unknown. Could be 'c'est'"
        locQualityIssueSeverity="50" />
    <its:locQualityIssue locQualityIssueType="grammar"
        locQualityIssueComment="Sentence is not capitalized"
```

```
        locQualityIssueSeverity="30" />
  </its:locQualityIssues>
  <segment >
    <source>This is the content</source>
    <target its:annotatorsRef="localization-quality-issue|tool2"><mrk id="m1" ty
  </segment>
</unit>
```

Localization Quality Rating

Expresses results of localization quality assessment in the form of aggreagated ratings, either as scores or as voting results. See [ITS] Localization Quality Rating [http://www.w3.org/TR/its20/#lqrating] for more details.

Processing Requirements

- *Writers must* use the ITS Localization Quality Rating Annotation to express the [ITS] Localization Quality Rating [http://www.w3.org/TR/its20/#lqrating] data category on inline spans within *XLIFF Documents*.

Structural Elements

Localization Quality Rating is usually expressed at structural levels as it normally expresses summary rating (scoring or voting) information for larger chunks of text. Rating information inherits to lower level elements but can be overridden at lower levels.

Attributes *may* be set on *XLIFF Core* structural elements, so that the following advanced Constraints are met.

Constraints

- Exactly one of the following *must* be set or inherited:
  - `its:locQualityRatingScore`:
    - `its:locQualityRatingScoreThreshold` *may* be set or inherited if and only if `its:locQualityRatingScore` is set.
  - `its:locQualityRatingVote`:
    - `its:locQualityRatingVoteThreshold` *may* be set or inherited if and only if `its:locQualityRatingVote` is set or inherited.
- The `its:locQualityRatingProfileRef` attribute is *optional*.

Inline Elements

Use `<mrk>` or an `<sm/>` / `<em/>` pair with the following attributes: `its:locQualityRatingProfileRef`, `its:locQualityRatingScore`, `its:locQualityRatingScoreThreshold`, `its:locQualityRatingVote`, `its:locQualityRatingVoteThreshold`.

See the ITS Localization Quality Rating Annotation for the normative usage description of those attributes inline.

ITS Localization Quality Rating Annotation

This is used to fully map to and from the [ITS] Localization Quality Rating [http://www.w3.org/TR/its20/#lqrating] data category on inline elements.

Usage:

- The `id` attribute is *required*.
- The `type` attribute is *optional* and set to `its:generic`.
- Exactly one of the following *must* be set:
  - `its:locQualityRatingScore`:
    - `its:locQualityRatingScoreThreshold` *may* be set or inherited if and only if `its:locQualityRatingScore` is set.
  - `its:locQualityRatingVote`:
    - `its:locQualityRatingVoteThreshold` *may* be set or inherited if and only if `its:locQualityRatingVote` is set or inherited.
- The `its:locQualityRatingProfileRef` attribute is *optional*.
- The `translate` attribute is *optional*.

Note

This annotation can be in scope of Localization Quality Rating attributes set at structural levels. So for instance a portion of target text with only a score set can inherit threshold and/or rating profile information set at a group or file level. Also summary 0-100 ratings set at higher levels can be for instance overridden with voting set at unit or inline elements. Keep in mind that for a specific portion of text only one can exist a rating or a vote result and these are to be accompanied with different threshhold attributes.

Example 6. Enriching XLIFF Documents with Localization Quality Rating Annotations

```
<unit id="1">
  <segment>
    <source>Some text and a term</source>
    <target>Du texte et un <mrk id="m1" type="its:generic"
        its:locQualityRatingVote="37"
        its:locQualityRatingVoteThreshold="15"
        its:locQualityRatingProfileRef="http://example.org/qaModel/v13">
        terme</mrk></target>
  </segment>
</unit>
```

Translation Candidates

In the Translation Candidates module, the Localization Quality Rating category attributes *may* be used to express the [ITS] Localization Quality Rating [http://www.w3.org/TR/its20/#lqrating] information.

Constraints

- When used on the `<match>` element, Constraints for Structural Elements apply,
- When used on eligible descendants of a `<match>` element, Constraints for Inline Elements apply.

Provenance

Communicate the identity of agents that have been involved in the translation of the content or the revision of the translated content. This allows translation and translation revision consumers, such as post-editors, translation quality reviewers, or localization workflow managers, to assess how the performance of these agents may impact the quality of the translation. Translation and translation revision agents can be identified as a person, a piece of software or an organization that has been involved in providing a translation or revision that resulted in the selected content. See [ITS] Provenance [http://www.w3.org/TR/its20/#provenance] for more details.

Warning

Provenance data category is used to record human, tools or organizational producers of *Translations* or revisions, in other words it records producers of the payload. To record [ITS] metadata producers, the ITS Tools Referencing mechanism needs to be used.

Processing Requirements

- *Writers must* use the attributes `its:org`, `its:orgRef`, `its:person`, `its:personRef`, `its:provenanceRecordsRef`, `its:revOrg`, `its:revOrgRef`, `its:revPerson`, `its:revPersonRef`, `its:revTool`, `its:revToolRef`, `its:tool`, and `its:toolRef` to express the [ITS] Provenance [http://www.w3.org/TR/its20/#provenance] data category in *XLIFF Documents*.
  - Within the Translation Candidates Module, *Enrichers must* map the `its:tool` attribute onto the `mtc:origin` attribute.
  - *Modifiers* populating *XLIFF Core* `<target>` elements with unmodified content from `<target>` children of `<mtc:match>` elements *may* map the `mtc:origin` onto the `its:tool` attribute.
    - The `its:tool` attribute value *must* be the same as the originating `<mtc:match>` `mtc:origin` value if this is the case.
  - *Modifiers may* store previous versions of subunit content and attributes and notes content and attributes in the Change Tracking Module elements according to the data model, Constraints, Processing Requirements, and usage descriptions of that module.

    If this was the case the `<revision>` element *must* be extended by the Provenance attributes defined in the ITS Module as needed and the `ctr:author` *should* reuse information from the corresponding [ITS] Provenance [http://www.w3.org/TR/its20/#provenance] attributes as follows:
    - space separated list of values
    - spaces " " and hyphens "-" in values are escaped using slashes "/"

- each value consists of the attribute name followed by a hyphen, followed by the ITS attribute value
- following attribute names to be used in that order if available:

  person
  tool
  revPerson
  revTool

- other attributes are ignored.

Structural Elements

Provenance metadata are more likely to appear on structural elements than on inline elements in source and target documents, therefore Provenance attributes listed in the above Processing Requirement are allowed on all structural levels.

It is possible that Provenance metadata will be *Extracted* from source content but more likely Provenance metadata will be first introduced into the translated content during the XLIFF based roundtrip.

Example 7. Provenance metadata added by *Modifiers* or *Enrichers* on structural levels

In this example a person of the name `Honza Novák` has been the translator of the whole unit content and `Franta Kocourek` the reviser of the whole translation.

```
...
<unit id="1" its:person="Honza Novák" its:revPerson="Franta Kocourek">
  <segment>
    <source>Economy has been growing in 2016.</source>
    <target>Hospodáství v pr b hu roku 2016 rostlo.</mrk></target>
  </segment>
  <segment>
    <source>Prognosis for 2017 is unclear.</source>
    <target>P edpov   o ekávaného r stu pro rok 2017 je nejasná.</target>
    </unit>
    ...
```

Preserving the Provenance metadata in the target content after *Merging* the *Translations* back to the original format can be useful, the metadata could be for instance used in a check in and publishing process within a content management system.

Example 8. Provenance metadata preserved by *Mergers* in the native format.

In this example the translator and reviser Provenance metadata introduced during the XLIFF roundtrip has been preserved after *Merging* of the *Translations* back to HTML.

```
...
<p its-person="Honza Novák" its-rev-person="Franta Kocourek"> Hospodáství
    v pr b hu roku 2016 rostlo. P edpov   o ekávaného r stu pro rok 2017
    je nejasná. </p>
...
```

If standoff Provenance elements are used at structural levels, these need to occur on the same or an ancestor element of the element where the standoff reference is used. See the `its:provenanceRecordsRef`

Inline Elements

Use `<mrk>` or an `<sm/>` / `<em/>` pair with the Provenance data category attributes listed in the above Processing Requirement.

See the ITS Provenance Annotation for the normative usage description of those attributes inline.

Because the same or overlapping spans of source or target text can be associated with more than one Provenance record, for instance over time, this category provides its own elements that are to be used at the unit level as a more expressive alternative to the inline only annotations.

For specifics of the standoff annotation, see the `<provenanceRecord>` and `<provenanceRecords>` elements and the attributes `provenanceRecordsRef` and `id`.

ITS Provenance Annotation

This is used to fully map to and from the [ITS] Provenance [http://www.w3.org/TR/its20/#provenance] data category when used inline.

Usage:

- The `id` attribute is *required*.
- The `type` attribute is *optional* and set to `its:generic`.
- The `translate` attribute is *optional*.
- The `its:provenanceRecordsRef` attribute is *optional*.
- The following attributes *must not* be set if and only if `its:provenanceRecordsRef` is declared, otherwise at least one the following *must* be set:
  - `its:org`,
  - `its:orgRef`,
  - `its:person`,
  - `its:personRef`,
  - `its:revOrg`,
  - `its:revOrgRef`,
  - `its:revPerson`,
  - `its:revPersonRef`,
  - `its:revTool`,
  - `its:revToolRef`,
  - `its:tool`,
  - `its:toolRef`,

Warning

Usage of the `its:provenanceRecordsRef` attribute implies usage of Provenance standoff elements. See `<provenanceRecords>` and `<provenanceRecord>` for related Constraints and Processing Requirements.

Example 9. *Enriching XLIFF Documents* with Provenance Annotations

Inline only (i.e. without stand off):

```
...
<unit id='1'>
  <segment>
    <source>Economy has been growing in 2016.</source>
    <target><mrk id="m1" type="its:generic" its:tool="Microsoft Hub"
        its:person="Honza Novák" its:revPerson="Franta Kocourek">
        Hospodá ství v pr b hu roku 2016 rostlo. </mrk></target>
  </segment>
  <segment>
    <source>Prognosis for 2017 is unclear.</source>
    <target><mrk id="m2" type="its:generic" its:tool="Microsoft Hub"
        its:person="Honza Novák"> P edpov   o ekávaného r stu pro rok
        2017 je nejasná. </mrk></target>
  </segment>
</unit>
...
```

In this example, both segments were translated by `Microsoft Hub` and by `Honza Novák` from `P eklady Novák, sro`. The first segment was also revised by `Franta Kocourek` from `Kocourkov s.r.o.`, while the second segment hasn't been revised. Because order of attributes cannot have semantics in XML, we can only speculate about the order in which the people and tools had contributed to the workflow and also each of the attributes can have only one value applied for the given span.

Stand off:

```
...
<unit id='1'>
  <its:provenanceRecords xml:id="prov1">
    <provenanceRecord revPerson="Franta Kocourek"
        revOrg="Kocourkov s.r.o."/>
    <provenanceRecord person="Honza Novák" org="P eklady Novák, sro"
        tool="GreatCATTool"/>
    <provenanceRecord tool="Microsoft Hub"/>
  </its:provenanceRecords>
  <its:provenanceRecords xml:id="prov2">
    <provenanceRecord revPerson="Kv to  Z ídkaveselý" revOrg="CoolCopy"/>
    <provenanceRecord revTool="ACME QA Checker" revOrg="CoolCopy"/>
    <provenanceRecord revPerson="Franta Kocourek"
        revOrg="Kocourkov s.r.o."/>
    <provenanceRecord person="Honza Novák" org="P eklady Novák, sro"
        tool="GreatCATTool"/>
    <provenanceRecord tool="Microsoft Hub"/>
  </its:provenanceRecords>
  <segment>
```

```
    <source>Economy has been growing in 2016.</source>
    <target><mrk id="m1" type="its:generic"
        its:provenanceRecordsRef="#its=prov1"> Hospodá ství v pr b hu
        roku 2016 rostlo. </mrk></target>
  </segment>
  <segment>
    <source>Prognosis for 2017 is unclear.</source>
    <target><mrk id="m2" type="its:generic"
        its:provenanceRecordsRef="#its=prov2"> Hospodá ství v pr b hu
        roku 2016 rostlo. </mrk></target>
  </segment>
</unit>
...
```

In this example, multiple records with the same attribute for the same span are possible, and if most recent records are stacked on top, it can also help indicate the sequence of agents. So both segments were most probably first translated by `Microsoft Hub`, then by `Honza Novák` from `P eklady Novák, sro` using `GreatCATTool`. Both segments were subsequently revised by `Franta Kocourek` from `Kocourkov s.r.o.` (using an unknown revision tool), and the second segment has been also revised at `CoolCopy` by a tool `ACME QA Checker` and once more by a human `Kv to  Z ídkaveselý` from `CoolCopy`. Indicating both the first and second revisers, as well as hinting on the sequence of different translation tools would have been impossible if the annotation was inline only.

Text Analysis

Annotates content with lexical or conceptual information for the purpose of contextual disambiguation of words and multiword phrases meanings. See [ITS] Text Analysis [http://www.w3.org/TR/its20/#textanalysis] for details.

Processing Requirements

- *Writers must* use the ITS Text Analysis Annotation to express the [ITS] Text Analysis [http://www.w3.org/TR/its20/#textanalysis] data category in *XLIFF Documents*.

Structural Elements

Text Analysis is not to be used at structural levels. If a structural element of the original document has [ITS] Text Analysis [http://www.w3.org/TR/its20/#textanalysis] information associated, it *may* be *Extracted* using the ITS Text Analysis Annotation.

Example 10. Extraction of Text Analysis at structural levels

Original:

```
<p its-ta-class-ref="http://nerd.eurecom.fr/ontology#Place"
    its-ta-ident-ref="http://dbpedia.org/resource/Arizona">Arizona</p>
```

Extraction:

```
...
<unit id="1">
  <segment>
    <source><mrk id="m1" type="its:generic"
```

```
            its:taClassRef="http://nerd.eurecom.fr/ontology#Place"
            its:taIdentRef="http://dbpedia.org/resource/Arizona">Arizona</mrk>
        </source>
      </segment>
    </unit>
    ...
```

Inline Elements

Use `<mrk>` or an `<sm/>` / `<em/>` pair with the following attributes: `its:taClassRef`, `its:taConfidence`, `its:taSource`, `its:taIdent`, and `its:taIdentRef`.

See the ITS Text Analysis Annotation for the normative usage description of those attributes.

ITS Text Analysis Annotation

This is used to fully map to and from the [ITS] Text Analysis [http://www.w3.org/TR/its20/#textanalysis] data category.

Usage:

- The `id` attribute is *required*.
- The `type` attribute is *optional* and set to `its:generic`.
- At least one of the following *must* be set:
  - `its:taClassRef`,
  - Exactly one of the following:
    - A pair of a `its:taSource` and `its:taIdent` both set,
    - `its:taIdentRef`.
- The `translate` attribute is *optional*.
- The `its:taConfidence` attribute is *optional* and used to map to and from the [ITS] defined `taConfidence` attribute.
- The `its:annotatorsRef` attribute is *required* if and only if the `its:taConfidence` attribute is present and not in scope of another relevant `its:annotatorsRef` attribute, in all other cases it is *optional*.

Example 11. Extraction of ITS Text Analytics metadata in scope of the ITS tools annotation

Original:

```
<div its-annotators-ref="text-analysis|http://enrycher.ijs.si [http://enrycher.
    ...
    <p><span its-ta-class-ref="http://nerd.eurecom.fr/ontology#Place"
        its-ta-confidence="0.99"
        its-ta-ident-ref="http://dbpedia.org/resource/Arizona">Arizona
        </span></p> ... </div>
```

Extracted:

```
<unit id="1" its:annotatorsRef="text-analysis|http://enrycher.ijs.si [http://en
```

```
<segment>
  <source><mrk id="m1" type="its:generic"
      its:taClassRef="http://nerd.eurecom.fr/ontology#Place"
      its:taIdentRef="http://dbpedia.org/resource/Arizona"
      its:taConfidence="0.99" > Arizona</mrk></source>
</segment></unit>
```

ITS data categories that have a partial overlap with XLIFF features

The following [ITS] data cetegories are partially covered with *XLIFF Core* or *Modules* other than the ITS Module:

1. Localization Note,
2. Terminology,
3. Language Information,
4. MT Confidence, and
5. Storage Size.

Localization Note

Provides a way to communicate notes to localizers about a particular item of content. See [ITS] Localization Note [http://www.w3.org/TR/its20/#locNote-datacat] for details.

Warning

There is a one-to-one mapping for all parts of the Localization Note [http://www.w3.org/TR/its20/#locNote-datacat] information to and from the *XLIFF Core* `<note>` and the Comment Annotation mechanism. This means that the whole data category can be losslessly *Extracted* from the native format, *Merged* back to the native format or even round-tripped. However, generic ITS Processors won't be able to fully access the Localization Note [http://www.w3.org/TR/its20/#locNote-datacat] information encoded in *XLIFF Documents*. The Localization Note [http://www.w3.org/TR/its20/#locNote-datacat] rules contained in the ITS Module Schematron Schema (http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/its.sch) won't be able to parse *XLIFF Core* `<note>` elements placed on `<unit>` unless they have set the attribute `appliesTo`.

Structural Elements

Localization Notes present in source content at structural levels are *Extracted* using the *XLIFF Core* `<note>` and the `<note>` element. ITS attribute `locNoteType` is mapped onto the *XLIFF Core* attribute `priority`. The value `alert` is mapped onto priority `1`. The value `description` is mapped onto any of the integers `2-10`.

Example 12. Extraction of a Localization note at a structural level

Original:

```
<msgList xmlns:its= "http://www.w3.org/2005/11/its" xml:space= "preserve"
    its:version= "2.0">
```

```
  <data name= "LISTFILTERS_VARIANT" its:locNote= "Keep the leading space!"
      its:locNoteType= "alert">
    <value> Variant {0} = {1} ({2}) </value>
  </data>
  <data its:locNote= "%1\$s is the original text's date in the format
      YYYY-MM-DD HH:MM always in GMT">
    <value>Translated from English content dated
       <span id= "version-info">%1\$s</span> GMT.</value>
  </data>
</msgList>
```

Extraction:

```
<file id="1" xml:space="preserve">
  <unit id="1" name="LISTFILTERS_VARIANT">
    <notes>
      <note priority="1">Keep the leading space! </note>
    </notes>
    <segment>
      <source> Variant {0} = {1} ({2}) </source>
    </segment>
  </unit>
  <unit id="2" name="LISTFILTERS_VARIANT">
    <notes>
      <note priority="2">%1\$s is the original text's date in the format
          YYYY-MM-DD HH:MM always in GMT </note>
    </notes>
    <segment>
      <source>Translated from English content dated
          <pc id="1">%1\$s</pc> GMT.</source>
    </segment>
  </unit>
</file>
```

Warning

The values of the ITS attribute `locNoteRef` are to be dereferenced during *Extraction*, so that the Localization Note text can be included verbatim in the XLIFF `<note>` element. A corresponding attribute is NOT provided through the ITS Module to discourage external references from XLIFF Notes. The `locNoteRef` attribute and its value still can be preserved on *Extraction* via extensibility, however this information will not have a guaranteed roundtrip protection and the XLIFF Note itself still better include the dereferenced Localization Note text.

Inline Elements

Localization Notes present on inline spans of source content are *Extracted* using the *XLIFF Core* Annotations mechanism. Use `<mrk>` or an `<sm/>` / `<em/>` pair with `type="comment"`. See Comment Annotation.

Comment Annotations can either contain the Localization Note text as the value of the attribute `value` or otherwise have to reference a `<note>` element within the same enclosing `<unit>`. In case no `<note>` element is referenced, it is assumed that the ITS `locNoteType` is `description`. In case the referenced `<note>` element has `priority`

`1` or does not have the `priority` attribute set explicitly, the ITS `locNoteType` is `alert`. Explicitly set values `2-10` map onto the ITS `locNoteType` value `description`.

Example 13. Extraction of an inline Localization Note

Original:

```
<!DOCTYPE html>
<html lang=en>
  <head>
    <meta charset=utf-8>
    <title>LocNote test: Default</title>
  </head>
  <body>
    <p>This is a
      <span its-loc-note="Check with terminology engineer"
          its-loc-note-type="alert"> motherboard</span>.
    </p>
  </body>
</html>
```

Extraction:

```
<xliff version="2.1" srcLang="EN">
  <file id=1>
    <unit id='1'>
      <notes>
        <note id="1" priority="1">Check with terminology engineer</note>
      </notes>
      <segment>
        <source>This is a <mrk id="1" type="comment" ref="#n=1">
            motherboard</mrk>.</source>
      </segment>
    </unit>
  </file>
</xliff>
```

Terminology

Marks terms and optionally associates them with information, such as definitions. See [ITS] Terminology [http://www.w3.org/TR/its20/#terminology] for details.

ITS Terminology information is useful during *Translation* and related localization processes. Thus it is beneficial when *Extractors* preserve the ITS Terminology information in *XLIFF Documents*.

Target language terminology data and metada introduced during the *Translation* can be *Merged* back into the target language content in the original format.

Warning

The *XLIFF Core* Term Annotation does not support all aspects of the [ITS] Terminology [http://www.w3.org/TR/its20/#terminology] data category. For instance, the *XLIFF Core* Term Annotation cannot be used to mark a span as not a term, which is needed to map ITS `term="no"`. In case lossless roundtrip of this category needs

to be achieved, the Core Annotation needs to be extended as defined by the ITS Terminology Annotation.

Structural Elements

Even if ITS Terminology metadata appears on structural elements in the source format, this information needs to be *Extracted* using the *XLIFF Core* Annotations mechanism. Use `<mrk>` or an `<sm/>` / `<em/>` pair with `type="term"`. See Term Annotation.

Example 14. Extraction of Terminology from structural elements

Original:

```
<p its-term='yes'>Term</p>
```

Extraction:

```
<unit id='1'>
  <segment>
    <source><mrk id="m1" type="term">Term</mrk></source>
  </segment>
</unit>
```

Inline Elements

Inline Terminology information *may* be *Extracted* using the *XLIFF Core* Annotations mechanism. Use `<mrk>` or an `<sm/>` / `<em/>` pair with `type="term"`. See Term Annotation.

Example 15. Extraction of inline Terminology using Annotation markers

Original:

```
<p>Text with a <span its-term='yes'>term</span>.</p>
```

Extraction:

```
<unit id='1'>
  <segment>
    <source>Text with a <pc id='1'><mrk id="m1" type="term">term</mrk>
        </pc>.</source>
  </segment>
</unit>
```

ITS Terminology Annotation

This is used to fully map to and from the [ITS] Terminology [http://www.w3.org/TR/its20/#terminology] data category, including the aspects that are not supported via the *XLIFF Core* Term Annotation.

Usage:

- The `id` attribute is *required*.
- The `type` attribute is *required* and set:
  - either to `its:term-no`, which maps to and from the [ITS] defined `term` attribute set to `no`,
  - or to `term`, which maps to and from the [ITS] defined `term` attribute set to `yes`.
- Not more than one of the following two attributes *may* be set:
  - The `value` attribute is *optional* and contains a short definition of the term that an *Extractor* obtained by dereferencing the [ITS] defined `termInfoPointer` or added by an *Enricher*.
  - The `ref` attribute is *optional* and used to map to and from the [ITS] defined `termInfoRef` attribute.
- The `translate` attribute is *optional*.
- The `its:termConfidence` attribute is *optional* and used to map to and from the [ITS] defined `termConfidence` attribute.
- The `its:annotatorsRef` attribute is *required* if and only if the `its:termConfidence` attribute is present and NOT in scope of another relevant `its:annotatorsRef` attribute, in all other cases it is *optional*.

Example 16. Extraction of ITS Terminology with termConfidence

```
<div its-annotators-ref="terminology|http://example.org/TermService">
  ...
  <p>Text with a <span its-term='yes'
      its-term-info-ref='http://en.wikipedia.org/wiki/Terminology'
      its-term-confidence='0.9'>term</span>.</p>
  ...
</div>
```

Extracted:

```
<unit id='1'
    its:annotatorsRef='terminology|http://example.com/termchecker'>
  <segment>
    <source>Text with a <pc id="1"><mrk id="m1" type="term"
        ref="http://en.wikipedia.org/wiki/Terminology"
        its:termConfidence="0.9">term</mrk></pc>.</source>
  </segment>
</unit>
```

Language Information

Indicates the natural langauge in which content is expressed. See [ITS] Language Information [http://www.w3.org/TR/its20/#language-information] for details.

Structural Elements

XLIFF Documents are normally bilingual, hence the source and target language are indicated at the top level using the `srcLang` and `trgLang` attributes set on the `xliff` element. The Language Information values set on the top level, strictly constrain the

values of `xml:lang` set or inherited on the `<source>` element for source content and on the `<target>` element for target content.

Note

Because *XLIFF Documents* are normally source-monolingual, whole paragraphs in the source document that are not in the main source language are generally not to be extracted. If there is a need to extract such content into a single *XLIFF Documents*, the XLIFF output has to use the inline Annotations mechanism together with the ITS Language Information Annotation, because the structurally set or inherited source language is constrained by the *XLIFF Core* `srcLang` attribute value. Analogically, the structurally set target language is constrained by the `trgLang` attribute value. Thus also paragraphs other than in the main target language have to be annotated inline using the same mechanism.

Inline Elements

It is not possible to use [XML namespace] on XLIFF inline elements. It is advised that content in different languages is NOT used inline in source formats. Still there are use cases for mixed language use inline, like referencing non-localized UI or hardware elements, discussing foreign vocabulary or analyzing poetry in the original language using short inline examples. These scenarios cannot be fully supported with *XLIFF Core* only.

In case the inline elements in other than the main language are not supposed to be translated (e.g. referenced non localized UI or hardware elements), they can be marked as not translatable using the *XLIFF Core* Translate annotation. However, the specific Language Information would not be readily accessible during the roundtrip if not combined with the Language Information Annotation defined here in the ITS Module.

Note

If there is a need to make the different language information available throughout the roundtrip, the XLIFF output has to use the inline Annotations mechanism together with the ITS Language Information Annotation, because the structurally set and thus inherited inline source language is constrained by the *XLIFF Core* `srcLang` attribute value. Analogically, the structurally set (and inline inherited) target language is constrained by the `trgLang` attribute value. Thus also inline portions in other than the main target language have to be inline annotated using the same mechanism.

Warning

Preserving source elements content that is in other than the main source language as original data stored outside of the translatable content at the unit level and referenced from placeholder codes is NOT advised, as important context would be very likely hidden from translators, human or machine.

Example 17. Core only extraction and roundtrip of a non localized hardware reference in other than the main source language

Original:

```
<p> Use the <span class="HWbutton" xml:lang="DE-DE">Aus</span> button to
    completely switch off the machine. </p>
```

Extraction:

```
<unit id='1'>
  <originalData>
    <data id="d1">&lt;span class="HWbutton" xml:lang="DE-DE"></data>
    <data id="d2">&lt;/span></data>
  </originalData>
  <segment>
    <source> Use the <pc id="1" dataRefStart="d1" dataRefEnd="d2">
        <mrk id=2 translate="no">Aus</mrk></pc> button to completely
        switch off the machine. </source>
  </segment>
</unit>
```

Please note that the Language Information has been preserved for *Merging* back in the referenced original data, is however not available in an interoperable way during the roundtrip.

ITS Language Information Annotation

This is used to fully map to and from the [ITS] Language Information [http://www.w3.org/TR/its20/#language-information] data category, including full inline support that cannot be provided via the *XLIFF Core* due to normative Constraints.

Usage:

- The `id` attribute is *required*.
- The `itsm:lang` attribute is *required*.
- The `type` attribute is *optional* and set to `its:generic`.
- The `translate` attribute is *optional*.

Example 18. Extraction of Language Information

Original:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>My Document</title>
  </head>
  <body>
    <p>Span of text <span lang="fr">en français</span>.</p>
  </body>
</html>
```

Extraction:

```
...
<unit id='2'>
  <segment>
    <source>Span of text <pc id='1'><mrk id="m1" itsm:lang="fr"
        type="its:generic" >en français</mrk></pc>.</source>
  </segment>
```

```
</unit>
...
```

Warning

Please note that the Language Information Annotation uses the `itsm:lang` attribute that belongs to the `urn:oasis:names:tc:xliff:itsm:2.1` namespace (prefixed with `itsm:`) and not to the `http://www.w3.org/2005/11/its` (prefixed with `its`) as most of the other attributes described in this module.

MT Confidence

communicates the confidence score from a machine translation engine for the accuracy of a translation it has provided [ITS] MT Confidence [http://www.w3.org/TR/its20/#mtconfidence] for details.

Warning

MT Confidence is not intended to provide a score that is comparable among or between Machine Translation engines and platforms. This data category does NOT aim to establish any sort of correlation between the confidence score and either human evaluation of MT usefulness, or post-editing cognitive effort.

Within the Translation Candidates module

The most natural step to introduce the MT Confidence metadata into the multilingual content life cycle is during the XLIFF roundtrip, when the *XLIFF Document* is being *Enriched* with Translation Candidates from a specific MT Service or via an MT Services broker. The MT Confidence metadata included with the MT provided matches *may* be used by human or machine *Modifiers* who populate the *XLIFF Core* `<target>` elements with matches.

In the Translation Candidates Module, there is a partial overlap between the [ITS] MT Confidence [http://www.w3.org/TR/its20/#mtconfidence] and *XLIFF-defined* features. See the `mtConfidence` attribute for the mapping details, Advanced Constraints and Processing Requirements.

Example 19. MT Confidence as Translation Candidates metadata

```
<xliff version="2.0" xmlns="urn:oasis:names:tc:xliff:document:2.0"
    xmlns:mtc="urn:oasis:names:tc:xliff:matches:2.0"
    xmlns:its="http://www.w3.org/2005/11/its" its:version="2.0"
    srcLang="en" trgLang="fr">
  <file id="f1" its:annotatorsRef="mt-confidence|MTServices-XYZ">
    <unit id="1">
      <mtc:matches>
        <!-- Score provided by MTServices-XYZ -->
        <mtc:match ref="#m1" matchQuality="89.82">
          <source>Text</source>
          <target >Texte</target>
        </mtc:match>
        <!-- Score provided by MTProvider-ABC -->
        <mtc:match ref="#m1" matchQuality="67.8"
            its:annotatorsRef="mt-confidence|MTProvider-ABC">
          <source>Text</source>
          <target >Texte</target>
```

```
          </mtc:match>
          <!-- Score provided by MTProvider-JKL -->
          <mtc:match ref="#m1" matchQuality="65"
              its:annotatorsRef="mt-confidence|MTProvider-JKL">
            <source>Text</source>
            <target >texte</target>
          </mtc:match>
          <!-- Score provided by MTServices-XYZ -->
          <mtc:match ref="#m1" matchQuality="89.82">
            <source>Some text</source>
            <target>Du texte</target>
          </mtc:match>
        </mtc:matches>
        <segment>
          <source><mrk id='m1' type='mtc:match'>Text</mrk></source>
        </segment>
      </unit>
    </file>
</xliff>
```

Warning

Generic ITS Processors cannot directly read MT Confidence data from the XLIFF Translation Candidates Module because ITS 2.0 does not define a global pointer for this data category.

Structural Elements

It is NOT advised that [ITS] MT Confidence [http://www.w3.org/TR/its20/#mtconfidence] be used at a structural level because meaningful MT Confidence scores will vary from segment to segment. If a structural element of an original document has an [ITS] MT Confidence [http://www.w3.org/TR/its20/#mtconfidence] annotation, it *may* be represented upon *Extraction* using the MT Confidence Inline Annotation. The whole unit source content *must* be enclosed within the annotation in such a case, possibly spanning multiple segments.

Inline Elements

Example 20. Extraction of ITS MT Confidence Metadata from a Raw MTed source document

Original:

```
<p><span its:mtConfidence="0.8982"
    its:annotatorsRef="mt-confidence|MTServices-XYZ">Some Machine
        Translated text. </span></p>
```

Extraction from a raw MT original:

```
<unit id="u1">
  <segment>
    <source><mrk id="m1" type="its:generic" its:mtConfidence="0.8982"
        its:annotatorsRef="mt-confidence|MTServices-XYZ" >Some Machine
        Translated text.</mrk></source>
```

```
  </segment>
</unit>
```

MT Confidence Annotation

This is used to fully map to and from the [ITS] MT Confidence [https://www.w3.org/TR/its20/#mtconfidence] data category in *XLIFF Core*.

Usage:

- The `id` attribute is *required*.
- The `type` attribute is *optional* and set to `its:generic`.
- The the [ITS] defined attribute `its:mtConfidence` *must* be set.
- The `translate` attribute is *optional*.
- The `its:annotatorsRef` attribute is *required* if and only if the `its:mtConfidence` attribute is not in scope of another relevant `its:annotatorsRef` attribute.

Example 21. Populating *XLIFF Core* targets with raw MT along with ITS MT Confidence metadata

Original:

```
<p> Some human authored text for translation. </p>
```

Extracted text Enriched with a Machine Translated candidate and the same candidate inserted into the core target:

```
<unit id="u1">
  <mtc:matches>
    <mtc:match ref="#t=m1" matchQuality="67.8"
        its:annotatorsRef="mt-confidence|GoogleTranslate">
      <source xml:lang="EN">Some human authored text for translation.
          </source>
      <target xml:lang="CS">N které lidské napsaný text ur ený k p ekladu .
          </target>
    </mtc:match>
  </mtc:matches>
  <segment>
    <source xml:lang="EN">Some human authored text for translation.
        </source>
    <target xml:lang="CS"><mrk id="m1" type="its:generic"
        its:mtConfidence="0.678"
        its:annotatorsRef="mt-confidence|GoogleTranslate">N které lidské
        napsaný text ur ený k p ekladu .</mrk></target>
  </segment>
</unit>
```

Raw MT Merged back into the original format with MT Confidence metadata:

```
<p><span its:mtConfidence="0.678"
    its:annotatorsRef="mt-confidence|GoogleTranslate"> N které lidské
```

```
napsaný text ur ený k p ekladu . </span></p>
```

Processing Requirements

- *Modifiers* populating *XLIFF Core* `<target>` elements with unmodified MT suggestions *may* annotate the exact unmodified target spans with MT Confidence Annotations.

> **Warning**
>
> The MT Confidence Annotations need to be removed whenever the original MT is modified, no matter if by human post-editors or some automated post-editing methods. This is however not enforceable since the subsequent *Modifiers* might not be aware of the ITS Module data. Thus it is not advised to transfer the MT Confidence data onto *XLIFF Core* targets if any sort of post editing is foreseen or possible in the subsesquent steps of the XLIFF Round-trip, unless the post-editors were instructed and equipped to remove the MT Confidence Annotations as soon as they touch the MT suggestions. Preserving the MT Confidence data in *XLIFF Core* `<target>` elements only makes sense if the data needs to be preserved throughout *Merging* back to the original format, for instance for data analytic purposes or to color code the raw MTed target text for the end user based on the MT Confidence scores.

Storage Size

Mapping for this metadata category has not been specified in XLIFF Version 2.1

Processing Requirements

- The [ITS] Storage Size data category [https://www.w3.org/TR/its20/#storagesize] *may* be expressed as an Extended profile within the Size and Length Restriction Module. No other parts of XLIFF *must* be extended to support this data category.

> **Note**
>
> An *XLIFF-defined* common profile could be made part of this module in a future Version of XLIFF.

ITS data categories available through XLIFF Core and other Modules

The following [ITS] data categories are fully available via *XLIFF Core* and other XLIFF modules:

1. Translate and
2. External Resource.
3. Preserve Space

Translate

Indicates whether content is translatable or not. See [ITS] Translate [http://www.w3.org/TR/its20/#trans-datacat] for details.

ITS data category Translate in source content influences how *Extractors* prepare source content for *Translation* via *XLIFF Documents*.

Structural Elements

Use the `translate` attribute:

Example 22. Extraction of Translate at structural levels

Original:

```
<p translate='yes'>Translatable text</p>
<p translate='no'>Non-translatable text</p>
```

Extraction:

```
<unit id='1' translate="yes">
  <segment>
    <source>Translatable text</source>
  </segment>
</unit>
<unit id='2' translate="no">
  <segment>
    <source>Non-translatable text</source>
  </segment>
</unit>
```

If an element is not translatable you can also simply not extract it.

Inline Elements

Use `<mrk>` or an `<sm/>` / `<em/>` pair with `translate='yes|no'`. Another option is to extract the non-translatable content as an inline code. However, it is worth noting that *Extracting* non-translatable text as inline code data can hide important context information from translators, human or machine. The *Extraction* as code data is preferable if the non-translatable text has purely programmatic purpose and bears no linguistic relationship to the surrounding translatable text.

Example 23. Extraction of non-translatable inline text using Annotation markers

Original:

```
<p>The  <span translate="no">World Wide Web Consortium</span> makes the
    World Wide Web world wide.</p>
```

Extraction:

In this case the non-translatable span is a critical part of the content (a brand name) and hiding it within a code could potentially cause lot of damage, albeit non-translatable.

```
<unit id='1'>
  <segment>
    <source>The <pc id='1'/><mrk id='m1' translate='no'>World Wide Web
        Consortium</mrk></pc> makes the World Wide Web world wide.
```

```
        </source>
  </segment>
</unit>
```

Example 24. Protection of non-translatable inline text using an inline code

```
<p>You have <code translate='no'>%1</span> messages.</p>
```

```
<unit id='1'>
  <originalData>
    <data id="1">%1</data>
  </originalData>
  <segment>
    <source>You have <ph id='1' dataRef="1" type="ui" subtype="xlf:var"
        disp="[a variable number]" equiv="%1"/></source>
  </segment>
</unit>
```

Protection of non-translatable code as a code is more fool proof. On the other hand, it can hide the nature of the placeholder and it's linguistic relationship to the rest of the content from the translators. Therefore, it's advised to use maximum redundancy on the <ph> to make sure that CAT tools can pickup up something useful to display in their editing GUI to the Translator. It's completely another challenge to make an MT engine understand that the placeholder has a significant linguistic relationship to the rest of the sentence.

External Resource

Indicates that a node represents or references potentially translatable data in a resource outside the document. Examples of such resources are external images and audio or video files. See [ITS] External Resource [https://www.w3.org/TR/its20/#externalresource] for details.

Structural Elements

External Resource is not to be used at structural levels. If a structural element of the original document has [ITS] External Resource [http://www.w3.org/TR/its20/#externalresource] information associated, it *may* be *Extracted* using the XLIFF Resource Data Module. The *Extractor* needs to determine the media type of the external resource, since this is not available via [ITS] External Resource [http://www.w3.org/TR/its20/#externalresource] information.

Example 25. Extraction of External Resource at structural levels

Original:

```
<its:rules version="2.0" xmlns:its="http://www.w3.org/2005/11/its"
    xmlns:html="http://www.w3.org/1999/xhtml">
  <its:externalResourceRefRule selector="//html:video/@src"
      externalResourceRefPointer="."/>
  <its:externalResourceRefRule selector="//html:video/@poster"
```

```
      externalResourceRefPointer="."/>
</its:rules>
...
<video height=360 poster=video-image.png
    src=http://www.example.com/video/v2.mp width=640>
```

Extraction:

```
...
<res:resourceData>
  <res:resourceItem id="r1" mimeType="image/png" context="no">
    <res:source href="video-image.png" />
  </res:resourceItem>
</res:resourceData>
...
```

Inline Elements

External resources is *Extracted* using the XLIFF Resource Data module. Use a `<res:source>` element as a child of a `<res:resourceItem>`element.

Example 26. Extraction of External Resource at inline levels

Original:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Data Category: External Resource</title>
    <script type="application/its+xml">
      <its:rules xmlns:its="http://www.w3.org/2005/11/its" version="2.0"
          xmlns:h="http://www.w3.org/1999/xhtml">
        <its:externalResourceRefRule selector="//h:img"
            externalResourceRefPointer="@src"/>
      </its:rules>
    </script>
  </head>
  <body>
    <p>Image: <img src="example.png" alt="Text for the image"></p>
  </body>
</html>
```

Extraction:

```
<unit id="1">
  <res:resourceData>
    <res:resourceItem id="r1" mimeType="image/png" context="no">
      <res:source href="example.png" />
    </res:resourceItem>
  </res:resourceData>
  <segment>
    <source>Image: <ph id="ph1" fs="img"
        subFs="src,example.png\alt,Text for the image" /></source>
```

```
  </segment>
</unit>
```

Preserve Space

Indicates how to handle whitespace in a given content portion. See [ITS] Preserve Space [http://www.w3.org/TR/its20/#preservespace] for details.

Structural Elements

Whitespace handling at the structural level is indicated with `xml:space` in *XLIFF Core* and extensions:

Example 27. Extraction of preserved whitespace at the structural level

Original:

```
<listing xml:space='preserve'>Line 1 Line 2</listing>
```

Extraction:

```
<unit id='1' xml:space='preserve'>
  <segment>
    <source>Line 1 Line 2</source>
  </segment>
</unit>
```

Inline Elements

It is not possible to use [XML namespace] on XLIFF inline elements. It is advised that mixed Preserve Space behavior is NOT used inline in source formats. The advised way to extract content with mixed Preserve Space behavior is for the *Extractor* to perform the following:

1. Normalize the whitespace in the content as needed, i.e. preserving whitespace spans where they need to be preserved, normalizing elsewhere.
2. Then, extract the content with `xml:space` set to `preserve` on the structural level, i.e. `unit` or higher.

Note

Even in case *Extractors* don't perform the normalization step, it is safer to set `xml:space` to `preserve` on the structural level, since any potentially superfluous whitespace characters can be removed by human translators or editors, whereas inheriting of the default value `default` could lead to irreversible loss of significant whitespace characters.

Whitespace handling can be also set independently for text segments and ignorable text portions within an *Extracted* unit and for the source and target language within the same `<segment>` or `<ignorable>` element using the *optional* `xml:space` attribute at the `<source>` and `<target>` elements.

It is important to note that the value of the `xml:space` attribute is restricted to `preserve` on the `<data>` element.

ITS data categories not represented in XLIFF

The following [ITS] data categories can be represented via *Extraction* and *Merging* behavior of XLIFF conformant *Agents* without including any ITS specific metadata in the *XLIFF Documents*:

1. Directionality,
2. Elements Within Text,
3. Locale Filter,
4. Target Pointer, and
5. ID Value.

Directionality

The Directionality [http://www.w3.org/TR/its20#directionality] data category allows the user to specify the base writing direction of blocks, embeddings, and overrides for the Unicode bidirectional algorithm [UAX #9]. In XLIFF the usage of this data category along the ITS lines is discouraged, since XLIFF provides its own mechanism to specify directionality, see Bidirectional Text.

Elements Within Text

The Elements Within Text [http://www.w3.org/TR#elements-within-text] data category reveals if and how an element affects the way text content behaves from a linguistic viewpoint. This information is for example relevant to provide basic text segmentation hints for tools such as translation memory systems. See [ITS] Elements Within Text [http://www.w3.org/TR#elements-within-text] for details.

The Elements Within Text [http://www.w3.org/TR/its20#elements-within-text] data category is used by ITS processors to generate XLIFF documents. This process is done by ITS processors, not by XLIFF *Writers* or other types of XLIFF implementations, to understand how to extract source content. The data category is not represented directly in *XLIFF Documents*.

The data category provides three values: `yes`, `no` and `nested`. See the ITS 2.0 specification [http://www.w3.org/TR/its20#within-text-definition] for examples of how to use these values in general XML vocabularies or in HTML. The below examples show how to deal with the values in XLIFF.

Elements Within Text Value `Yes`

The element needs to be mapped to one of the XLIFF 2.1 inline elements: <pc>, <sc>/<ec> or <ph>, while its content is extracted.

Example for using `pc` - Original:

```
...
<p>This paragraph contains <span its-within-text="yes">a spanned part
    </span>.</p>
...
```

Extraction:

```
...
<unit id="u1">
  <originalData>
    <data id="d1">&lt;span its-within-text="yes"&gt;</data>
    <data id="d2">&lt;/span&gt;</data>
  </originalData>
  <segment>
    <source>This paragraph contains <pc id="pc1" dataRefStart="d1"
        dataRefEnd="d2">a spanned part </pc>. </source>
  </segment>
</unit>
...
```

Example for using `sc`/`ec` - Original:

```
...
<p>A paragraph where <u>the formatted text appears in more than one
    segment. The second sentence here.</u></p>
...
```

Extraction:

```
...
<unit id="u1">
  <originalData>
    <data id="d1">&lt;u&gt;</data>
    <data id="d2">&lt;/u&gt;</data>
  </originalData>
  <segment>
    <source>A paragraph where <sc id="sc1" dataRef=”d1” type="fmt"
        subType="xlf:u"/>the formatted text takes more than one segment.
        </source>
  </segment>
  <segment>
    <source> The second sentence here.<ec dataRef=”d2” startRef="sc1"/>
        </source>
  </segment>
</unit>
...
```

Example for using `ph` - Original:

```
...
<p>This sentence has a breakpoint<br/>inside.</p>
...
```

Extraction:

```
...
<unit id="u1">
  <originalData>
    <data id="d1">&lt;br/&gt;</data>
```

```
  </originalData>
  <segment>
    <source>This sentence has a breakpoint<ph id="ph1" dataRef="d1"
        type=”fmt” subType=”xlf:lb”/>inside. </source>
  </segment>
</unit>
...
```

Elements Within Text Value `Nested`

The sub-flow (i.e. element's content) should be stored in a different `unit` while the original element is replaced by a `ph` element and order of the flow defined by the `subFlows` attribute.

Example - Original:

```
...
<para>Some text with a figure:
  <figure>
    <title its:withinText="nested">Some image description</title>
    <mediaobject>
      <imageobject>
        <imagedata fileref="images/example.jpg" scale="75"/>
      </imageobject>
    </mediaobject>
  </figure>
</para>
...
```

Extraction:

```
...
<unit id="u1">
  <segment>
    <source>Some image description</source>
  </segment>
</unit>
<unit id="u2">
  <segment>
    <source>Some text with a figure: <ph id="ph1" subFlows="u1"/></source>
  </segment>
</unit>
...
```

All the sub-flows and the `unit` element which invokes them have to be in the same `file` element.

Elements Within Text Value `No`

In XLIFF 2.1 such element content should be stored in separate `unit` elements.

Example - Original:

```
...
<ul>
  <li>First sentence</li>
```

```
  <li>Second sentence</li>
</ul>
...
```

Extraction:

```
...
<unit id="u1">
  <segment>
    <source>First sentence</source>
  </segment>
</unit>
<unit id="u2">
  <segment>
    <source>Second sentence</source>
  </segment>
</unit>
...
```

Locale Filter

Expresses that a node is only applicable to certain locales. See [ITS] Locale Filter [https://www.w3.org/TR/its20/#LocaleFilter] for further details.

It is *recommended* that Locale Filter [http://www.w3.org/TR/its20/#LocaleFilter] metadata is fully consumed on *Extraction*, so that only the relevant source content is present in each *XLIFF Document* with the trgLang attributes set as per the Locale Filter [http://www.w3.org/TR/its20/#LocaleFilter] metadata.

Dependent on workflow specifics and business requirements, this data category can be most of the times fully represented by *Extraction* and *Merging* behavior without explicitly representing Locale Filter [http://www.w3.org/TR/its20/#LocaleFilter] metadata in *XLIFF Documents*. See the Locale Filter section within the defined categories section for the normative description of how this metadata can be explicitly represented if necessary.

Target Pointer

Is used to associate the node of a given source content (i.e., the content to be translated) and the node of its corresponding target content (i.e., the source content translated into a given target language). See [ITS] Target Pointer [http://www.w3.org/TR/its20/#target-pointer] for details.

This data category is not mapped to XLIFF but used by extracting and merging tools to get the source content from the original document and put back the translated content at its proper location.

Note that ITS processors working on XLIFF documents should use the following rule to locate the source and target content:

```
<its:targetPointerRule selector="//xlf:source"
    targetPointer="../xlf:target"/>
```

ID Value

The ID Value [http://www.w3.org/TR/its20#idvalue] data category indicates a value that can be used as a unique identifier for a given part of the content. As XLIFF identifiers

are not globally unique, this data category does cannot have a normative correspondence in XLIFF. Still the ID information could be represented in XLIFF, e.g. if there is an HTML file with id attributes, the attributes could be stored as names (e.g. with the XLIFF `name` attribute) or ids (with the XLIFF `id` attribute), yet being unique per XLIFF `file` element (not per *XLIFF Document*). In general the ID Value information is fully consumed by the *Extraction/Merge* behavior and there is no normative mapping relationship between ID Value as used in native formats and during the XLIFF Roundtrip.

Example - Original:

```
...
<p id="p1>A paragraph</p>
...
```

Extraction:

```
...
<unit id="u1" name="p1>
  <segment>
    <source>A paragraph</source>
  </segment>
</unit>
...
```

ITS Mapping Annotations

This lists all custom Annotations that are needed for [ITS] support in *XLIFF Documents* but are not available through *XLIFF Core* Annotations or other module specific annotations. Use of *XLIFF Core* Annotations for the ITS Mapping purposes is described in sections ITS data categories available through XLIFF Core and ITS data categories that have a partial overlap with XLIFF features sections of this ITS Module.

The following is the summary of internal links to all relevant Annotations:

- Generic Annotation
  - ITS Tools Annotation
- Annotations for Data Categories fully defined in the ITS Module
  - ITS Allowed Characters
  - ITS Domain Annotation
  - ITS Locale Filter Annotation
  - ITS Localization Quality Issue Annotation
  - ITS Localization Quality Rating Annotation
  - ITS Provenance Annotation
  - ITS Text Analysis Annotation
- Annotations for Data Categories partially defined in the ITS Module
  - ITS Language Information Annotation

- ITS MT Confidence Annotation
- ITS Terminology Annotation

Module Elements

All ITS Module elements belong to the `http://www.w3.org/2005/11/its` namespace. The ITS Module defines the following elements:

`<locQualityIssue>`, `<locQualityIssues>`, `<provenanceRecord>`, and `<provenanceRecords>`.

Tree Structure

Legend:

1 = one
+ = one or more
? = zero or one
* = zero, one or more

```
<locQualityIssues>
|
+---<locQualityIssue> +



<provenanceRecords>
|
+---<provenanceRecord> +
```

locQualityIssue

Localization Quality Issue - a standoff element to hold information about a single [ITS] defined Localization Quality Issue.

Contains:

This element is always empty.

Parents:

- `<locQualityIssues>`

Attributes:

- `locQualityIssueType`, *optional*
- `locQualityIssueComment`, *optional*
- `locQualityIssueSeverity`, *optional*
- `locQualityIssueProfileRef`, *optional*
- `locQualityIssueEnabled`, *optional*

Constraints

- At least one of the attributes `locQualityIssueType` or `locQualityIssueComment` *must* be set.

Processing Requirements

- For all *Agents*, when any of the attributes `locQualityIssueType`, `locQualityIssueComment`, `locQualityIssueSeverity`, `locQualityIssueProfileRef`, or `locQualityIssueEnabled` are declared on the `<locQualityIssue` element, these apply to the respective marker delimited inline spans of ITS Localization Issue Annotation, from which their enclosing `<locQualityIssues>` element is referenced.

locQualityIssues

Localization Quality Issues - a standoff wrapper element to group any number of single issue elements related to the same span of source or target content.

Contains:

- One or more `<locQualityIssue>` elements

Parents:

- `<unit>`

Attributes:

- `xml:id` , *required*

Constraints

- Each locQualityIssues element *should* be referenced by at least one `locQualityIssuesRef` attribute within the same `<unit>` element as per Constraints for the `locQualityIssuesRef` attribute.

Processing Requirements

- *Modifiers* detecting an orphaned locQualityIssues element *may* delete that locQualityIssues element.

provenanceRecord

Provenance Record - a standoff element to hold information of a single [ITS] defined Provenance Record.

Contains:

This element is always empty.

Parents:

- `<provenanceRecords>`

Attributes:

- `its:org`, *optional*
- `its:orgRef`, *optional*
- `its:person`, *optional*
- `its:personRef`, *optional*
- `its:revOrg`, *optional*
- `its:revOrgRef`, *optional*
- `its:revPerson`, *optional*

- `its:revPersonRef`, *optional*
- `its:revTool`, *optional*
- `its:revToolRef`, *optional*
- `its:tool`, *optional*
- `its:toolRef`, *optional*

Constraints

- At least one of the following *must* be set:
  - `its:org`,
  - `its:orgRef`,
  - `its:person`,
  - `its:personRef`,
  - `its:revOrg`,
  - `its:revOrgRef`,
  - `its:revPerson`,
  - `its:revPersonRef`,
  - `its:revTool`,
  - `its:revToolRef`,
  - `its:tool`,
  - `its:toolRef`,

Processing Requirements

- For all *Agents*, when any of the attributes `its:org`, `its:orgRef`, `its:person`, `its:personRef`, `its:revOrg`, `its:revOrgRef`, `its:revPerson`, `its:revPersonRef`, `its:revTool`, `its:revToolRef`, `its:tool`, or `its:toolRef` are declared on the `<provenanceRecord>` element, these apply to the respective structural elements' content or the marker delimited inline spans of ITS Provenance Annotation, from which their enclosing `<provenanceRecords>` element is referenced.

provenanceRecords

Provenance Records - a standoff wrapper element to group any number of single Provenance Record elements related to the same span of source or target content.

Contains:

- One or more `<itsm:provenanceRecord>` elements

Parents:

- `<unit>`
- `<group>`
- `<file>`

Attributes:

- `xml:id` , *required*

Constraints

- Each provenanceRecords element *should* be referenced by at least one `provenanceRecordsRef` attribute from the common parent element or one of the common parent's descendants as per Constraints for the `provenanceRecordsRef` attribute.

Processing Requirements

- *Modifiers* detecting an orphaned provenanceRecords element *may* delete that provenanceRecords element.

Module Attributes

The ITS Module uses the following attributes from the `http://www.w3.org/2005/11/its` namespace: `allowedCharacters`, `annotatorsRef`, `localeFilterList`, `localeFilterType`, `locQualityIssueComment`, `locQualityIssueEnabled`, `locQualityIssueProfileRef`, `locQualityIssuesRef`, `locQualityIssueSeverity`, `locQualityIssueType`, `locQualityRatingProfileRef`, `locQualityRatingScore`, `locQualityRatingScoreThreshold`, `locQualityRatingVote`, `locQualityRatingVoteThreshold`, `mtConfidence`, `org`, `orgRef`, `person`, `personRef`, `provenanceRecordsRef`, `revOrg`, `revOrgRef`, `revPerson`, `revPersonRef`, `revTool`, `revToolRef`, `taClassRef`, `taConfidence`, `taIdent`, `taIdentRef`, `taSource`, `termConfidence`, `tool`, `toolRef`, and `version`

The attributes defined in the ITS Module that belong to the `urn:oasis:names:tc:xliff:itsm:2.1` namespace are: `domains` and `lang`.

The ITS Module also uses the `xml:id` attribute.

Allowed Characters

AllowedCharacters - the `allowedCharacters` attribute is the [ITS] defined `allowedCharacters` attribute. See the allowedCharacters [https://www.w3.org/TR/its20/#allowedchars-local] definition in the [ITS] specification for details on the purpose of the attribute and permitted values.

Value description: See the allowedCharacters [https://www.w3.org/TR/its20/#allowedchars-local] definition in the [ITS] specification.

Default value: none.

Used in: `<mrk>` and `<sm>`.

See the ITS Allowed Characters Annotation for the normative usage description of this attribute when used inline; advanced Constraints follow from that normative usage description.

annotatorsRef

Annotators Reference - the `annotatorsRef` attribute holds provenance information about tools that produced [ITS] metadata. See [ITS] Tools Annotation [http://www.w3.org/TR/2013/REC-its20-20131029/#its-tool-annotation] mechanism.

Value description: Text.

Default value:: default values for this attribute depend on the element in which it is used:

- When used in `<file>`:

  The value is undefined.

- When used in any other admissible structural element (`<group>` or `<unit>`) or any of the elements defined in the ITS Module:

  The value of the `annotatorsRef` attribute of its parent element (which can be undefined).

- When used in annotations markers `<mrk>` or `<sm>`:

  The value of the `annotatorsRef` attribute of the innermost `<mrk>`, `<unit>`, or `<mtc:match>` element, element, in which the marker in question is located (which can be undefined).

- When used in top level module elements (`<mtc:match>`) :

  The value is undefined.

Constraints

- All Constraints that follow from [ITS] Tools Annotation [https://www.w3.org/TR/its20/#its-tool-annotation].

Note

The IRI part of the value string is used as the annotator identifier. The semantics of how the IRI identifies the ITS producing tool is not prescribed. Possible mechanisms are for instance: to encode information directly in the IRI, as parameters or similar; to reference an external resource that provides such information, an XML file, an RDF declaration and so on; or to reference another part of the document that provides such information.

Used in: `<file>` `<group>` `<unit>`, `<mrk>`, `<sm>`, `<mtc:match>`, `<ctr:revisions>`, or `<ctr:revision>` .

Processing Requirements

- All Processing Requirements that follow from [ITS] Tools Annotation [https://www.w3.org/TR/its20/#its-tool-annotation].

See the ITS Tools Annotation for the normative usage description of this attribute inline.

itsm:domains

Domains - the `itsm:domains` attribute expresses the [ITS] Domain [http://www.w3.org/TR/its20/#domain] data category.

Value description: The value is a text string, however commas if present separate distinct domain values within the string.

Default value:: default values for this attribute depend on the element in which it is used:

- When used in `<file>`:

  The value is undefined.

- When used in any other admissible structural element (`<group>` or `<unit>`):

  The value of the `domains` attribute of its parent element (which can be undefined).

- When used in annotations markers `<mrk>` or `<sm>`:

  The value of the `domains` attribute of the innermost `<mrk>`, `<unit>`, or `<mtc:match>` element, element, in which the marker in question is located (which can be undefined).

- When used in the `<mtc:match>` element:

  The value is undefined.

Used in: `<file>` `<group>` `<unit>`, `<mrk>`, `<sm>`, and `<mtc:match>`.

See the ITS Domain Annotation for the normative usage description of this attribute inline.

Warning

This attribute belongs to the `urn:oasis:names:tc:xliff:itsm:2.1` namespace that is being prefixed with `itsm:` throughout this specification, unlike the original W3C ITS namespace `http://www.w3.org/2005/11/its` that is being prefixed with `its:`.

xml:id

Identifier - the `id` attribute from the `http://www.w3.org/XML/1998/` namespace is used to identify a `<locQualityIssues>` or `<provenanceRecords>` element.

Value description: xs:ID

Default value: undefined

Used in:`<locQualityIssues>` and `<provenanceRecords>`.

Warning

Since the ITS Module reuses the W3C namespace `http://www.w3.org/2005/11/its` it cannot use `xs:NMTOKEN` identifiers as the *XLIFF Core* or other *Modules*. Implementers need to be aware that `xs:ID` has to be globally unique.

itsm:lang

Inline language information - the `itsm:lang` attribute specifies *an inline foreign language span* within the source or target content of the otherwise bilingual *XLIFF Document*. For example: `itsm:lang="fr-FR"` indicates the French language as spoken in France.

Note

This is NEVER used on structural elements that have their Language Information set by the *XLIFF Core* `xlf:srcLang` and `xlf:trgLang` attributes. It is not advisable

to use this attribute on structural elements even outside of XLIFF where the Language Information is typically given by the `xml:lang` attribute.

Value description: A language code as described in [BCP 47].

Default value:

The value of the `xml:lang` or `itsm:lang` attribute set or inherited on the parent element of the `<mrk>` or `<sm>` element in question.

Used in: `<mrk>` and `<sm>`.

See the ITS Language Information Annotation for the normative usage description of this attribute.

Note

`itsm:lang` is an attribute analogical to `xml:lang`. Unlike `xml:lang`, it is allowed on XLIFF inline Annotations. The normative behavior of this attribute results from the *XLIFF Core* behavior as further specified by the ITS Language Information Annotation.

Warning

This attribute belongs to the `urn:oasis:names:tc:xliff:itsm:2.1` namespace that is being prefixed with `itsm:` throughout this specification, unlike the original W3C ITS namespace `http://www.w3.org/2005/11/its` that is being prefixed with `its:`.

localeFilterList

LocaleFilterList - the `localeFilterList` attribute is the [ITS] defined `localeFilterList` attribute. See the localeFilterList [https://www.w3.org/TR/its20/#localefilter-local] definition in the [ITS] specification for details on the purpose of the attribute and permitted values.

Value description: See the localeFilterList [https://www.w3.org/TR/its20/#localefilter-local] definition in the [ITS] specification.

Default value: "*".

Used in: `<mrk>` and `<sm>`.

See the ITS Locale Filter Annotation for the normative usage description of this attribute when used inline; advanced Constraints follow from that normative usage description.

localeFilterType

LocaleFilterType - the `localeFilterType` attribute is the [ITS] defined `localeFilterType` attribute. See the localeFilterType [https://www.w3.org/TR/its20/#localefilter-local] definition in the [ITS] specification for details on the purpose of the attribute and permitted values.

Value description: See the localeFilterType [https://www.w3.org/TR/its20/#localefilter-local] definition in the [ITS] specification.

Default value: "include".

Used in: <mrk> and <sm>.

See the ITS Locale Filter Annotation for the normative usage description of this attribute when used inline; advanced Constraints follow from that normative usage description.

locQualityIssueComment

Localization Quality Issue Comment - the `locQualityIssueComment` attribute is the [ITS] defined `locQualityIssueComment` attribute.

This attribute is intended for human readable comments pertaining to or guidance how to address a specific Localization Quality Issue.

Value description: text string.

Default value:: undefined

Used in: <mrk> and <sm>, or in <locQualityIssue>.

See the ITS Localization Issue Annotation for the normative usage description of this attribute when used inline; advanced Constraints follow from that normative usage description.

See <locQualityIssue> for standoff usage Constraints.

locQualityIssueEnabled

Localization Quality Issue Enabled - the `locQualityIssueEnabled` attribute is the [ITS] defined `locQualityIssueEnabled` attribute.

This is a flag to enable or disable a particular issue.

Value description: `yes` when issue enabled , `no` otherwise.

Default value: `yes`.

Note

The attribute locQualityIssueEnabled set to `no` can be used for instance to disable false positives that were produced by an automated QA tool.

Used in: <mrk> and <sm>, or in <locQualityIssue>.

See the ITS Localization Issue Annotation for the normative usage description of this attribute when used inline; advanced Constraints follow from that normative usage description. See <locQualityIssue> for standoff usage Constraints.

locQualityIssueProfileRef

Localization Quality Issue Quality Model Profile Reference - the `locQualityIssueProfileRef` attribute is the [ITS] defined `locQualityIssueProfileRef` attribute.

This attribute references a quality model that has been used to identify and evaluate a particular issue.

Value description: IRI.

Default value: undefined

Note

It is strongly advised that the IRI value of the locQualityIssueProfileRef attribute is resolvable, so that human evaluators can find out about the referenced Quality Model.

Used in: `<mrk>` and `<sm>`, or in `<locQualityIssue>`.

See the ITS Localization Issue Annotation for the normative usage description of this attribute when used inline; advanced Constraints follow from that normative usage description.

See `<locQualityIssue>` for standoff usage Constraints.

locQualityIssuesRef

Localization Quality Standoff Reference - the `locQualityIssuesRef` attribute is the [ITS] defined `locQualityIssuesRef` attribute.

This attribute references the collection of Localization Issues that pertain to the content span from where the reference is declared.

Value description: IRI

Default value: undefined.

Used in: `<mrk>` and `<sm>`.

See the ITS Localization Issue Annotation for the normative usage description of this attribute.

Constraints

- The IRI value of the locQualityIssuesRef attribute *must* be an IRI referencing a `<locQualityIssues>` element within the same `<unit>`.
- Multiple locQualityIssuesRef attributes *may* reference the same `<locQualityIssues>` element.

Processing Requirements

- *Modifiers* removing the last `locQualityIssuesRef` attribute referencing a `locQualityIssues` element *must* delete that `locQualityIssues` element.

locQualityIssueSeverity

Localization Quality Issue Severity - the `locQualityIssueSeverity` attribute is the [ITS] defined `locQualityIssueSeverity` attribute.

This attribute provides the severity score for a particular issue, the higher the number the higher the severity. Tools are expected to interpret this score within their own severity rating system.

Value description: a decimal number between 0.0 and 100.0.

Default value: undefined

Warning

The locQualityIssueSeverity attribute is intended to be used in concert with the `locQualityIssueProfileRef` attribute that is to provide information on the applicable Quality Model. Without providng quality model information, the severity score between 0 and 100 is very likely to be useless and not interoperable.

Used in: `<mrk>` and `<sm>`, or in `<locQualityIssue>`.

See the ITS Localization Quality Issue Annotation for the normative usage description of this attribute when used inline; advanced Constraints follow from that normative usage description.

See `<locQualityIssue>` for standoff usage Constraints.

locQualityIssueType

Localization Quality Issue Type - the `locQualityIssueType` attribute is the [ITS] defined `locQualityIssueType` attribute.

Value description: a text string, exactly one value from the following list:

```
terminology
mistranslation
omission
untranslated
addition
duplication
inconsistency
grammar
legal
register
locale-specific-content
locale-violation
style
characters
misspelling
typographical
formatting
inconsistent-entities
numbers
markup
pattern-problem
whitespace
internationalization
length
non-conformance
uncategorized
other
```

For normative usage description and informative guidance for the above values, see [ITS] http://www.w3.org/TR/its20/#lqissue-typevalues.

Default value:: undefined

Used in: <mrk> and <sm>, or in <locQualityIssue>.

See the ITS Localization Issue Annotation for the normative usage description of this attribute when used inline; advanced Constraints follow from that normative usage description.

See <locQualityIssue> for standoff usage Constraints.

locQualityRatingProfileRef

Localization Quality Rating Quality Model Profile Reference - the locQualityRatingProfileRef attribute is the [ITS] defined locQualityRatingProfileRef attribute.

This attribute references a quality assessment model that has been used for the rating (either scoring or voting).

Value description: IRI.

Default value: default values for this attribute depend on the element in which it is used:

- When used in <file>:

  The value is undefined.

- When used in any other admissible structural element (<group> or <unit>):

  The value of the locQualityRatingProfileRef attribute of its parent element (which can be undefined).

- When used in annotations markers <mrk> or <sm>:

  The value of the locQualityRatingProfileRef attribute of the innermost <mrk>, <unit>, or <mtc:match> element, element, in which the marker in question is located (which can be undefined).

- When used in the <mtc:match> element:

  The value is undefined.

Note

It is strongly advised that the IRI value of the locQualityRatingProfileRef attribute is resolvable, so that human evaluators can find out about the referenced Quality Assessment Model.

Used in: <file> <group> <unit>, <mrk>, <sm>, and the <mtc:match> element..

See the ITS Localization Rating Annotation for the normative usage description of this attribute when used inline; advanced Constraints follow from that normative usage description.

See Localization Quality Rating on Structural Elements for advanced Constraints when used on strcutural elements (<file>, <group>, and <unit>).

See Localization Quality Rating in Translation Candidates Module for advanced Constraints when used within the Translation Candidates Module.

locQualityRatingScore

Localization Quality Rating Score - the `locQualityRatingScore` attribute is the [ITS] defined `locQualityRatingScore` attribute.

This attribute provides the quality rating score pertaining to a structural or inline portion of target text, the higher the number the better the quality rating. Tools are expected to interpret this score within their own quality rating system.

Value description: a decimal number between 0.0 and 100.0.

Default value: default values for this attribute depend on the element in which it is used:

- When used in `<file>`:

  The value is undefined.

- When used in any other admissible structural element (`<group>` or `<unit>`):

  The value of the `locQualityRatingScore` attribute of its parent element (which can be undefined).

- When used in annotations markers `<mrk>` or `<sm>`:

  The value of the `locQualityRatingScore` attribute of the innermost `<mrk>`, `<unit>`, or `<mtc:match>` element, in which the marker in question is located (which can be undefined).

  In the special case that the parent element of the marker is a `<mtc:match>` element, the value is inherited from the `mtc:matchQuality` attribute of the parent `<mtc:match>` (which can be undefined).

Warning

The locQualityRatingScore attribute is intended to be used in concert with the locQualityRatingProfileRef attribute that is to provide information on the applicable Quality Assessment Model and with the locQualityRatingScoreThreshold attribute. Without providng quality assessment model information and/or an acceptance threshold, the score between 0 and 100 is very likely to be useless and not interoperable.

Used in: `<file>`, `<group>`, `<unit>`, `<mrk>`, and `<sm>`.

See the ITS Localization Quality Rating Annotation for the normative usage description of this attribute when used inline; advanced Constraints follow from that normative usage description.

See `Localization Quality Rating on Structural Elements` for advanced Constraints when used on structural elements (`<file>`, `<group>`, and `<unit>`).

See `Localization Quality Rating in Translation Candidates Module` for advanced Constraints when used within the Translation Candidates Module.

locQualityRatingScoreThreshold

Localization Quality Rating Score Threshold - the `locQualityRatingScoreThreshold` attribute is the [ITS] defined `locQualityRatingScoreThreshold` attribute.

This attribute provides the quality rating score threshold pertaining to any `locQualityRatingScore` attribute in scope. Scores under the given threshold indicte a quality check fail.

Value description: a decimal number between 0.0 and 100.0.

Default value: default values for this attribute depend on the element in which it is used:

- When used in `<file>`:

  The value is undefined.

- When used in any other admissible structural element (`<group>` or `<unit>`):

  The value of the `locQualityRatingScoreThreshold` attribute of its parent element (which can be undefined).

- When used in annotations markers `<mrk>` or `<sm>`:

  The value of the `locQualityRatingScoreThreshold` attribute of the innermost `<mrk>`, `<unit>`, or `<mtc:match>` element, element, in which the marker in question is located (which can be undefined).

- When used in the `<mtc:match>` element:

  The value is undefined.

Warning

The locQualityRatingScoreThreshold attribute is intended to be used in concert with the locQualityRatingProfileRef attribute that is to provide information on the applicable Quality Assessment Model. Without providing quality assessment model information behind the acceptance threshold, the score between 0 and 100 is very likely to be useless and not interoperable.

Used in: `<file>`, `<group>`, `<unit>`, `<mrk>`, and `<sm>`.

See the ITS Localization Quality Rating Annotation for the normative usage description of this attribute when used inline; advanced Constraints follow from that normative usage description.

See Localization Quality Rating on Structural Elements for advanced Constraints when used on strcutural elements (`<file>`, `<group>`, and `<unit>`).

See Localization Quality Rating in Translation Candidates Module for advanced Constraints when used within the Translation Candidates Module.

locQualityRatingVote

Localization Quality Rating Vote - the `locQualityRatingVote` attribute is the [ITS] defined `locQualityRatingVote` attribute.

This attribute provides the quality rating voting (crowd assessment) results pertaining to a structural or inline portion of target text, the higher the number the more positive votes or the better margin of positive votes over negative votes. Tools are expected to interpret this value within their own quality rating system.

Value description: an Integer.

Default value: default values for this attribute depend on the element in which it is used:

- When used in `<file>`:

  The value is undefined.

- When used in any other admissible structural element (`<group>` or `<unit>`):

  The value of the `locQualityRatingVote` attribute of its parent element (which can be undefined).

- When used in annotations markers `<mrk>` or `<sm>`:

  The value of the `locQualityRatingVote` attribute of the innermost `<mrk>`, `<unit>`, or `<mtc:match>` element, element, in which the marker in question is located (which can be undefined).

- When used in the `<mtc:match>` element:

  The value is undefined.

Warning

The locQualityRatingVote attribute is intended to be used in concert with the locQualityRatingScoreThreshold attribute, that encodes the vote's success or failure criteria and ideally also the locQualityRatingProfileRef attribute that is to provide information on the applicable Quality Assessment Model. Without providng a success threshold or quality assessment model information, the integer encoding the voting (crowd assessment) results is very likely to be useless and not interoperable.

Used in: `<file>`, `<group>`, `<unit>`, `<mrk>`, `<sm>`, and `<mtc:match>`.

See the ITS Localization Quality Rating Annotation for the normative usage description of this attribute when used inline; advanced Constraints follow from that normative usage description.

See `Localization Quality Rating on Structural Elements` for advanced Constraints when used on strcutural elements (`<file>`, `<group>`, and `<unit>`).

See `Localization Quality Rating in Translation Candidates Module` for advanced Constraints when used within the Translation Candidates Module.

locQualityRatingVoteThreshold

Localization Quality Rating Vote Threshold - the `locQualityRatingVoteThreshold` attribute is the [ITS] defined `locQualityRatingVoteThreshold` attribute.

This attribute provides the minimum passing vote threshold for any Localization Quality Rating Votes that are in scope of the `locQualityRatingVoteThreshold` attribute.

Value description: an Integer.

Default value: default values for this attribute depend on the element in which it is used:

- When used in `<file>`:

  The value is undefined.

- When used in any other admissible structural element (<group> or <unit>):

  The value of the locQualityRatingVoteThreshold attribute of its parent element (which can be undefined).

- When used in annotations markers <mrk> or <sm>:

  The value of the locQualityRatingVoteThreshold attribute of the innermost <mrk>, <unit>, or <mtc:match> element, element, in which the marker in question is located (which can be undefined).

- When used in the <mtc:match> element:

  The value is undefined.

Warning

The locQualityRatingVoteThreshold attribute is intended to be used in concert with the locQualityRatingProfileRef attribute that is to provide information on the applicable Quality Assessment Model. Without providng the quality assessment model information, the voting threshold integer is very likely to be useless and not interoperable.

Used in: <file>, <group>, <unit>, <mrk>, <sm>, and <mtc:match>.

See the ITS Localization Quality Rating Annotation for the normative usage description of this attribute when used inline; advanced Constraints follow from that normative usage description.

See Localization Quality Rating on Structural Elements for advanced Constraints when used on strcutural elements (<file>, <group>, and <unit>).

See Localization Quality Rating in Translation Candidates Module for advanced Constraints when used within the Translation Candidates Module.

mtConfidence

Machine Translation Confidence - the mtConfidence attribute is the [ITS] defined mtConfidence attribute.

Value description: floating point number between 0 and 1.

The number represents the self reported confidence of the application or service providing the MT Confidence [http://www.w3.org/TR/its20/#mtconfidence] metadata, the higher the better.

Default value:: undefined

Used in: <mrk>, and <sm>.

Constraints

- When the attribute mtConfidence is set, the element where it is set *must* be in the scope of an annotatorsRef attribute with the [ITS]Data category identifier [https://www.w3.org/TR/its20/#datacategories-overview] part of exactly one list value equal to the string mt-confidence.

See the ITS MT Confidence Annotation for the full normative usage description of this attribute. Other advanced Constraints follow from that normative usage description.

Processing Requirements

- *Writers must* use the `mtc:matchQuality` attribute to express the `MTConfidence` attribute on an `<mtc:match>` element.
  - The floating point number between 0 and 1 *must* be expressed as a decimal number between 0.0 and 100.0 [%].
  - The `mtc:matchQuality` attribute used by the *Writer* to express the `MTConfidence` attribute *must* be in scope of an `annotatorsRef` attribute with the [ITS]Data category identifier [https://www.w3.org/TR/its20/#datacategories-overview] part of exactly one list value equal to the string `mt-confidence`.
- *Modifiers may* use this `MTConfidence` attribute, when populating the *XLIFF Core* `<target>` elements with exact unmodified MT matches from `<mtc:match>` elements with the `mtc:matchQuality` attribute set and in scope of an `annotatorsRef` attribute with the [ITS]Data category identifier [https://www.w3.org/TR/its20/#datacategories-overview] part of exactly one list value equal to the string `mt-confidence`.
  - The decimal number between 0.0 and 100.0 [%] *must* be expressed as a floating point number between 0 and 1.

org

Organization - the `org` attribute is the [ITS] defined `org` attribute.

Value description: Text

The text string is supposed to identify an organizational translation agent as per Organizational provenance information [https://www.w3.org/TR/its20/#provenanceDefs].

Default value: default values for this attribute depend on the element in which it is used:

- When used in any admissible element WITH the `provenanceRecordsRef` attribute set:

  The value of the `org` attribute of the first `<provenanceRecord>` element with the `org` attribute set within the referenced `<provenanceRecords>` element (which can be undefined).
- When used in any admissible element WITHOUT the `provenanceRecordsRef` attribute set:

  The default values depending on the element in which it is used:
  - When used in `<file>` or `<its:provencanceRecord>`:

    The value is undefined.
  - When used in any other admissible structural element (`<group>` or `<unit>`):

    The value of the `org` attribute of its parent element (which can be undefined).
  - When used in annotations markers `<mrk>` or `<sm>`:

The value of the `org` attribute of the innermost `<mrk>`, `<unit>`, `<mtc:match>`, or `<ctr:revision>` element, in which the marker in question is located (which can be undefined).

Used in: `<file>`, `<goup>`, `<unit>`, `<mrk>`, `<sm>`, `<provenanceRecord>`, `<mtc:match>`, and `<ctr:revision>`.

See the ITS Provenance Annotation for the normative usage description of this attribute on inline elements. Advanced Constraints follow from that normative usage description.

orgRef

Organization Reference - the `orgRef` attribute is the [ITS] defined `orgRef` attribute.

Value description: IRI

The IRI is supposed to resolve as human or machine readable Organizational provenance information [https://www.w3.org/TR/its20/#provenanceDefs].

Default value: default values for this attribute depend on the element in which it is used:

- When used in any admissible element WITH the `provenanceRecordsRef` attribute set:

  The value of the `orgRef` attribute of the first `<provenanceRecord>` element with the `orgRef` attribute set within the referenced `<provenanceRecords>` element (which can be undefined).

- When used in any admissible element WITHOUT the `provenanceRecordsRef` attribute set:

  The default values depening on the element in which it is used:

  - When used in `<file>` or `<its:provencanceRecord>`:

    The value is undefined.

  - When used in any other admissible structural element (`<group>` or `<unit>`):

    The value of the `orgRef` attribute of its parent element (which can be undefined).

  - When used in annotations markers `<mrk>` or `<sm>`:

    The value of the `orgRef` attribute of the innermost `<mrk>`, `<unit>`, `<mtc:match>`, or `<ctr:revision>` element, in which the marker in question is located (which can be undefined).

Used in: `<file>`, `<goup>`, `<unit>`, `<mrk>`, `<sm>`, `<provenanceRecord>`, `<mtc:match>`, and `<ctr:revision>`.

See the ITS Provenance Annotation for the normative usage description of this attribute on inline elements. Advanced Constraints follow from that normative usage description.

person

Person - the `person` attribute is the [ITS] defined `person` attribute.

Value description: Text

The text string is supposed to identify a human translation agent as per Human provenance information [https://www.w3.org/TR/its20/#provenanceDefs].

Default value: default values for this attribute depend on the element in which it is used:

- When used in any admissible element WITH the `provenanceRecordsRef` attribute set:

  The value of the `person` attribute of the first `<provenanceRecord>` element with the `person` attribute set within the referenced `<provenanceRecords>` element (which can be undefined).

- When used in any admissible element WITHOUT the `provenanceRecordsRef` attribute set:

  The default values depending on the element in which it is used:

  - When used in `<file>` or `<its:provencanceRecord>`:

    The value is undefined.

  - When used in any other admissible structural element (`<group>` or `<unit>`):

    The value of the `person` attribute of its parent element (which can be undefined).

  - When used in annotations markers `<mrk>` or `<sm>`:

    The value of the `person` attribute of the innermost `<mrk>`, `<unit>`, `<mtc:match>`, or `<ctr:revision>` element, in which the marker in question is located (which can be undefined).

Used in: `<file>`, `<goup>`, `<unit>`, `<mrk>`, `<sm>`, `<provenanceRecord>`, `<mtc:match>`, and `<ctr:revision>`.

See the ITS Provenance Annotation for the normative usage description of this attribute on inline elements. Advanced Constraints follow from that normative usage description.

personRef

Person Reference - the `personRef` attribute is the [ITS] defined `personRef` attribute.

Value description: IRI

The IRI is supposed to resolve as human or machine readable Human provenance information [https://www.w3.org/TR/its20/#provenanceDefs].

Default value: default values for this attribute depend on the element in which it is used:

- When used in any admissible element WITH the `provenanceRecordsRef` attribute set:

  The value of the `personRef` attribute of the first `<provenanceRecord>` element with the `personRef` attribute set within the referenced `<provenanceRecords>` element (which can be undefined).

- When used in any admissible element WITHOUT the `provenanceRecordsRef` attribute set:

  The default values depening on the element in which it is used:

- When used in `<file>` or `<its:provencanceRecord>`:

  The value is undefined.

- When used in any other admissible structural element (`<group>` or `<unit>`):

  The value of the `personRef` attribute of its parent element (which can be undefined).

- When used in annotations markers `<mrk>` or `<sm>`:

  The value of the `personRef` attribute of the innermost `<mrk>`, `<unit>`, `<mtc:match>`, or `<ctr:revision>` element, in which the marker in question is located (which can be undefined).

Used in: `<file>`, `<goup>`, `<unit>`, `<mrk>`, `<sm>`, `<provenanceRecord>`, `<mtc:match>`, and `<ctr:revision>`.

See the ITS Provenance Annotation for the normative usage description of this attribute on inline elements. Advanced Constraints follow from that normative usage description.

provenanceRecordsRef

Provenance Records Standoff Reference - the `provenanceRecordsRef` attribute is the [ITS] defined `provenanceRecordsRef` attribute.

This attribute references the collection of Provenance Records that pertain to the content span or structural element content from where the reference is declared.

Value description: IRI

Default value: undefined.

Used in: `<file>`, `<goup>`, `<unit>`, `<mrk>`, `<sm>`, `<mtc:match>`, and `<ctr:revision>`.

See the ITS Localization Issue Annotation for the normative usage description of this attribute inline.

Constraints

- Multiple provenanceRecordsRef attributes *may* reference the same `<provenanceRecords>` element.
- In case the provenanceRecordsRef attribute is used on an `<mrk>` or `<sm>` element,
  - The value of the provenanceRecordsRef attribute *must* be an IRI referencing a `<provenanceRecords>` element within the innermost enclosing `<unit>`, `<mtc:match>`, or `<ctr:revision>`element.
- In case the provenanceRecordsRef attribute is used on a `<file>`, `<goup>`, or `<unit>` element,
  - The value of the provenanceRecordsRef attribute *must* be an IRI referencing a `<provenanceRecords>` element within the same element or its ancestor element.

Processing Requirements

- *Modifiers* removing the last `provenanceRecordsRef` attribute referencing a `provenanceRecords` element *must* delete that `provenanceRecords` element.

revOrg

Organization - the `revOrg` attribute is the [ITS] defined `revOrg` attribute.

Value description: Text

The text string is supposed to identify an organizational translation agent as per Organizational provenance information [https://www.w3.org/TR/its20/#provenanceDefs].

Default value: default values for this attribute depend on the element in which it is used:

- When used in any admissible element WITH the `provenanceRecordsRef` attribute set:

  The value of the `revOrg` attribute of the first `<provenanceRecord>` element with the `revOrg` attribute set within the referenced `<provenanceRecords>` element (which can be undefined).

- When used in any admissible element WITHOUT the `provenanceRecordsRef` attribute set:

  The default values depening on the element in which it is used:

  - When used in `<file>` or `<its:provencanceRecord>`:

    The value is undefined.

  - When used in any other admissible structural element (`<group>` or `<unit>`):

    The value of the `revOrg` attribute of its parent element (which can be undefined).

  - When used in annotations markers `<mrk>` or `<sm>`:

    The value of the `revOrg` attribute of the innermost `<mrk>`, `<unit>`, `<mtc:match>`, or `<ctr:revision>` element, in which the marker in question is located (which can be undefined).

Used in: `<file>`, `<goup>`, `<unit>`, `<mrk>`, `<sm>`, `<provenanceRecord>`, `<mtc:match>`, and `<ctr:revision>`.

See the ITS Provenance Annotation for the normative usage description of this attribute on inline elements. Advanced Constraints follow from that normative usage description.

revOrgRef

Revision Organization Reference - the `revOrgRef` attribute is the [ITS] defined `revOrgRef` attribute.

Value description: IRI

The IRI is supposed to resolve as human or machine readable Organizational revision provenance information [https://www.w3.org/TR/its20/#provenanceDefs].

Default value: default values for this attribute depend on the element in which it is used:

- When used in any admissible element WITH the `provenanceRecordsRef` attribute set:

The value of the `revOrgRef` attribute of the first `<provenanceRecord>` element with the `revOrgRef` attribute set within the referenced `<provenanceRecords>` element (which can be undefined).

- When used in any admissible element WITHOUT the `provenanceRecordsRef` attribute set:

  The default values depending on the element in which it is used:

  - When used in `<file>` or `<its:provencanceRecord>`:

    The value is undefined.

  - When used in any other admissible structural element (`<group>` or `<unit>`):

    The value of the `revOrgRef` attribute of its parent element (which can be undefined).

  - When used in annotations markers `<mrk>` or `<sm>`:

    The value of the `revOrgRef` attribute of the innermost `<mrk>`, `<unit>`, `<mtc:match>`, or `<ctr:revision>` element, in which the marker in question is located (which can be undefined).

Used in: `<file>`, `<goup>`, `<unit>`, `<mrk>`, `<sm>`, `<provenanceRecord>`, `<mtc:match>`, and `<ctr:revision>`.

See the ITS Provenance Annotation for the normative usage description of this attribute on inline elements. Advanced Constraints follow from that normative usage description.

revPerson

Revision Person - the `revPerson` attribute is the [ITS] defined `revPerson` attribute.

Value description: Text

The text string is supposed to identify a human translation revision agent as per Human revision provenance information [https://www.w3.org/TR/its20/#provenanceDefs].

Default value: default values for this attribute depend on the element in which it is used:

- When used in any admissible element WITH the `provenanceRecordsRef` attribute set:

  The value of the `revPerson` attribute of the first `<provenanceRecord>` element with the `revPerson` attribute set within the referenced `<provenanceRecords>` element (which can be undefined).

- When used in any admissible element WITHOUT the `provenanceRecordsRef` attribute set:

  The default values depening on the element in which it is used:

  - When used in `<file>` or `<its:provencanceRecord>`:

    The value is undefined.

  - When used in any other admissible structural element (`<group>` or `<unit>`):

    The value of the `revPerson` attribute of its parent element (which can be undefined).

- When used in annotations markers <mrk> or <sm>:

  The value of the revPerson attribute of the innermost <mrk>, <unit>, <mtc:match>, or <ctr:revision> element, in which the marker in question is located (which can be undefined).

Used in: <file>, <goup>, <unit>, <mrk>, <sm>, <provenanceRecord>, <mtc:match>, and <ctr:revision>.

See the ITS Provenance Annotation for the normative usage description of this attribute on inline elements. Advanced Constraints follow from that normative usage description.

revPersonRef

Revision Person Reference - the revPersonRef attribute is the [ITS] defined revPersonRef attribute.

Value description: IRI

The IRI is supposed to resolve as human or machine readable Human revision provenance information [https://www.w3.org/TR/its20/#provenanceDefs].

Default value: default values for this attribute depend on the element in which it is used:

- When used in any admissible element WITH the provenanceRecordsRef attribute set:

  The value of the revPersonRef attribute of the first <provenanceRecord> element with the revPersonRef attribute set within the referenced <provenanceRecords> element (which can be undefined).

- When used in any admissible element WITHOUT the provenanceRecordsRef attribute set:

  The default values depening on the element in which it is used:

  - When used in <file> or <its:provencanceRecord>:

    The value is undefined.

  - When used in any other admissible structural element (<group> or <unit>):

    The value of the revPersonRef attribute of its parent element (which can be undefined).

  - When used in annotations markers <mrk> or <sm>:

    The value of the revPersonRef attribute of the innermost <mrk>, <unit>, <mtc:match>, or <ctr:revision> element, in which the marker in question is located (which can be undefined).

Used in: <file>, <goup>, <unit>, <mrk>, <sm>, <provenanceRecord>, <mtc:match>, and <ctr:revision>.

See the ITS Provenance Annotation for the normative usage description of this attribute on inline elements. Advanced Constraints follow from that normative usage description.

revTool

Revision Tool - the `revTool` attribute is the [ITS] defined `revTool` attribute.

Value description: Text

The text string is supposed to identify a software tool translation revision agent as per Tool-related revision provenance information [https://www.w3.org/TR/its20/#provenanceDefs].

Default value: default values for this attribute depend on the element in which it is used:

- When used in any admissible element WITH the `provenanceRecordsRef` attribute set:

  The value of the `revTool` attribute of the first `<provenanceRecord>` element with the `revTool` attribute set within the referenced `<provenanceRecords>` element (which can be undefined).

- When used in any admissible element WITHOUT the `provenanceRecordsRef` attribute set:

  The default values depening on the element in which it is used:

  - When used in `<file>` or `<its:provencanceRecord>`:

    The value is undefined.

  - When used in any other admissible structural element (`<group>` or `<unit>`):

    The value of the `revTool` attribute of its parent element (which can be undefined).

  - When used in annotations markers `<mrk>` or `<sm>`:

    The value of the `revTool` attribute of the innermost `<mrk>`, `<unit>`, `<mtc:match>`, or `<ctr:revision>` element, in which the marker in question is located (which can be undefined).

Used in: `<file>`, `<goup>`, `<unit>`, `<mrk>`, `<sm>`, `<provenanceRecord>`, `<mtc:match>`, and `<ctr:revision>`.

See the ITS Provenance Annotation for the normative usage description of this attribute on inline elements. Advanced Constraints follow from that normative usage description.

revToolRef

Revision Tool Reference - the `revToolRef` attribute is the [ITS] defined `revToolRef` attribute.

Value description: IRI

The IRI is supposed to resolve as human or machine readable Tool-related revision provenance information [https://www.w3.org/TR/its20/#provenanceDefs].

Default value: default values for this attribute depend on the element in which it is used:

- When used in any admissible element WITH the `provenanceRecordsRef` attribute set:

The value of the `revToolRef` attribute of the first `<provenanceRecord>` element with the `revToolRef` attribute set within the referenced `<provenanceRecords>` element (which can be undefined).

- When used in any admissible element WITHOUT the `provenanceRecordsRef` attribute set:

  The default values depening on the element in which it is used:

  - When used in `<file>` or `<its:provencanceRecord>`:

    The value is undefined.

  - When used in any other admissible structural element (`<group>` or `<unit>`):

    The value of the `revToolRef` attribute of its parent element (which can be undefined).

  - When used in annotations markers `<mrk>` or `<sm>`:

    The value of the `revToolRef` attribute of the innermost `<mrk>`, `<unit>`, `<mtc:match>`, or `<ctr:revision>` element, in which the marker in question is located (which can be undefined).

Used in: `<file>`, `<goup>`, `<unit>`, `<mrk>`, `<sm>`, `<provenanceRecord>`, `<mtc:match>`, and `<ctr:revision>`.

See the ITS Provenance Annotation for the normative usage description of this attribute on inline elements. Advanced Constraints follow from that normative usage description.

taClassRef

Text Analysis Class Reference - the `taClassRef` attribute is the [ITS] defined `taClassRef` attribute.

Value description: IRI

The IRI is supposed to resolve as human or machine readable Entity type / concept class [http://www.w3.org/TR/its20/#textAnalysis-info-pieces] information.

Default value:: undefined

Used in: `<mrk>`, and `<sm>`.

See the ITS Text Analysis Annotation for the normative usage description of this attribute. Advanced Constraints follow from that normative usage description.

taConfidence

Text Analysis Confidence - the `taConfidence` attribute is the [ITS] defined `taConfidence` attribute.

Value description: floating point number between 0 and 1.

The number represents the self reported confidence of the application or service providing the Text Analysis [http://www.w3.org/TR/its20/#textanalysis] metadata, the higher the better.

Default value:: undefined

Used in: <mrk>, and <sm>.

Constraints

- When the attribute taConfidence is set, the element where it is set *must* be in the scope of an its:annotatorsRef attribute with the [ITS]Data category identifier [https://www.w3.org/TR/its20/#datacategories-overview] part of exactly one list value equal to the string text-analysis.

See the ITS Text Analysis Annotation for the full normative usage description of this attribute. Other advanced Constraints follow from that normative usage description.

taIdent

Text Analysis Concept Identifier - the taIdent attribute is the [ITS] defined taIdent attribute.

Value description: text string

The text string is supposed to be a human or machine redable identifier of a concept within a collection of text analysis concept resources, in the sense of an identifier of the concept in the collection [http://www.w3.org/TR/its20/#textAnalysis-info-pieces].

Default value:: undefined

Used in: <mrk>, and <sm>.

Constraints

- When the attribute taIdent is set, the taSource attribute *must* be set as well.

See the ITS Text Analysis Annotation for the full normative usage description of this attribute. Other advanced Constraints follow from that normative usage description.

taIdentRef

Text Analysis Identifier - the taIdentRef attribute is the [ITS] defined taIdentRef attribute.

Value description: IRI

The IRI is supposed to reference an external resource for the disambiguated entity in the sense of identifier of the text analysis target [http://www.w3.org/TR/its20/#textAnalysis-info-pieces].

Default value:: undefined

Used in: <mrk>, and <sm>.

See the ITS Text Analysis Annotation for the normative usage description of this attribute. Advanced Constraints follow from that normative usage description.

taSource

Text Analysis Source - the taSource attribute is the [ITS] defined taSource attribute.

Value description: text string

The text string is supposed to be a human or machine redable name of a collection of text analysis concept resources, in the sense of an identifier of the collection source [http://www.w3.org/TR/its20/#textAnalysis-info-pieces].

Default value:: undefined

Used in: `<mrk>`, and `<sm>`.

Constraints

- When the attribute `taSource` is set, the `taIdent` attribute *must* be set as well.

See the ITS Text Analysis Annotation for the full normative usage description of this attribute. Other advanced Constraints follow from that normative usage description.

termConfidence

Terminology Confidence - the `termConfidence` attribute is the [ITS] defined `termConfidence` attribute.

Value description: floating point number between 0 and 1.

The number represents the self reported confidence of the application or service providing the Terminology [http://www.w3.org/TR/its20/#terminology] metadata, the higher the better.

Default value:: undefined

Used in: `<mrk>`, and `<sm>`.

Constraints

- When the attribute `termConfidence` is set, the element where it is set *must* be in the scope of an `its:annotatorsRef` attribute with the [ITS]Data category identifier [https://www.w3.org/TR/its20/#datacategories-overview] part of exactly one list value equal to the string `terminology`.

See the ITS Terminology Annotation for the full normative usage description of this attribute. Other advanced Constraints follow from that normative usage description.

tool

Tool - the `tool` attribute is the [ITS] defined `tool` attribute.

Value description: Text

The text string is supposed to identify a software tool translation agent as per Tool-related provenance information [https://www.w3.org/TR/its20/#provenanceDefs].

Default value: default values for this attribute depend on the element in which it is used:

- When used in any admissible element WITH the `provenanceRecordsRef` attribute set:

  The value of the `tool` attribute of the first `<provenanceRecord>` element with the `tool` attribute set within the referenced `<provenanceRecords>` element (which can be undefined).

- When used in any admissible element WITHOUT the `provenanceRecordsRef` attribute set:

  The default values depening on the element in which it is used:

  - When used in `<file>` or `<its:provencanceRecord>`:

    The value is undefined.

  - When used in any other admissible structural element (`<group>` or `<unit>`):

    The value of the `tool` attribute of its parent element (which can be undefined).

  - When used in annotations markers `<mrk>` or `<sm>`:

    The value of the `tool` attribute of the innermost `<mrk>`, `<unit>`, `<mtc:match>`, or `<ctr:revision>` element, in which the marker in question is located (which can be undefined).

Used in: `<file>`, `<goup>`, `<unit>`, `<mrk>`, `<sm>`, `<provenanceRecord>`, `<mtc:match>`, and `<ctr:revision>`.

See the ITS Provenance Annotation for the normative usage description of this attribute on inline elements. Advanced Constraints follow from that normative usage description.

toolRef

Tool Reference - the `toolRef` attribute is the [ITS] defined `toolRef` attribute.

Value description: IRI

The IRI is supposed to resolve as human or machine readable Tool-related provenance information [https://www.w3.org/TR/its20/#provenanceDefs].

Default value: default values for this attribute depend on the element in which it is used:

- When used in any admissible element WITH the `provenanceRecordsRef` attribute set:

  The value of the `toolRef` attribute of the first `<provenanceRecord>` element with the `toolRef` attribute set within the referenced `<provenanceRecords>` element (which can be undefined).

- When used in any admissible element WITHOUT the `provenanceRecordsRef` attribute set:

  The default values depening on the element in which it is used:

  - When used in `<file>` or `<its:provencanceRecord>`:

    The value is undefined.

  - When used in any other admissible structural element (`<group>` or `<unit>`):

    The value of the `toolRef` attribute of its parent element (which can be undefined).

  - When used in annotations markers `<mrk>` or `<sm>`:

The value of the `toolRef` attribute of the innermost `<mrk>`, `<unit>`, `<mtc:match>`, or `<ctr:revision>` element, in which the marker in question is located (which can be undefined).

Used in: `<file>`, `<goup>`, `<unit>`, `<mrk>`, `<sm>`, `<provenanceRecord>`, `<mtc:match>`, and `<ctr:revision>`.

See the ITS Provenance Annotation for the normative usage description of this attribute on inline elements. Advanced Constraints follow from that normative usage description.

version

ITS Version - the `version` attribute is the [ITS] `version` [https://www.w3.org/TR/its20/#its-version-attribute] attribute.

Value description: The value is a text string restricted to the string `2.0`.

Default value:: default values for this attribute depend on the element in which it is used:

- When used in `<xliff>`:

  The value is undefined.

- When used in any other admissible *XLIFF Core* structural element (`<file>`, `<group>` or `<unit>`):

  The value of the `version` attribute of its parent element (which can be undefined).

- When used in annotations markers `<mrk>` or `<sm>`:

  The value of the `version` attribute of the innermost `<mrk>`, `<unit>`, or `<mtc:match>` element, element, in which the marker in question is located (which can be undefined).

- When used in the `<mtc:match>` element:

  The value is undefined.

- When used in any of the ITS Module defined elements:

  The value of the `version` attribute of its parent element (which can be undefined).

Used in: `<xliff>`, `<file>` `<group>` `<unit>`, `<mrk>`, `<sm>`, `<mtc:match>`, `<its:locQualityIssue>`, `<its:locQualityIssues>`, `<its:provenanceRecord>`, and `<its:provenanceRecords>`.

Example file

Example 28. ITS Data Categories Example

The following example file includes markup related to several ITS 2.0 data categories.

```
<!-- xliff-its-example.xlf: Example file that shows several features of
using ITS 2.0 as part of XLIFF 2.1
Version: 0.2.1
Date: 04 April 2017 -->

<xliff version="2.1" xmlns="urn:oasis:names:tc:xliff:document:2.0"
```

```
  srcLang="en" trgLang="fr"
  xmlns:its="http://www.w3.org/2005/11/its" its:version="2.0"
  xmlns:itsm="urn:oasis:names:tc:xliff:itsm:2.1">
<!-- Each unit in the file element shows one ITS data category. -->
<!-- The its:annotatorsRef attribute inherits through the whole file
    but is only relevant for some elements-->
<file id="f1" its:annotatorsRef="allowed-characters|http://example.com/myAllo
              terminology|http://example.com/mytermTool
              localization-quality-issue|http://example.com/anotherQualityChe
  <unit id="u1">
    <its:locQualityIssues xml:id="lqi1">
      <its:locQualityIssue locQualityIssueType="misspelling"
          locQualityIssueComment="'c'es' is unknown. Could be 'c'est'"
          locQualityIssueSeverity="50"/>
      <its:locQualityIssue locQualityIssueType="grammar"
          locQualityIssueComment="Sentence is not capitalized"
          locQualityIssueSeverity="30"/>
    </its:locQualityIssues>
    <its:provenanceRecords xml:id="prov1">
      <its:provenanceRecord revPerson="Franta Kocourek"
          revOrg="Kocourkov s.r.o."/>
      <its:provenanceRecord person="Honza Novák"
          org="P eklady Novák, sro" tool="GreatCATTool"/>
      <its:provenanceRecord tool="Microsoft Hub"/>
    </its:provenanceRecords>
    <its:provenanceRecords xml:id="prov2">
      <its:provenanceRecord revPerson="Kv to  Z ídkaveselý"
          revOrg="CoolCopy"/>
      <its:provenanceRecord revTool="ACME QA Checker"
          revOrg="CoolCopy"/>
      <its:provenanceRecord revPerson="Franta Kocourek"
          revOrg="Kocourkov s.r.o."/>
      <its:provenanceRecord person="Honza Novák"
          org="Preklady Novák, sro" tool="GreatCATTool"/>
      <its:provenanceRecord tool="Microsoft Hub"/>
    </its:provenanceRecords>
    <notes>
      <note id="1" priority="1">Check with terminology engineer
          </note>
    </notes>

    <!-- Example for allowed characters data category -->

    <segment>
      <source>
        <mrk id="m1" type="its:generic"
            its:allowedCharacters="[a-ZA-Z]">Text</mrk>
      </source>
    </segment>

    <!-- Example for domain data category -->

    <segment>
      <source>Text in the domain domain1</source>
    </segment>

    <!-- Example for locale filter data category -->
```

```
<segment>
  <source>Text <pc id="2"><mrk id="m2" type="its:generic"
      its:localeFilterList="fr" its:localeFilterType="exclude">
      text</mrk></pc></source>
</segment>

<!-- Example for localization quality issue data category. The
    standoff information in its:locQualityIssues has the
    annotatorsRef information from this element: <mrk id="m1"
    type="its:generic" its:locQualityIssuesRef="#its=lqi1"
    its:annotatorsRef="localization-quality-issue|http://example.com/myQu
-->

<segment>
  <source>This is the content</source>
  <target>
    <mrk id="m3" type="its:generic" its:locQualityIssuesRef="#its=lqi1"
        its:annotatorsRef="localization-quality-issue|http://example.com/
        le contenu</mrk>
  </target>
</segment>

<!-- Example for localization quality rating data category -->

<segment>
  <source>Some text and a term</source>
  <target>Du texte et un <mrk id="m4" type="its:generic"
      its:locQualityRatingVote="37"
      its:locQualityRatingVoteThreshold="15"
      its:locQualityRatingProfileRef="http://example.org/qaModel
          /v13">terme</mrk></target>
</segment>

<!-- Example for text analytics data category -->

<segment>
  <source>
    <mrk id="m5" type="its:generic"
        its:taClassRef="http://nerd.eurecom.fr/ontology#Place"
        its:taIdentRef="http://dbpedia.org/resource/Arizona">
        Arizona</mrk>
  </source>
</segment>

<!-- Example for terminology data category, expressed via native XLIFF ma

<segment>
  <source>This is a <mrk id="m6" type="comment" ref="#n=1">
      motherboard</mrk>.</source>
</segment>

<!-- Example for terminology data category, expressed via native XLIFF ma

<segment>
  <source>Text with a <pc id="3"><mrk id="m7" type="term"
      ref="http://en.wikipedia.org/wiki/Terminology"
      its:termConfidence="0.9">term</mrk></pc>.</source>
</segment>
```

```
        <!-- Example for language information data category -->

        <segment>
          <source>Span of text <pc id="4"><mrk id="m8" itsm:lang="fr"
              type="its:generic">en français</mrk></pc>.</source>
        </segment>

        <!-- Example for provenance information data category -->

        <segment>
          <source>Economy has been growing in 2016.</source>
          <target>
            <mrk id="m9" type="its:generic"
                its:provenanceRecordsRef="#its=prov1">Hospodá ství v pr behu
                roku 2016 rostlo. </mrk>
          </target>
        </segment>
        <segment>
          <source>Prognosis for 2017 is unclear.</source>
          <target>
             <mrk id="m10" type="its:generic"
                 its:provenanceRecordsRef="#its=prov2">P edpov   o ekávaného r st
                 pro rok 2017 je nejasná. </mrk>
          </target>
        </segment>

        <!-- Example for MT confidence data category -->

        <segment>
          <source>
            <mrk id="m11" type="its:generic" its:mtConfidence="0.8982"
                its:annotatorsRef="mt-confidence|MTServices-XYZ">Some
                Machine Translated text.</mrk>
          </source>
        </segment>

        <!-- Example for Translate data category, expressed via native XLIFF mark

        <segment>
          <source><mrk translate="no" id="m12">Non-translatable text
              </mrk></source>
        </segment>

      </unit>
    </file>
  </xliff>
```

A. Media Type Registration Template for XLIFF Version 2.0 and higher Versions

This Appendix is based on the Committee Specification 01 of the Media Type Registration Template for XLIFF Version 2.0 [http://docs.oasis-open.org/xliff/xliff-media/v2.0/cs01/xliff-media-v2.0-cs01.html] published on 22 September 2014, which is also the latest version of the Registration Template,

as the TC decided to merge the template (with some necessary editorial updates) with this XLIFF Version 2.2 specification. Additionally, the Security Considerations have been substantially expanded based on an IESG Expert feedback, as per this archived conversation https://lists.oasis-open.org/archives/xliff/201502/msg00004.html. XLIFF 2.0 was provisionally registered as `application/xliff+xml` in the IANA Provisional Standard Media Type Registry [https://www.iana.org/assignments/provisional-standard-media-types/provisional-standard-media-types.xhtml].

This Appendix content will be used to seek final media type registration for XLIFF Version 2.0 and higher Versions (including this XLIFF Version 2.2).

Registration Template

- *Type name:* application
- *Subtype name:* xliff+xml
- *Required parameters:* N/A
- *Optional parameters:* N/A
- *Encoding considerations:*

  Same as encoding considerations of application/xml as specified in [RFC 7303]

- *Security considerations:*

  Apart from all of the security considerations described in [RFC 7303], XLIFF Version 2.0 and higher has the following Security considerations:

  Extensibility: XLIFF permits extensions. Hence it is possible that application xliff+xml may describe content that has security implications beyond those described here.

  Direct external reference mechanisms: An XLIFF document has a number of attributes of the type URI or IRI, all of which may be dereferenced and some of them should be dereferenced. Therefore, the security issues of [RFC 3987] Section 8 should be considered. In addition, the contents of resources identified by `file:` URIs can in some cases be accessed, processed and returned as results.

  More details can be found in the Detailed Security Considerations section of this Appendix.

- *Interoperability considerations:*

  Same as interoperability considerations described in [RFC 7303]

  Also, interoperability requirements are specified throughout the specification and summarized in its Conformance Section.

- *Published specification:*

  XLIFF Version 2.0 (OASIS Standard) http://docs.oasis-open.org/xliff/xliff-core/v2.0/os/xliff-core-v2.0-os.html will be superseded by XLIFF Version 2.2 (OASIS Standard) http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html that was last published on 18 April 2021 in the Specification Draft (csprd) stage at http://docs.oasis-open.org/xliff/xliff-core/v2.2/xliff-core-v2.2.html and is backwards compatible with XLIFF Version 2.0.

- *Applications that use this media type:*

XLIFF conformant applications, according to the Conformance Section of this specification

- *Fragment identifier considerations:*

  Generic XML processors won't be able to resolve XLIFF fragment identifiers, as the fragment identification syntax is specific for XLIFF and has been defined in its Fragment Identification section as of csd03/csprd03 of XLIFF Version 2.0.

- *Intended usage:* COMMON
- *Restrictions on usage:* N/A
- *Author:*

  OASIS XML Localisation Interchange File Format (XLIFF) TC Editors: Tom Comerford, <tom@supratext.com [mailto:tom@supratext.com]>; David Filip, <david.filip@adaptcentre.ie [mailto:david.filip@adaptcentre.ie]>; Yves Savourel, <ysavourel@enlaso.com [mailto:ysavourel@enlaso.com]>

- *Change controller:*

  OASIS XML Localisation Interchange File Format (XLIFF) TC https://www.oasis-open.org/committees/xliff/

  Bryan Schnabel, <bryan.s.schnabel@tektronix.com [mailto:bryan.s.schnabel@tektronix.com]>, Chair

  Tom Comerford, <tom@supratext.com [mailto:tom@supratext.com]>, Secretary

  David Filip, <david.filip@adaptcentre.ie [mailto:david.filip@adaptcentre.ie]>, Secretary

- *Provisional registration? (standards tree only):* NO
- *Additional information:*
  - *Deprecated alias names for this type:* N/A
  - *Magic number(s):* N/A
  - *File extension(s):* xlf
  - *Macintosh file type code(s):* "TEXT"
- *Person & email address to contact for further information:*

  OASIS Technical Committee administration <tc-admin@oasis-open.org [mailto:tc-admin@oasis-open.org]>

Detailed Security Considerations

Privacy, trust and integrity

XLIFF is a format for localization and translation, privacy, trust and integrity requirements will widely depend on the type of content that is being exchanged translating end user manuals for a dishwasher will have lower privacy requirements than translating clinical tests results for a pharma company.

The XLIFF format does not offer any internal mechanisms to provide privacy, convey trust or verify the integrity of XLIFF documents. If such features are needed varies from case to case. Implementations that will process documents in cases where one or more of these features are required need to implement that outside of the XLIFF format. Transport privacy may for example be provided by SSL/TLS. Storage privacy could be implemented by encrypting the XLIFF content using XML encryption or some other appropriate means. Likewise the trust and integrity checks could be implemented using XML signatures or by some other technology that is appropriate for the particular implementation.

Core

`<skeleton>` via attribute `href`

There is no requirement that an implementation dereference and load the skeleton. But it must be assumed that some do. An implementation is free to provide any type of resource as the skeleton including executables.

`<mrk>` via attribute `ref` for Term Annotations and some custom annotations

For term annotations there may be a risk by downloading or directing the user to access an external resource. For custom annotations the same applies but an implementation is not required to process the `ref` attribute on custom annotations but it must be expected that some will. Especially the term annotation one may be an issue as a reasonable implementation may just launch the URI expecting a web browser or viewer application to handle it.

Resource Data Module

`<res:source>` via attribute `href`

`<res:target>` via attribute `href`

Both of these may reference executable or otherwise unsafe external data. Either as a resource that need processing or to present additional information to the user from a resource of arbitrary type. Essentially the same considerations as for the term annotation in core applies here especially for reference material. The intent is to present arbitrary typed data to the user.

ITS Module

As the ITS Module brings a large number of ITS features natively to XLIFF, Security considerations of `application/its+xml`, as described in [ITS] https://www.w3.org/TR/its20/#its-mime-type should be taken into consideration, albeit largely overlapping with XLIFF general Security considerations described above.

Other potentially security sensitive constructs

Extension by arbitrary XML on `<file>`, `<group>` and `<unit>`

Allows embedding of arbitrary XML structures at these points.

Extension by custom attributes on `<xliff>`, `<file>`, `<group>`, `<unit>`,`<note>`,`<mrk>` and `<sm>`

Custom attribute extension is likely not as sensitive as embedding of arbitrary XML structures and will not in itself pose any threat except potentially for the implementers of the extension.

Format Style Module

Uses HTML element names as values of the attribute `fs`

Validating allowed element names may decrease risk, but due to the attribute `subFs` cannot eliminate it. Attribute `subFs` allows arbitrary additional attributes for injection into HTML elements defined in the `fs` attributes. This could be used to inject active content such as JavaScript into the preview HTML document or reference external resources. Implementations need to take normal precautions when rendering, as if rendering an arbitrary page on the web unless it can know for sure it can trust the document. XLIFF itself does not provide a facility to communicate trust or protect a document from modification. If such features are needed they must be implemented external the XLIFF format.

Actual consumable HTML is only produced by implementers of this modules via XSLT or similar.

B. Machine Readable Validation Artifacts

This appendix summarizes information on machine readable validation artifacts for XLIFF Version 2.2

1. *XLIFF Core* [XML Schema],

   http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/xliff_core_2.0.xsd

2. [XML Catalog] of *XLIFF Defined* XML Schemas,

   http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/catalog.xml

3. Master [NVDL] file governing validation of all *XLIFF Defined* namespaces by XML Schemas, Schematron Schemas and other rules if and as required,

   http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/xliff_2_advanced_validation.nvdl

4. *XLIFF Core* [Schematron] Schema,

   http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/xliff_core_2..sch

5. XML and Schematron Schemas of *XLIFF Modules* are referenced from those modules.

The basic grammar and structure of XLIFF 2.2 is defined using ten (10) XML Schemas and one (1) XML catalog. The module schemas are specifically referenced from their respective modules.

Advanced static *Constraints* and dynamic *Processing Requirements* that could not be expressed using [XML Schema] 1.0 are expressed in nine (9) Schematron schemas.

Relationships among all of the above mentioned machine readable validation artifacts provided as part of this multipart product is expressed using one (1) NVDL schema.

Warning

Albeit the TC has made effort to cover the prose specification with standardized declarative validation artifacts to the maximum possible extent, there are some inherent limitations to the Schema languages employed to perform the validation. The informative Test Suite [http://tools.oasis-open.org/version-control/browse/wsvn/xliff/trunk/xliff-21/test-suite/#_trunk_xliff-21_test-suite_]

provided through the XLIFF TC SVN does contain a number of invalid files that cannot be caught using only the normative validation artifacts that are distributed as part of this multipart Standard product. For instance [BCP47] compliance of `srcLang`, `trgLang`, or `xml:lang` cannot be fully validated by either W3C XML Schema or Schematron. Custom code is required to check this.

Warning

NVDL is not capable of discerning Schemtaron Warnings from Schematron Errors. Therefore all Schematron Warnings will be reported as Errors when initiating the validation from the NVDL schema. Also most of the existing Schematron implementations are not capable of discerning Warnings from Errors, thus implementers are encouraged to re-use the provided Schematron schemas in custom made validation services that can make this distinction. Currently, the Warning/Error distinction is only important when evaluating adherence to Processing Requirements for Editing Hints in relation to segment `state`. It will be also beneficial for implementers who want to add project specific rules based on the Validation Module.

XML Schemas Tree

```
Master NVDL Schema [http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schem
   |      |
   |    Core XML Schema [http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/
   |      |
   |      +---Candidates Module XML Schema [http://docs.oasis-open.org/xliff/xli
   |      |
   |      +---Glossary Module XML Schema [http://docs.oasis-open.org/xliff/xlif
   |      |
   |      +---Format Style Module XML Schema [http://docs.oasis-open.org/xliff/x
   |      |
   |      +---Metadata Module XML Schema [http://docs.oasis-open.org/xliff/xlif
   |      |
   |      +---Resource Data Module XML Schema [http://docs.oasis-open.org/xliff/
   |
   |      |
   |      +---Size and Length Restriction Module XML Schema [http://docs.oasis-o
   |      |
   |      +---Validation Module XML Schema [http://docs.oasis-open.org/xliff/xli
   |      |
   |      +---ITS Module XML Schema (W3C namespace subset) [http://docs.oasis-op
   |      |
   |      +---ITS Module XML Schema (additional attributes) [http://docs.oasis-o
   |
   +---Core constraints [http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/
          |
          +---Candidates Module Constraints [http://docs.oasis-open.org/xliff/xli
          |
          +---Glossary Module Constraints [http://docs.oasis-open.org/xliff/xlif
          |
          +---Format Style Module Constraints [http://docs.oasis-open.org/xliff/x
          |
          +---Resource Data Module Constraints [http://docs.oasis-open.org/xliff/
          |
          +---Size and Length Restriction Module Constraints [http://docs.oasis-o
```

```
|
+---Metadata Module Constraints [http://docs.oasis-open.org/xliff/xlif
|
+---Validation Module Constraints [http://docs.oasis-open.org/xliff/xl
|
+---ITS Module Constraints [http://docs.oasis-open.org/xliff/xliff-cor
```

Support Schemas

Third party support schemas that are normatively referenced from this specification or from the machine readable artifacts that are a part of this multipart product are distributed along with the *XLIFF-defined* schemas in a subfolder named `informative-CopiesOf3rdPartySchemas` and further subdivided in folders according to the owner/maintainer of the schema.

Warning

Schema copies in this sub-folder are provided solely for implementers convenience and are NOT a part of the OASIS multipart product. These schemas belong to their respective owners and their use is governed by their owners' respective IPR policies. The support schemas are organized in folders per owner/maintainer. It is the implementer's sole responsibility to ensure that their local copies of all schemas are the appropriate up to date versions.

Currently the only included third party support schema is http://www.w3.org/2001/xml.xsd [http://www.w3.org/2009/01/xml.xsd] at http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/informativeCopiesOf3rdPartySchemas/w3c/xml.xsd in this distribution.

C. Specification Change Tracking

High Level Summary of Changes made in Comparison to XLIFF Version 2.0

This is to facilitate human tracking of changes between XLIFF Versions 2.2 and 2.0.

1. Two major features are being added in XLIFF Version 2.2:
   a. Advanced Validation methods
   b. Native Support for ITS 2.0
2. The Change Tracking Module was demoted to an extension to free hands of the TC and other implementers while working on a new version of the Change Tracking Module for XLIFF 2.2.
3. A major bug fix was performed on the core xsd. The core xsd now enforces the `xs:language` data type on the `srcLang` and `trgLang` attributes. It was critical to make this fix, because -- as per OASIS policy -- validation artifacts would prevail over the prose provisions that are correct in both XLIFF 2.2 and XLIFF 2.0.
4. Also an erroneously omitted Constraint of the `xml:lang` attribute on the `<source>` element has been added/restored in the normative text.

5. Apart from the five (5) major changes mentioned above, numerous editorial bugfixes were made to secure greater clarity, either by fixing example errors or omissions, or by reorganizing normative content, so that the intent becomes clear and unequivocal at some troublesome places highlighted by XLIFF 2.0 implementers.

6. Importantly, the TC decided to drop informative listings of the validation artifacts that had bloated the spec extent unnecessarily, were hard to keep in sync with the actual normative artifacts, while their actual usability proved rather limited -- readers who were able to read schema languages would not actually read them as printed listings and would anyways refer to the actual validation artifacts that are now referenced more prominently.

In spite of the above mentioned changes, fixes, clarifications, and additions, the practical workings of the *XLIFF Core* hasn't been affected and none of the changes (except the bugfixes under 3 and 4) have affected the core namespace `"urn:oasis:names:tc:xliff:document:2.0"` or the *XLIFF Core* [XML Schema], http://docs.oasis-open.org/xliff/xliff-core/v2.2/csprd/schemas/xliff_core_2.0.xsd that expresses its basic grammar and structure.

Tracking of changes made in response to Public Reviews

This is to facilitate human tracking of changes in the specification made since the first Public Review publication on 26th October 2016.

Tracking of changes in response to the Public Review of the Candidate OASIS Standard 01

This section tracks all changes made to this specification compared to the Candidate OASIS Standard 01 http://docs.oasis-open.org/xliff/xliff-core/v2.1/cos01/xliff-core-v2.1-cos01.html. This Public Review took place from 20th October 2017 until 19th December 2017.

1. An important typo in several occurrences of the W3C ITS namespace has been fixed in response to Comment/Issue https://issues.oasis-open.org/browse/XLIFF-73. The typo did confuse implementers although the correct namespace had been used throughout the validation artifacts.

2. A minor editorial improvement was made in response to Comment/Issue https://issues.oasis-open.org/browse/XLIFF-72.

Tracking of changes in response to the 4th Public Review

This section tracks all changes made to this specification compared to the Committee Specification Draft 04 / Public Review Draft 04 http://docs.oasis-open.org/xliff/xliff-core/v2.1/csprd04/xliff-core-v2.1-csprd04.html. This subsequent Public Review took place from 5th June 2017 until 20th June 2017.

1. Xpath expressions have been fixed in ITS Rules in response to Comment/Issue https://issues.oasis-open.org/browse/XLIFF-58.

2. Minor editorial fixes and improvements were made in response to Comments/Issues https://issues.oasis-open.org/browse/XLIFF-57, https://issues.oasis-open.org/browse/XLIFF-62, https://issues.oasis-open.org/browse/XLIFF-63, https://issues.oasis-open.org/browse/XLIFF-64, https://issues.oasis-open.org/browse/XLIFF-65,

https://issues.oasis-open.org/browse/XLIFF-66, and https://issues.oasis-open.org/browse/XLIFF-70.

3. Trivial editorial fixes and improvements were made in response to Comments/Issues https://issues.oasis-open.org/browse/XLIFF-59, https://issues.oasis-open.org/browse/XLIFF-60, and https://issues.oasis-open.org/browse/XLIFF-61.

Tracking of changes in response to the 3rd Public Review

This section tracks all changes made to this specification compared to the Committee Specification Draft 03 / Public Review Draft 03 http://docs.oasis-open.org/xliff/xliff-core/v2.1/csprd03/xliff-core-v2.1-csprd03.html. This subsequent Public Review took place from 17th April 2017 until 1st May 2017.

1. Major bug fix of the core Schematron Schema has been made in response to Comment/Issue https://issues.oasis-open.org/browse/XLIFF-48. The core Schematron now enforces that non-reorderable sequences start with a code with `canReorder` set to `firstNo` and also enforces the repetition of non-reorderable sequences in `<target>` elements. In connection with this issue, issues https://issues.oasis-open.org/browse/XLIFF-10 and https://issues.oasis-open.org/browse/XLIFF-11 were reopened and changes in core Schematron made to ensure that the rules for enforcing of editing hints compliance in `target` elements worked properly in concert with reporting of invalid `<segment>` `state`.

2. Another major bug fix was due reopened https://issues.oasis-open.org/browse/XLIFF-38, validation methods had to be adjusted for core and core reused in modules in the NVDL.

3. An erroneously omitted Constraint of the `xml:lang` attribute on the `<source>` element has be added/restored in the normative text and check therefore introduced in the Core Schematron Schema in response to Issue/Comment https://issues.oasis-open.org/browse/XLIFF-55.

4. `its:version` attribute was introduced in response to Issue/Comment https://issues.oasis-open.org/browse/XLIFF-54.

5. Due to reopening of https://issues.oasis-open.org/browse/XLIFF-8 [https://issues.oasis-open.org/browse/XLIFF-38], mapping of `locQualityRatingScore` was removed from `matchQuality`. This eased the implementation of both ITS MT Confidence end Localization Quality Rating considerably.

6. Changes were made to validation of `annotatorsRef` attribute in response to Issue/Comment https://issues.oasis-open.org/browse/XLIFF-52. Examples using `annotatorsRef` had be reformatted not to suggest a wrong interpretation of the attribute.

7. Editorial fixes were made in response to Comments/Issues: https://issues.oasis-open.org/browse/XLIFF-47, https://issues.oasis-open.org/browse/XLIFF-50, https://issues.oasis-open.org/browse/XLIFF-51, https://issues.oasis-open.org/browse/XLIFF-53 and https://issues.oasis-open.org/browse/XLIFF-56.

Tracking of changes in response to the 2nd Public Review

This section tracks major changes made to this specification compared to the Committee Specification Draft 02 / Public Review Draft 02 http://docs.oasis-open.org/xliff/xliff-core/v2.1/csprd02/xliff-core-v2.1-csprd02.html. This subsequent Public Review took place from 10th February 2017 until 24th February 2017.

1. The Change Tracking Module has been demoted to an Extension and the update of the Change Tracking Module has been postponed for XLIFF 2.2 in response to the Comment/Issue https://issues.oasis-open.org/browse/XLIFF-32. This radical move obsoleted the dependent Comments/Issues: https://issues.oasis-open.org/browse/XLIFF-21 [https://issues.oasis-open.org/browse/XLIFF-22] https://issues.oasis-open.org/browse/XLIFF-22, https://issues.oasis-open.org/browse/XLIFF-30 [https://issues.oasis-open.org/browse/XLIFF-22], and https://issues.oasis-open.org/browse/XLIFF-44.

2. Major bugfix of the core XML Schema has been made in response to Comment/Issue https://issues.oasis-open.org/browse/XLIFF-46. The core xsd now enforces the `xs:language` type on the `srcLang` and `trgLang` attributes.

3. Major fix to the NVDL Schema has been made in response to Comment/Issue https://issues.oasis-open.org/browse/XLIFF-38.

4. Erroneous namespace, data type, and/or rules provisions have been fixed in the ITS Module prose and validation artifacts in response to Comments/Issues: https://issues.oasis-open.org/browse/XLIFF-33, https://issues.oasis-open.org/browse/XLIFF-34, https://issues.oasis-open.org/browse/XLIFF-35 [https://issues.oasis-open.org/browse/XLIFF-34] and https://issues.oasis-open.org/browse/XLIFF-45.

5. Material changes have been made to the Locale Filter data category in the ITS Module in response to Comment/Issue https://issues.oasis-open.org/browse/XLIFF-43.

6. Major editorial changes have been made in response to Comments/Issues: https://issues.oasis-open.org/browse/XLIFF-23, https://issues.oasis-open.org/browse/XLIFF-24, https://issues.oasis-open.org/browse/XLIFF-37, and https://issues.oasis-open.org/browse/XLIFF-42.

7. Minor editorial changes have been made in response to Comments/Issues: https://issues.oasis-open.org/browse/XLIFF-26, https://issues.oasis-open.org/browse/XLIFF-28, https://issues.oasis-open.org/browse/XLIFF-31, https://issues.oasis-open.org/browse/XLIFF-36, https://issues.oasis-open.org/browse/XLIFF-39, https://issues.oasis-open.org/browse/XLIFF-40, and https://issues.oasis-open.org/browse/XLIFF-41.

8. Trivial editorial changes have been made in response to Comments/Issues: https://issues.oasis-open.org/browse/XLIFF-25, https://issues.oasis-open.org/browse/XLIFF-27, and https://issues.oasis-open.org/browse/XLIFF-29.

Tracking of changes in response to the 1st Public Review

This section tracks major changes made to this specification compared to the Committee Specification Draft 01 / Public Review Draft 01

http://docs.oasis-open.org/xliff/xliff-core/v2.1/csprd01/xliff-core-v2.1-csprd01.html. The initial Public Review took place from 26th October 2016 until 25th November 2016.

1. Major changes were made in the ITS Module and validation artifacts in response to Comment/Issues https://issues.oasis-open.org/browse/XLIFF-5 and most importantly https://issues.oasis-open.org/browse/XLIFF-9 and its child issues: https://issues.oasis-open.org/browse/XLIFF-6, https://issues.oasis-open.org/browse/XLIFF-18, and https://issues.oasis-open.org/browse/XLIFF-19.

2. Major changes were made in the Change Tracking Module and validation artifacts in response to Comment/Issue https://issues.oasis-open.org/browse/XLIFF-4

3. Clarifications to Core with Advanced Validation impact, non-of which were normative changes were provided in response to Comments/Issues: https://issues.oasis-open.org/browse/XLIFF-10, https://issues.oasis-open.org/browse/XLIFF-11, https://issues.oasis-open.org/browse/XLIFF-12, https://issues.oasis-open.org/browse/XLIFF-13, https://issues.oasis-open.org/browse/XLIFF-14

4. Material clarification with Advanced Validation Impact was provided for the Translation Candidate Module in response to Issue https://issues.oasis-open.org/browse/XLIFF-20.

5. Editorial changes have been made in response to Comments/Issues: https://issues.oasis-open.org/browse/XLIFF-1, https://issues.oasis-open.org/browse/XLIFF-2, https://issues.oasis-open.org/browse/XLIFF-3, https://issues.oasis-open.org/browse/XLIFF-7, https://issues.oasis-open.org/browse/XLIFF-15, and https://issues.oasis-open.org/browse/XLIFF-17.

D. Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

- Comerford, Tom - Individual
- Estreen, Fredrik - Lionbridge
- Filip, David - TCD, ADAPT Centre (formerly Localisation Research Centre)
- King, Ryan - Microsoft
- Loomis, Steven - IBM
- Morado Vázquez, Lucía - University of Geneva
- Ritchie, Phil - Vistatec
- Soroush Saadatfar, Localisation Research Centre
- Felix Sasaki - Individual
- Savourel, Yves - ENLASO Corporation
- Schnabel, Bryan - Individual
- Tingley, Chase - Spartan Software Inc.