XLIFF 2 Extraction and Merging Best Practice, Version 1.0

Edited by David Filip and Ján Husarčík
Rodolfo M. Raya
Andreas Galambos

TAPICC T1/WG3

Additional artifacts

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

- Extraction and merging examples from https://galaglobal.github.io/TAPICC/T1/WG3/wd01/extraction_examples/

  An unstable editorial version of the examples might exist at https://galaglobal.github.io/TAPICC/T1/WG3/extraction_examples/

Related work

This note provides informative best practice for XLIFF 2 Specifications:

- XLIFF Version 2.1 [[XLIFF-2.1]]
- XLIFF Version 2.0 [[XLIFF-2.0]]
- ISO 21720:2017 [[ISO XLIFF]]

Status

This Informational Best Practice was last revised by TAPICC T1/WG3 or the TAPICC Steering Committee 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.

Contributions to this deliverable or subsequent versions of this deliverable can be made via the GALA TAPICC GitHub Repository [https://github.com/GALAglobal/TAPICC] subject to signing the TAPICC Legal Agreement [https://www.gala-global.org/tapicc-legal-agreement].

Citation format

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

[XLIFF-EM-BP]

XLIFF 2 Extraction and Merging Best Practice, Version 1.0 Edited by David Filip and Ján Husarčík. 06 June 2018. Working Draft 01. https://galaglobal.github.io/TAPICC/T1/WG3/wd01/XLIFF-EM-BP-V1.0-wd01.html. Latest version: N/A.html.

Notices

06 June 2018

Abstract

This Informational Best Practice specification targets designers of XLIFF *Extracting* and *Merging* Tools for content owners. It gathers common problems that are prone to appear when *Extracting XLIFF Documents* from HTML, generic XML, or MarkDown. This specification shows why some *Extraction* approaches will cause issues during an *XLIFF Roundtrip*. This best practice guidance provides better thought through alternatives and shows how to use many of advanced XLIFF features for lossless Localization roundtrip of HTML and XML based content.

Table of Contents

Terminology and Concepts 2
Introduction 3
Specification 3
Inline Codes 3
Target Content in Extracted XLIFF 6
Editing and Context Hints 8
XLIFF Structure 10
Miscellaneous 11
XLIFF Validations 12
Summary 12
References 12

Terminology and Concepts

Apart form terminology and concepts defined here, this specification makes heavy use of terms defined in the XLIFF Standards [[XLIFF-2.1]] such as: *Extractor*, *Merger*, *XLIFF Document*, *XLIFF defined*, etc.

context hints	*XLIFF defined* attributes on structural or inline elements providing additional contexts, such as `disp [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#disp]` or `equiv [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#equiv]`. Attributes `fs` and `subFs` defined in the XLIFF Format Style Module are also considered *context hints*.
inline codes	`<sc/>`/`<ec>` pairs, orphaned `<sc/>` or `<ec/>`, well formed `<pc>`, standalone `<ph/>` and `<cp>` are inline codes used to represent native format inline markup in *XLIFF Documents*.

markers

`<sm/>`/`<em/>` pairs and well formed `<mrk>` are *XLIFF defined* inline marker elements designed for inline annotations of content with metadata.

Note

Markers are distinct from *inline codes* (see). Markers can be combined with standoff elements for rich metadata that would be complicated or impossible to display inline.

Introduction

This Informational Best Practice targets designers of XLIFF *Extracting* and *Merging* Tools for content owners. *XLIFF Roundtrip* designers of all kinds will benefit, no matter if they design their *XLIFF Extractor/Merger* for corporate or blog use.

Extraction and *Merging* behavior is out of the normative scope of OASIS XLIFF Specifications. Although those specifications do provide some guidance for *Extractor* and *Merger Agents*, XLIFF TC did not attempt to prescribe how exactly to use XLIFF to represent native content. This is mostly because XLIFF is a native-format-agnostic Localization Interchange Format.

This specification gathers common problems that are prone to appear when Extracting XLIFF Documents from HTML, generic XML, or MarkDown. This specification shows why some *Extraction* approaches will cause issues during an *XLIFF Roundtrip*, issues often so severe that *Merging* back of target content will not be possible without costly post-processing or could fail utterly. This best practice guidance provides better thought through alternatives and shows how to use many of advanced XLIFF features for lossless Localization roundtrip of HTML and XML based content. Most of the times there are no ultimate prescribed solutions, rather possible design goals are described and best methods how to achieve them proposed.

Specification

Inline Codes

Guidance for processing standalone and spanning inline functional and formatting elements of localizable content.

- Perform complete extraction
- Represent spanning code using `<sc/>` `[http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#sc]` and `<ec/>` `[http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#ec]` (or `<pc></pc>` `[http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#pc]` where possible)
- Represent standalone code using `<ph>` `[http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#ph]`
- Include *inline codes* in *Extracted* content
- XLIFF2 prose [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html].

Representing Spanning Codes

Spanning codes in the original format are created by opening code, content and closing code. In HTML that can be `<bold>text</bold>`, in RTF `\b text \b0`.

In XLIFF, such code can be always represented with an <sc/> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#sc]/<ec/> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#ec] pair, or with spanning <pc></pc> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#pc], only for well formed markup.

Ideally, the original format is documented well enough to instruct *Extractors* about the role of each *inline code*. For example, XML Schema allows to declare elements using the keyword EMPTY. This way, all elements that are not declared EMPTY, can be represented as described above. To further help the *Extraction* process, the following recommendation could be implemented in the original XML format: “For interoperability, the empty-element tag SHOULD be used, and SHOULD only be used, for elements which are declared EMPTY.”[[XML]].

Following this recommendation of the XML specification, an empty ought to be encoded as and therefore represented as an <sc/> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#sc]/<ec/> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#ec] pair in *XLIFF Documents*, unlike the always empty
 that has to be represented as a standalone placeholder code <ph/> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#ph].

This concept is illustrated by the *bad practice* example spanning_as_ph [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/spanning_as_ph].

This kind of encoding doesn't inform the *Translating Agent* (human or machine *Modifier*) that the original code formed a span and effectively the original spanning code is not protected during the roundtrip. The standalone <ph/> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#ph] codes can be switched, one of them omitted, the span is likely to end up misplaced, malformed, or empty simply because the *Translation* editor cannot convey to the translator that the codes enclose a certain portion of the original content and what is the semantics of the original code span.

Outermost Tag Pairs

In some cases, the *inline codes* can enclose the localizable string in a way that could suggest omitting them in the *Extracted* text. For example, a paragraph containing only a link text, could be *Extracted* as the link text only, without the <a> decoration being represented. This relates to the previous *bad practice* example [https://github.com/GALAglobal/TAPICC/blob/master/extraction_examples/spanning_as_ph/bad_opening_excluded.xlf] with the spanning tag represented as two empty <ph/> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#ph] elements.

In case the <a> decoration is not represented, the translator loses valuable context (they cannot check the link), more importantly they don't know that the text is a link text, and moreover are unable to add any text outside of the link span, which might be advisable or even mandatory in certain locales.

Ideally, a consistent approach to all *inline codes* ought to be used during *Extraction*.

See the relevant *bad practice* example outermost_inline_excluded [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/outermost_inline_excluded].

Incomplete Extraction of Inline Codes

Some implementers choose not to *Extract inline codes* at all and use instead one of the following approaches:

- CDATA sections as content of <source> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#source] and <target> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-

`core-v2.1.html#target]` elements (cdata [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/cdata])

- Escaping of native codes using XML entities (inline_codes_plain_text [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/inline_codes_plain_text])

Doing one of the above can be used as a useful interim *Extraction* step when producing *XLIFF Documents* that are fit for roundtrip. However, it is strongly discouraged to send *XLIFF Documents* with the inline content not fully parsed for Localization roundtrip.

Such incomplete *Extraction* leaves *inline codes* unprotected and increases the risk of their corruption during the roundtrip, simply pushing the problem of *inline code* handling downstream.

According to the XLIFF [[XLIFF-2.1]] Standards, *Modifiers* can perform secondary parsing:

"Writers *may* preserve original CDATA sections [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#d0e8112]" (meaning that it is entirely optional to preserve CDATA sections and that *XLIFF Writers* are not obliged to preserve CDATA sections) and

text can be converted into inline codes [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#d0e8993].

Mergers have to accept XLIFF files with valid modifications [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#ApplicationConformance], even though they may ignore the added codes [addingcodeshttp://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#addingcodes].

Finally, it is considered an XML internationalization best practice [https://www.w3.org/TR/xml-i18n-bp/#AuthCDATA] to avoid CDATA sections in localizable content. This best practice is of course also valid for XLIFF `<source> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#source]` elements.

Strictly speaking, it is not illegal to create *XLIFF Documents* that contain CDATA sections or unparsed entities instead of fully parsed XLIFF inline content. However, considering all of the above, it is clear that unparsed inline content makes XLIFF Documents unfit for a fully interoperable roundtrip. Again, strictly speaking, *XLIFF Documents* with unparsed inline content are *capable* of roundtrip but all the effort that is saved on *Extraction* will cause unpredictable issues and hence even more effort when *Merging* back.

Implementers need to consider that *XLIFF Documents* with unparsed inline content are very likely to return with critical inline syntax or formatting corruptions that cannot be prevented on CDATA sections or entities that are both opaque to *XLIFF Modifiers*. Such corruptions are likely to prevent proper functionality of target content in the native environment. In case *XLIFF Modifiers* do perform the secondary parsing of content unparsed on *Extraction*, which is allowed by the standard, corruption will be prevented, however, *Mergers* will need to perform *unparsing* to facilitate merging back into the native environment, because XLIFF Modifiers are not and cannot be obliged to *unparse* back to CDATA sections or entities not knowing the *Extraction* and *Merging* logic of the *XLIFF Document* originator.

Representing Multiple Subsequent Codes

As original *inline codes* can occur in clusters, for instance as nested formatting, implementers could be tempted to combine such markup on *Extraction* and represent it as a single inline element.

This kind of *Extraction* is likely to prevent potentially desirable *Modification* of *inline codes*, affecting *Translation* quality. It will also prevent usage of fine grained code metadata (for instance context, display, and editing hints) or automated format validation during the roundtrip.

On the other hand, some potential benefit can be perceived in reducing markup inside segment content, which is useful in CAT tools that cannot properly display the inline codes (reneder information avail-

able through original data or context hints). In such tolls, less markup reduces the visual clutter and makes the translatable text more readable. This can be solved by proper choice of CAT tools (short term) or by large buyers requesting that offending tool vendors do support proper rendering of *inline code* data and metadata (mid and long term).

Implementers need to consider the pros and cons of both approaches and use the one that best matches their business need.

For examples see multiple_codes_represented_as_single [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/multiple_codes_represented_as_single].

Target Content in Extracted XLIFF

This section focuses on reasons whether or not to populate the `<target>` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#target] element during *Extraction* or *Enriching* and when to do so, if at all.

Generally, one should omit the `<target>` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#target] element, unless there is an added value and also in cases where the specification offers another dedicated solution. Proper support of the state machine during the whole roundtrip helps *Agents* to process and validate the *XLIFF Documents* as intended.

When looking at the situation from the *microservices* point of view, the *Extractor*/*Merger* ought to be implemented as just that — a single purpose *Extraction*/*Merging* service that delegates any other operations, such as segmentation or *Enriching* to other specialized services.

Output of such extractor would be a *target language* agnostic *XLIFF Document* with source content only, possibly with additional modules/extensions which could not be generated after extraction, for example Size and Length Restriction Module [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#size_restriction_module] or Format Style Module [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#fs-mod].

Unless the implementer has a specific need to create *target language* specific instances of the extracted *XLIFF Document*, for instance by *Enriching* with translation candidates, the *Extracted XLIFF Document* could and ought to be sent downstream for the Localization roundtrip as-is.

Inserting Source Content into `<target>`

The copy of the source content in `<target>` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#target] elements generally does not provide any advantage during the XLIFF roundtrip. On the contrary, it brings disadvantages such as needlessly increasing the size of the *XLIFF Document* or enforcing existence of the `trgLang` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#trgLang] attribute with a specific BCP 47 compliant value. Populated `<target>` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#target] elements are also likely to prevent segmentation modification, unless the target content is intentionally removed (which service providers are understandably hesitant to do). Not the least issue is that the source content copied to the target actually is *not* in the target language indicated by the BCP 47 tag on the XLIFF root element, which can cause a host of other processing issues. The *bad practice* of populating `<target>` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#target] elements with source content used to facilitate parsing and editing of XLIFF in *Translation* editors or generic XML editors that didn't have XLIFF support and could only open for translation certain elements in generic XML formats. As such, this practice is strongly discouraged.

Bad practice example: source_in_target [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/source_in_target].

Inserting Possible Translations into <target> elements

Enriching Agents can use translation memories, machine translation engines, or other means to obtain suitable translation candidate strings in the target language to be used later in the roundtrip, for example as suggestions for translators, to achieve better leverage, or to get higher consistency with previous translations.

Using the <target> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#target] element for storing such translation candidates limits the number of the possible proposed translations to a single one per segment. Moreover, this way it's not possible to pass critical metadata about the translation candidate, such as its origin, similarity, or quality (all those are available in a dedicated module), causing interoperability issues for *Agents* without prior knowledge of the workflow.

Inserting translation candidates into <target> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#target] elements during *Extraction* or *Enriching* constitutes an illegal overload of the core element with a clearly set purpose. The Translation Candidates Module [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#candidates] was designed exactly to provide translators with multiple translation candidates along with metadata that facilitate decision making and effective reuse.

Example: pre-populated_target [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/pre-populated_target].

State Machine

The XLIFF specification contains attributes for managing a segment state machine. The attributes used are <state> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#state] and <subState> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#substate]. The <subState> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#substate] attribute can only be used as long as the <state> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#state] attribute is used. The <state> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#state] attribute is for high level interoperability. The <subState> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#substate] attribute allows implementers to define private sub-state machines that can give more fine-grained sub-states based on their private workflow needs.

The <state> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#state] attribute defines just a high level four states state engine. The values are `initial`, `translated`, `reviewed`, and `final`. Although this attribute is *optional* on the [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#segment] element, it is assumed as having the default value `initial` whenever not used explicitly. There are some important advantages to using the state machine explicitly. Importantly, <target> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#target] elements are optional in the `initial` state. So if you want to even enforce <target> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#target] existence in your deliverables you should be using at least the high level four states state engine provided by the attribute `state` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#state]. Setting the `state` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#state] attribute to `translated` or later does enforce <target> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#target] existence within segments.

Using the high level states `reviewed` and `final` gives you even more control over the progressive validation of the *XLIFF Documents* you're roundtripping. Both of the

states `translated` and `reviewed` will trigger validation of the inline data model within `<target>` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html#target] elements, which is not being validated in the `initial` state where even the existence of `<target>` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#target] is not assumed. Violations of the inline data model including Editing hints are being tested in all states more advanced than `initial`. Those violations are considered "Warnings" at `translated` and `reviewed` states. Only in the `final` state, those violations will become actual "Errors" that make the *XLIFF Document* invalid.

Editing and Context Hints

XLIFF specification provides a plethora of attributes that allow to manage the behavior and validation of structural and inline elements; such as controlling the localizability of text, protecting non-deletable inline codes, or preserving their order, controlling the segmentation modification; or providing additional context to other agents downstream.

The default values of the editing hits should be considered when creating extraction rules, to prevent issues, which cannot be identified by automated validation.

Non-deletable Inline Codes

Original source text can contain also functional inline codes, not only formatting ones, such as software placeholder replaced during runtime. Removal of these placeholders, either intentional or accidental, during translation roundtrip can produce *XLIFF document*, which cannot be merged to create target document, or create functional issues in the translated product.

XLIFF specification provides attribute `canDelete` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#candelete] with default value `yes`, which can be used on any inline code, and allow for fine grained control over importance of the inline codes, rather than setting a global rule to prevent removal of any code at all.

Preserving Order of Codes

In case the order/nesting of the inline codes in the original document is prescribed (e. g. by a schema), it has to be preserved in the target content during the localization roundtrip to prevent merge issues, or validation fails after the merging.

Using attribute `canReorder` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#canreorder] (default value `yes`) on the inline code determines, whether the code can be moved before, or after another code. This attribute can also be used to create non-reorderable sequence of inline codes. This attribute is supported by native XLIFF validation artifacts.

Example: editing_hints_canReorder [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/editing_hints_canReorder].

Controlling Segmentation

Depending *Extractor*'s rules for mapping original document's structure into XLIFF units, items of a list; rows, or cells of a table; or items in a dialog windows might be extracted as segments of a single unit.

While it's generally desirable to be able to modify segmentation within a unit during the roundtrip, doing so in above cases will most likely prevent merging, cause build issues, or have negative impact on user experience.

Attribute `canResegment` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#canresegment] on XLIFF core structural elements can be

used with care to control *Modifier*'s behavior. It's value can be controlled by rules devised from logic based on role of structural and inline codes of native format, e.g. set to `no` for:

- lists
- tables or table rows
- UI elements

Example: mapping_to_unit [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/mapping_to_unit].

Providing Context

The actors in the roundtrip, be it humans or machines, need enough information to make appropriate decision regarding operations on inline codes, and how the codes impact the adequacy and fluency of target text, i. e. the context.

These additional metadata can be provided using *context hints* attributes: `disp` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#disp] (`dispStart` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#dispstart]/`dispEnd` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#dispend]), `equiv` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#equiv] (`equivStart` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#equivstart]/`equivEnd` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#equivend]), `type` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#type], and `subType` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#subtype].

Example: context_hints [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/context_hints].

Considerations for Using Spanning Codes

Compared to `<pc>` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#pc] pair, which can be used to represent only well-formed spanning codes within single `<segment>` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#segment], the more universal `<sc>` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#sc]/`<ec>` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#ec] can handle segmentation changes; span across units; and even represent orphaned, or overlapping codes.

The support for overlapping codes can, however, pose a problem in a situation, where `<sc>` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#sc]/`<ec>` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#ec] are used to represent multiple well-formed spanning codes within a unit and their `canOverlap` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#canoverlap] attribute is not specified, thus the default value `yes` is used. In such case the well-formedness can be corrupted during the roundtrip without possibility to validate using native XLIFF capabilities.

It's thus recommended to set `canOverlap` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#canoverlap] to `no` when representing well-formed spanning codes.

Example: editing_hints_canOverlap [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/editing_hints_canOverlap].

XLIFF Structure

Taking time to consider not only what to extract, but how to extract it, and how to structure the *XLIFF document* can significantly reduce number of issues during the roundtrip and enable usage of additional features offered by XLIFF spec.

File Structure

Native file format can contain structural elements dividing its content into parts, such as title, body, header and footer, or tables, lists and divs for markup languages; or windows, dialogs, and menus for software resources.

Representing native structural elements in XLIFF using nested `<group>` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#group] elements can be useful for providing, and correctly scoping:

- additional context (`name` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#name], `type` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#type], `subType` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#subtype], attributes from Format Style Module [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#fs-mod])
- restrictions (`canResegment`, `translate`, attributes from Size and Length Restriction Module [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#size_restriction_module])
- whitespaces handling (`xml:space` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#xml_space])
- information from modules:
  - Metadata [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#metadata_module]
  - Validation [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#validation_module]
  - ITS [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#ITS-module]

Most of the above can still be achieved without the `<group>` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#group]s at the cost of high redundancy and overloading some of the XLIFF features.

Example: group [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/group].

Role of `<unit>`

Extractor sets the XLIFF structure, which cannot be modified during the roundtrip above the <unit> level. Appropriate relationship between structures of native format and `<unit>` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#unit] can make all the difference between hindering the roundtrip and making the most of XLIFF.

Problems can be caused by both extremes: too many and not enough `<unit>` [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#unit]s.

Example: mapping_to_unit [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/mapping_to_unit].

Miscellaneous

There are many other concepts, which does not belong to a particular category with some of them listed here.

Value of attribute `id`

Implementers could be tempted to store values of resource Ids in the `id [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#id]` attribute of XLIFF structural elements. While the attribute value is restricted to *NMTOKEN* by schema, the native format may not have such restrictions. Invalid characters in `id [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#id]` would be discovered as soon as the first validation occurs, though it can still be costly to fix in the long running projects.

Instead, XLIFF's `name [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#name]` is designed to store the original identifier of the resource, without the risk of possible conflicts due to *NMTOKEN* restrictions. Alternatively, `original [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#original]` can be used on `<file> [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#file]` for the same purpose.

Example: id_and_name [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/id_and_name].

Whitespace Handling

Whitespaces can be important inside nodes like `<pre>`, containing e.g. code samples, and modifying them during roundtrip is not desirable.

They are, however, often insignificant in the text nodes of markup format, such as XML or HTML, and can be modified by, for example, reformatting and indentation (i. e. pretty-print) without affecting the layout or rendered document.

Thus one cannot indiscriminately either preserve, or normalize; as in the former case, with most TMS and CAT tools penalizing whitespace discrepancies, the leverage would be negatively affected; and in the latter, the target layout could be corrupted.

Generally, the *Extractor* itself should normalize native content where possible, not relying on other agents in the roundtrip to do that, thus ensuring consistent approach; and protect XLIFF structural elements with `xml:space [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#xml_space]` set to `preserve` where necessary.

Additional details in the XLIFF spec [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#preserve_space].

Example: xml_space_preserve [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/xml_space_preserve].

Protecting Non-localizable Content

Example: ph_and_mrk [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/ph_and_mrk].

Merging Translated Content

Example: merging [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/merging].

Selecting Language Tags

Example: language_tags [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/language_tags].

Validation of Extracted Content

Example: sanity_check [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/sanity_check].

XLIFF Validations

Since XLIFF is a roundtrip oriented format that is supposed to facilitate complex workflows bringing together best of breed specialized agents, it makes big business sense to validate. In fact successful validation on every input and output step is the critical factor for successful "blind", plug-and-play, or unsupervised interoperability. If you are designing a service architecture facilitated by XLIFF as the canonical data model, or if you are otherwise integrating many services that are consuming and outputting XLIFF, it makes sense to expose XLIFF validation as a reusable microservice that is freely available and even mandated in your ecosystem, service layer, or service bus. When your tool is supposed to receive XLIFF, check first if the XLIFF you are trying to consume is indeed valid. Also if you are outputting XLIFF that other agents or service providers are supposed to consume, do check that you are outputting valid XLIFF.

XLIFF is a format that has been blessed with multiple low level implementations, therefore you have also multiple options for XLIFF validation. Since XLIFF Version 2.1 most of the advanced validation checks that required custom code in XLIFF 2.0 can be validated using XLIFF TC provided Schematron schemas that are in fact an integral normative part of the OASIS Standard. XLIFF TC provided artifacts (xsd, sch, and NVDL) can be used for validation in any XML editor. However, if you are trying to design an automated workflow you'd not typically rely on manual or semi-automated validation in XML editors, you ought rather try and build a service that is for instance using the xslt rendering of the Schematron schemas or create a service out of the command line Lynx service that is part of the OAKAPI XLIFF Toolkit.

Summary

References

Normative references

[XML] W3C: Extensible Markup Language (XML) 1.026 November 2008 https://www.w3.org/TR/xml/

[XLIFF-2.1] Edited by David Filip, Tom Comerford, Soroush Saadatfar, Felix Sasaki, and Yves Savourel: XLIFF Version 2.113 February 2018 http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html-http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html

[XLIFF-2.0] Edited by Tom Comerford, David Filip, Rodolfo M. Raya, and Yves Savourel: XLIFF Version 2.004 August 2014 http://docs.oasis-open.org/xliff/xliff-core/v2.0/os/xliff-core-v2.0-os.htmlhttp://docs.oasis-open.org/xliff/xliff-core/v2.0/xliff-core-v2.0.html

[ISO XLIFF] Edited by Tom Comerford, David Filip, Rodolfo M. Raya, and Yves Savourel: ISO 21720:2017 - XLIFF (XML Localisation interchange file format)November 2017 https://www.iso.org/standard/71490.html