DITA 1.3 proposed feature #13010

Contents

DITA 1.3 proposed feature #13010

Provides an element that specifies the string on which its associated element should be sorted. Analogous to <index-sort-as>.

Date and version information

Include the following information:

- Completion date: June 2012
- Change history:
  - 7 Oct 2012:
    - Resolved TBD for affected topics
    - Corrected a typo
- Champion: Eliot Kimber
- Email discussion: *https://lists.oasis-open.org/archives/dita/200903/msg00060.html*

Original requirement

From Su-Laine Yeo, 23 March 2009:

> A requirement that comes to our attention from time to time is for glossary elements to include a <glossary-sort-as> element, equivalent to the <index-sort-as> element. It appears to be essential for making a glossary sort properly in Japanese; one thing I've heard from multiple organizations using DITA is that no existing tool does correct automatic sorting of Japanese glossary and index entries.

Subsequent discussion on the TC list led to agreement that the solution must be more general than just glossary entries, as there is a general requirement for authors to be able to control sorting of any element that might be sorted.

Use cases

The glossary use case is the most obvious: when glossary lists are automatically generated, it is often necessary to specify an alternative sort key for the term. This is especially true in ideographic languages such as Japanese and Simplified Chinese, where the collation order of ideographs cannot be determined from the characters themselves in most cases. In Japanese, a given sequence of ideographs may have different pronunciations depending on the meaning intended by the author. In Simplified Chinese, characters are sorted by their Pin-Yin transliteration. The transliteration can only be known reliably through dictionary lookup. Thus, in both cases it is often necessary to provide the pronunciation or transliteration of the ideographs in order to ensure correct sort order.

Of course, Japanese and Simplified Chinese are only the most extreme examples. Even in English there may be a need to specify sort keys, such as having numeric terms sort by the spelling of the number or visa-versa.

Other elements may be sorted, including topic specializations that have a sortable semantic, items in definition lists, or whatever. In short, any DITA element might need to be sorted in some presentation, so it is impossible to say with certainty that any given element type would never be sorted.

Benefits

Address the following questions:

- This feature benefits users that need to reliably sort content where the content cannot be reliably sorted based only on the term value in all cases. This include all users generating glossaries.

- Users will be able to automate sorting reliably, where such automation is not today possible without the use of custom solutions.
- It is difficult to quantify the number of potential users, but glossary sorting, in particular, seems like a common requirement.
- This feature will have a significant impact for users who need it, as they will be able to produce their content in way that is not today possible.

Costs

Outline the impact (time and effort) of the feature on the following groups:

- Maintainers of the DTDs and XSDs:
  - Adds one new element type, <sort-as>
- Editors of the DITA specification:
  - One new reference topic required.
  - At least one topic in the architectural discussion of translation and localization to discuss the <index-sort-as> and <sort-as> elements and sorting and collation in general. Should mention the fact that for many languages collation and grouping details are implementation dependent and often depend on editorial rules or practice.
  - How many existing topics will need to be edited?

    I have identified the following topics that need to be updated in the Architectural Specification:

    - "DITA Processing" mentions <index-sort-as> and therefore also needs to mention <sort-as>
    - "Translation and localization" mentions <index-sort-as> and therefore also needs to mention <sort-as>
  - This feature does not change the core DITA architecture in any way.
- Vendors of tools:
  - DITA transformers that do collation will need to be updated to take sort keys into account when doing sorting.
  - XML editors will need to allow authoring of the new element type. The element type should not require any particular additional support in the editor.
- DITA community-at-large:
  - This feature provides a natural facility that will be familiar to anyone who has a sort control requirement. It is consistent with similar markup features in other common XML applications. It is consistent with the existing <index-sort-as> feature. It should not add to the perception of complexity.

Technical requirements

Provide a detailed description of how the solution will work. Be sure to include the following details:

Element

Define a new element type, <sort-as>, specialized from <data>, to the utilities domain, declared like so:

```
<!ENTITY % sort-as.content
  (#PCDATA |
   %text; |
   %keyword;)*
>
<!ENTITY % sort-as.attributes
  "%data-element-atts;
  "
>
<!ELEMENT sort-as
  %sort-as.content;
>
<!ATTLIST sort-as
  %sort-as.attributes;
>
```

```
<!ATTLIST sort-as   %global-atts;
  class CDATA "+ topic/data ut-d/
sort-as "    >
```

The <sort-as> element defines the string ("sort key") by which the containing element should be sorted or grouped (or the nearest ancestor to which sorting or grouping will be applied). The sort key may be specified in the @value attribute or in the content of <sort-as>.

The <text> and <keyword> elements are allowed in order to enable content referencing for specifying the sort phrase. If <keyword> is used within <sort-as>, it may get its effective value via @keyref in order to set the sort phrase value. If a <keyword> uses @keyref and would otherwise also act as a navigation link, the link aspect of the <keyword> element is ignored.

For topics, the <sort-as> element may be included directly in <title>, <searchtitle>, or <navtitle> when the different forms of title need different sort phrases or, if the sort phrase is common to all titles, within the topic's prolog.

For <glossentry>, the <sort-as> element may be included directly in <glossterm> or within <prolog> as for other topics.

For topic references, the <sort-as> element may be included directly in the <navtitle> or <title> element within <topicmeta> or as a child of <topicmeta>.

For definition list items, the <sort-as> should be included in the <dt> element.

TBD: Other special cases we need to account for?

For other elements, the <sort-as> element should be specified as a child of the element that contains the <title> element (or otherwise determines the presentation title, where there is no literal title in the DITA source).

Processing

Processors may choose to sort any elements for any reason. Common sort processing include glossary entries sorted to produce automatically-generated or automatically-sorted glossary lists, lists of parameters or reference entries in custom navigation structures, and so on.

The details of sorting and grouping are necessarily implementation specific. Processors may provide any mechanism for defining or configuring collation and grouping details. Even where <sort-as> has been specified two processors may produce different sorted and grouped results because they may use different collation and grouping rules. For example, one processor may be configured to sort English terms before non-English terms, another may be configured to sort them after.

When sorting elements, each element to be sorted must have some inherent text on which it will be sorted in the absence of a <sort-as> value. This text may be literal content in the DITA XML source or may be generated or constructed based on the semantics of the element involved (for example, it may be constructed from various attribute or metadata values). This text is the "base sort key" for the element. For elements with explicit or implicit titles, such as topics, figures, tables, and sections, the base sort key would normally be the title. Processors that do sorting should clearly document how the base sort key is determined for a given element type for which sorting is provided where the base sort key is not the title content (for elements that have titles).

When <sort-as> is specified, processors must construct the effective sort key by prepending the <sort-as> sort key to the base sort key. This ensures that two items with the same sort-as key but different base sort keys will sort in the appropriate order.

Note: For example, if the title of a topic is "24 Hour Support Hotline" and the value of the <sort-as> element is "twenty-four hour", then the effective sort key would be "twenty-four hour24 Hour Support Hotline".

Overall usability

This proposal allows users to include <sort-as> anywhere it might be necessary in order to facilitate correct sorting and grouping of content sorted for any purpose.

Examples

A glossary entry with a <sort-as> within the <glossterm>:

```
<glossentry id="glossentry_rkr_xfn_ng">
  <glossterm><sort-as value="dada"/>&#x5927;&#x5927;</glossterm>
    <glossdef>Literally "big big".</glossdef>
</glossentry>
```

A glossary entry with a <sort-as> within <prolog>:

```
<glossentry id="glossentry_rkr_xfn_ng">
  <glossterm>&#x5927;&#x5927;</glossterm>
  <prolog>
    <sort-as value="dada"/>
  </prolog>
    <glossdef>Literally "big big".</glossdef>
</glossentry>
```