1 Stage three: #345 New element for defining variable text

Add a new `<keytext>` element to the base and establish new rules for determining effective text content for variable text that is defined using keys

Champion

Kristen James Eberlein, Eberlein Consulting LLC

Tracking information: Stage 2

Event	Date	Links
Initial suggestion	03 December 2019	Minutes, 03 December 2019
Stage 1 proposal accepted	28 January 2020	Minutes, 28 January 2020 GitHub issue
Stage 2 proposal submitted to TC for early feedback	E-mail, 26 October 2020	Minutes, 27 October 2020
Stage 2 proposal submitted to reviewers	E-mail, 30 October 2020	Robert Anderson, Oracle Dawn Stevens, Comtech Services Chris Nitchie, Individual member Carsten Brennecke, SAP Frank Wegmann, Software AG
Stage 2 proposal submitted to TC	10 November 2020	Minutes, 10 November 2020
Stage 2 proposal discussed by TC	17 November 2020	Minutes, 17 November 2020
Stage 2 proposal approved by TC	01 December 2020	Minutes, 01 December 2020

Tracking information: Stage three

Event	Date	Links
Stage 3 proposal submitted to reviewers	01 December 2020	Deb Bissantz, Vasont Zoe Lawson, Casenet Eliot Kimber, Individual member
Stage 3 proposal submitted to TC	08 December 2020	
Stage 3 proposal discussed		
Stage 3 proposal approved		

Approved technical requirements

The `<keytext>` element will be added to the base vocabulary. This element will be an optional child of `<topicmeta>` in `<map>`, occurring at most once.

The rules for determining effective text content, which in DITA 1.3 were defined in *2.3.4.9 Processing key references to generate text or link text*, will change to the following:

> Processors **MUST** resolve variable text that is defined using keys by using the following sequence:
>
> 1. Effective text content is taken from the `<keytext>` element.
> 2. Effective text content is taken from the `<titlealt>` element with `@title-role` set to "linking".
> 3. Effective text content is taken from the `<titlealt>` element with `@title-role` set to "navigation".
> 4. Effective text content is taken from the `<titlealt>` element with `@title-role` set to a processor-recognized value.
> 5. Effective text content is taken from the title of the referenced document, if available.
> 6. Effective text content is determined by the processor.

The DITA 1.3 rules for resolving the content of the `<abbreviated-form>` will be moved to the "Rendering expectations" section in the DITA 2.0 element reference topic for `<abbreviated-form>`.

Note This proposal intersects with proposal #16, which establishes the creation of a new `<titlealt>` element that carries the `@title-role` attribute.

Dependencies or interrelated proposals

Proposal #16, Improvements to alternate titles

Modified grammar files

The following files must be modified:

- (DTD) `map.mod`
- (RNG) `mapMod.rng`

In the content below, the following conventions are used:

- Bold is used to indicate code to be added, for example, **addition**.
- Line-through and red text is used to indicate code to be removed, for example, ~~removal~~.
- Ellipses (…) indicate where code is snipped for brevity.

Figure 1: Changes to map.mod

```
...
<!-- ============================================================= -->
<!--                  ELEMENT NAME ENTITIES                        -->
<!-- ============================================================= -->
...
<!ENTITY % topicmeta   "topicmeta"                                >
<!ENTITY % keytext     "keytext"                                  >
...
<!--  ============================================================= -->
<!--                    ELEMENT DECLARATIONS                        -->
<!--  =============================================================  -->
...
<!--                   LONG NAME: Topic Metadata                   -->
<!ENTITY % topicmeta.content
                       "((%keytext;)?,
                         (%navtitle;)?,
                         (%linktext;)?,
                         (%searchtitle;)?,
                         (%shortdesc;)?,
                         (%author;)*,
                         (%source;)?,
                         (%publisher;)?,
                         (%copyright;)*,
                         (%critdates;)?,
                         (%permissions;)?,
                         (%metadata;)*,
                         (%audience;)*,
                         (%category;)*,
                         (%keywords;)*,
                         (%prodinfo;)*,
                         (%othermeta;)*,
                         (%resourceid;)*,
                         (%ux-window;)*,
                         (%data.elements.incl; |
                          %foreign.unknown.incl;)*)"
>
<!ENTITY % topicmeta.attributes
              "%univ-atts;"
>
<!ELEMENT  topicmeta %topicmeta.content;>
<!ATTLIST  topicmeta %topicmeta.attributes;>

<!--                    LONG NAME: Key text                            -->
<!ENTITY % keytext.content
                        "(#PCDATA |
                          %cite; |
                          %data; |
                          %keyword; |
                          %ph; |
                          %q; |
                          %term; |
                          %text; |
                          %tm;)*"
>
<!ENTITY % keytext.attributes
              "%univ-atts;"
>
<!ELEMENT  keytext %keytext.content;>
<!ATTLIST  keytext %keytext.attributes;>
...
<!-- ============================================================= -->
<!--             SPECIALIZATION ATTRIBUTE DECLARATIONS             -->
<!-- ============================================================= -->
...
<!ATTLIST  keytext        class CDATA "- map/keytext "      >
...
```

Figure 2: Changes to mapMod.rng

```
...
  <div>
    <a:documentation>ELEMENT TYPE NAME PATTERNS</a:documentation>
    ...
    <define name="keytext">
      <ref name="keytext.element"/>
    </define>
    ...
    <div>
      <a:documentation>LONG NAME: Key text</a:documentation>
      <define name="keytext.content">
        <zeroOrMore>
          <choice>
            <text/>
            <ref name="cite"/>
            <ref name="data"/>
            <ref name="keyword"/>
            <ref name="ph"/>
            <ref name="q"/>
            <ref name="term"/>
            <ref name="text"/>
            <ref name="tm"/>
          </choice>
        </zeroOrMore>
      </define>
      <define name="keytext.attributes">
        <ref name="univ-atts"/>
      </define>
      <define name="keytext.element">
        <element name="keytext" dita:longName="Key text">
          <ref name="keytext.attlist"/>
          <ref name="keytext.content"/>
        </element>
      </define>
      <define name="keytext.attlist" combine="interleave">
        <ref name="keytext.attributes"/>
      </define>
    </div>
    ...
    </div>
    <div>
      <a:documentation>LONG NAME: Topic Metadata</a:documentation>
      <define name="topicmeta.content">
        <optional>
          <ref name="keytext"/>
        </optional>
        <optional>
          <ref name="navtitle"/>
        </optional>
        <optional>
          <ref name="linktext"/>
        </optional>
        <optional>
          <ref name="searchtitle"/>
        </optional>
        <optional>
          <ref name="shortdesc"/>
        </optional>
        <zeroOrMore>
          <ref name="author"/>
        </zeroOrMore>
        <optional>
          <ref name="source"/>
        </optional>
        <optional>
          <ref name="publisher"/>
        </optional>
        <zeroOrMore>
          <ref name="copyright"/>
        </zeroOrMore>
        <optional>
          <ref name="critdates"/>
        </optional>
        <optional>
          <ref name="permissions"/>
        </optional>
        <zeroOrMore>
          <ref name="metadata"/>
        </zeroOrMore>
        <zeroOrMore>
          <ref name="audience"/>
        </zeroOrMore>
        <zeroOrMore>
          <ref name="category"/>
        </zeroOrMore>
        <zeroOrMore>
          <ref name="keywords"/>
        </zeroOrMore>
        <zeroOrMore>
          <ref name="prodinfo"/>
        </zeroOrMore>
        <zeroOrMore>
          <ref name="othermeta"/>
        </zeroOrMore>
        <zeroOrMore>
          <ref name="resourceid"/>
        </zeroOrMore>
        <zeroOrMore>
          <ref name="ux-window"/>
        </zeroOrMore>
        <zeroOrMore>
          <choice>
            <ref name="data.elements.incl"/>
            <ref name="foreign.unknown.incl"/>
          </choice>
        </zeroOrMore>
      </define>
      ...
      <define name="keytext.attlist" combine="interleave">
        <optional>
          <attribute name="class" a:defaultValue="- map/keytext "/>
        </optional>
    </define>
    ...
```

Modified terminology

None

Modified specification documentation

The following element-reference topic will be added to the specification: keytext (12)

In addition the following topics will be modified:

- 2.2.4.5 Reconciling topic and map metadata elements
- 2.3.4.9 Processing key references to generate text or link text
- 2.3.4.10.10 Example: Link modification or removal
- 2.3.4.10.2 Examples: Key definitions for variable text
- 2.3.4.10.3 Example: Scoped key definitions for variable text
- 3.1.1 Base DITA elements, A to Z
- 3.3.2.2 `<keydef>`
- 3.10.5.2.1 `<abbreviated-form>`
- B.6 Element-by-element recommendations for translators

The numeric references are to the version 1.3, errata 02 version of the DITA specification.

The following figures contain precise suggestions for changes to be made to topics. Deletions are indicated with line through and red text, for example, ~~deletion~~. Additions are indicated with underlines and green text; for example, addition.

Figure 3: 2.2.4.5 Reconciling topic and map metadata elements

Make the following changes:

- In the paragraphs before the table, replace `<linktext>` with `<keytext>`.

- Add a row to the table for the `<keytext>` element.

Figure 4: 2.3.4.9 Processing key references to generate text or link text

See Processing key references to generate text or link text (9). Because the topic was extensively edited, it does not show additions and deletions.

Figure 5: 2.3.4.10.2 Examples: Key definitions for variable text

See Examples: Key definitions for variable text (10). The topic does not show additions and deletions.

Figure 6: 2.3.4.10.3 Example: Scoped key definitions for variable text

1. Modify the first paragraph of the topic to clarify the example:

   The Acme Tractor Company produces two models of tractor: X and Y. Their product manual contains sets of instructions for each model; until now, the maintenance procedures have been different for each model. Now, the product manual needs to add instructions for changing the oil, and the procedure is identical for both model X and model Y. While most maintenance procedures are different for each model, the instructions for changing the oil are identical for both model X and model Y. The company policies call for including the specific model number in each topic, so a generic topic that could be used for both models is not permitted. Scoped keys can solve this problem.
2. Modify step one to clarify the procedure:

   ~~The authoring team references the model information in the `changing-the-oil.dita` topic by using the following mark-up:~~ The authoring team creates the new `changing-the-oil.dita`. The new topic uses the following markup to reference the product model:

   ```
   <keyword keyref="model"/>
   ```
3. In step 4, make the following replacements:

   ```
   <map>
     <!-- Model X: Maintenance procedures -->
     <topicgroup keyscope="model-x">
       <keydef keys="model">
         <topicmeta>
           ~~<linktext>X</linktext>~~
           <keytext>X</keytext>
         </topicmeta>
       </keydef>
       <topicref href="model-x-procedures.dita">
         <topicref href="model-x/replacing-a-tire.dita"/>
         <topicref href="model-x/adding-fluid.dita"/>
         <topicref href="common/changing-the-oil.dita"/>
       </topicref>
     </topicgroup>

   <!-- Model Y: Maintenance procedures -->
     <topicgroup keyscope="model-y">
       <keydef keys="model">
         <topicmeta>
           ~~<linktext>Y</linktext>~~
           <keytext>Y</keytext>
         </topicmeta>
       </keydef>
       <topicref href="model-y-procedures.dita">
         <topicref href="model-y/replacing-a-tire.dita"/>
         <topicref href="model-y/adding-fluid.dita"/>
         <topicref href="common/changing-the-oil.dita"/>
       </topicref>
   ```

```
  </topicgroup>
</map>
```

Figure 7: 2.3.4.10.10 Example: Link modification or removal

In the code block in step 4, replace `<linktext>` with `<keytext>`.

Figure 8: 3.3.2.2 <keydef>

See keydef (11). The topic does not show additions and deletions. (Note that this topic was revised extensively during the rework for alignment with LwDITA, so do not compare it with the DITA 1.2 specification.)

Figure 9: 3.10.5.2.1 <abbreviated-form>

See abbreviated-form (13). This topic does not show additions and deletions. (Confused why this is here? The `<abbreviated-form>`element has rules for how variable text is rendered.)

Migration plans for backwards incompatibilities

If implementations have defined variable text using the `<keyword>` element, they will need to migrate to using `<keytext>` or `<linktext>` (as appropriate). This can be accomplished by the following mechanisms:

- Manual updates
- (For very simple cases only) Global search-and-replace across DITA maps that include key definitions
- Scripting

Processing key references to generate text or link text

Variable text can be specified by key definitions. Processors determine the effective text by retrieving the content of elements in a specific sequence.

Empty elements

Empty elements that specify a key reference might get their effective content from the referenced key definitions. For the purpose of determining variable text, *empty elements* are defined as elements that meet the following criteria:

- Have no text content, including white space
- Have no sub-elements
- Have no attributes that would be used as text content

Key definitions with child <topicmeta> elements

When an empty element references a key definition that has a child `<topicmeta>` element, content from that `<topicmeta>` element is used to determine the effective content of the referencing element. Effective content from the key definition becomes the element content, with the following exceptions:

- For empty `<image>` elements, the effective content is used as alternate text. This is equivalent to creating an `<alt>` sub-element to hold that content.
- For empty `<link>` elements, the effective content is used as link text. This is equivalent to creating a `<linktext>` sub-element to hold that content.
- For empty `<link>` and `<xref>` elements, a key definition can provide a short description in addition to the normal effective content. If the key definition includes `<shortdesc>` inside of

<topicmeta>, the content of the <shortdesc> element also provides effective content for a <desc> sub-element.
- The <longdescref> and <longquoteref> elements are empty elements with no effective content. Key definitions do not set effective text for these elements.
- The <param> element does not have any effective content, so key definitions do not result in effective content for <param> elements.

Processing rules

Processors **MUST** resolve variable text that is defined using keys by using the following sequence:

1. Effective text content is taken from the <keytext> element.
2. Effective text content is taken from the <titlealt> element with @title-role set to "linking".
3. Effective text content is taken from the <titlealt> element with @title-role set to "navigation".
4. Effective text content is taken from the <titlealt> element with @title-role set to a processor-recognized value.
5. Effective text content is taken from the title of the referenced document, if available.
6. Effective text content is determined by the processor.

Generalization of effective content

When the effective content for a key reference element results in invalid elements, those elements **SHOULD** be generalized to produce a valid result.

For example, <keytext> in the key definition might use a domain specialization of <keyword> that is not valid in the key reference context, in which case the specialized element is generalized to <keyword>. If the generalized content is also not valid, a text equivalent is used instead. For example, <keytext> might include <ph> or a specialized <ph> in the key definition, but neither of those are valid as the effective content for a <keyword>. In that case, the text content of the <ph> is used.

Examples: Key definitions for variable text

Key definitions can store variable text, such as product names and slogans. Depending on the key definition, the key reference also might represent a hyperlink to a resource.

Variable text defined using keys

In the following example, the "product-name" key definition stores variable text. The key definition contains a child <keytext> element nested within a <topicmeta> element.

```
<map>
  <keydef keys="product-name">
    <topicmeta>
     <keytext>Thing-O-Matic</keytext>
    </topicmeta>
  </keydef>
</map>
```

A topic can reference the variable text by using @keyref on an inline element such as <keyword>, <ph>, or <text>, for example:

```
<keyword keyref="product-name"/> is a product designed to ...
```

When processed, the effective value of the `<keyword>` element is "Thing-O-Matic", and the effective sentence is "Thing-O-Matic is a product designed to …".

Variable text defined using keys and a linked resource

In the following example, the key definition contains both a reference to a resource and variable text.

```
<map>
  <keydef keys="product-name" href="thing-o-matic.dita">
    <topicmeta>
     <keytext>Thing-O-Matic</keytext>
    </topicmeta>
  </keydef>
</map>
```

When the key reference from the first example is processed, the effective sentence is "Thing-O-Matic is a product designed to …". The phrase "Thing-O-Matic" also is a link to the `thing-o-matic.dita` topic. Note that the title of the `thing-o-matic` topic is not used as link text, as the processing rules specify that the content of the `<keytext>` element takes precedence.

<keydef>

A key definition provides a simple way to define a key without making the definition itself a part of rendered content.

Usage information

The `<keydef>` element is a convenience element. It is equivalent to a `<topicref>` element with `@processing-role` set to "resource-only".

Attributes defaulted on the `<keydef>` element ensure that key definitions do not appear in tables of contents, do not add extra links, and are not rendered as topics.

Specialization hierarchy

The `<keydef>` element is specialized from `<topicref>`. It is defined in the mapgroup-domain module.

Attributes

<Content removed, so as to generate proposal without generating errors re broken attributes. This proposal DOES NOT makes changes to attributes.>

Example

The following code sample shows several different types of key definitions:

```
<map>
  <title>Possible keys for use in the DITA specification</title>
  <!-- Key definition #1-->
  <keydef keys="dita-tc" scope="external" format="html"
          href="https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dita">
    <topicmeta>
      <linktext>DITA Technical Committee</linktext>
    </topicmeta>
  </keydef>
  <!-- Key definition #2-->
  <keydef keys="addressing" href="dita-addressing.dita"/>
  <!-- Key definition #3-->
  <keydef keys="dita-version">
    <topicmeta>
```

```
      <linktext>2.0</linktext>
    </topicmeta>
  </keydef>
</map>
```

1. The first `<keydef>` element defines a key that links to a Web page. It contains link text; it also specifies the necessary `@scope` and `@format` attributes, so that authors do not need to include them when they reference this key.
2. The second `<keydef>` element defines a key for a local DITA topic about addressing in DITA; that topic is available to resolve link text.
3. The third `<keydef>` element defines a text-only key that specifies the current DITA version number.

<keytext>

A `<keytext>` element specifies variable or link text; it also specifies alternate text for images that are referenced by keys.

Processing expectations

See Processing key references to generate text or link text (9).

Comment by Kristen J Eberlein on 31 October 2020

Should this be a related link instead of a cross reference?

Attributes

The following attributes are available on this element: Universal attribute group.

Examples

The section contains examples of how the `<keytext>` element can be used.

Figure 10: Simple example

The following code sample shows how variable text can be defined using the `<keytext>` element:

```
<keydef keys="company-name">
  <topicmeta>
    <keytext translate="no">Acme Widget Company</keytext>
  </topicmeta>
</keydef>
```

Figure 11: More complex example

The following code sample shows a variable-text definition that includes highlighting elements:

```
<keydef keys="company-name">
  <topicmeta>
    <keytext translate="no">
      <i>Super</i> Widget Squared<sup>2</sup>
    </keytext>
  </topicmeta>
</keydef>
```

Figure 12: Alternate text for an image

DITA implementations often reference images using keys. In such cases, the `<keytext>` element provides the alternate text for the image. The following code sample shows the markup for the `<keytext>` element

```
<keydef keys="company-logo" href="images/logo.jpg" format="jpg">
  <topicmeta>
    <keytext>Blue Acorn logo</keytext>
  </topicmeta>
</keydef>
```

The image can be referenced by `<image keyref="company-logo"/>`. When rendered to mediums that support alternate text, the effective alternative text for the image is "Blue Acorn," as though a literal `<alt>`element had been a child of the `<image>`

Figure 13: Variable text that is conditionally processed

DITA implementations often need to conditionally process product names. The following code sample shows a `<keytext>` element that contains `<ph>` elements that are conditionally processed:

```
<keydef keys="company-name">
  <topicmeta>
    <keytext translate="no">
      <ph product="cat">Acme Widgets for Cats</ph
      <ph product="dog">Acme Widgets for Dogs</ph>
      <ph product="pig">Acme Widgets for Pigs</ph>
    </keytext>
  </topicmeta>
</keydef>
```

Figure 14: Processing logic

The following sample shows a key definition that includes several elements within the `<topicmeta>` element:

```
<keydef href="http://www.example.com" keys="company-name">
  <topicmeta>
    <keytext>Acme Tools</keytext>
    <navtitle>Acme Tools web site</navtitle>
    <linktext>Acme Tools Web Portal</linktext>
  </topicmeta>
</keydef>
```

Once processed, the effective text content of both `<ph keyref="company-name"/>` and `<xref keyref="company-name"/>` is "Acme Tools".

To set distinct text values for both the company name and the link text that is associated with the company Web site, best practices call for using two different key definitions.

<abbreviated-form>

The `<abbreviated-form>` element specifies a term that might appear in both abbreviated and expanded forms.

Usage information

The abbreviated and expanded forms of the term typically are defined in a `<glossentry>` topic. The applicable form of the term is displayed when processors render an `<abbreviated-form>` element.

Rendering expectations

The `<abbreviated-form>` element is designed to reference a `<glossentry>` topic that contains both a term and an abbreviated form of that term. The topic also might provide a surface form that differs from the base term. The full term or surface form is rendered in introductory contexts where the term might be unfamiliar to a reader. In other contexts, a processor substitutes the abbreviated form of the term. Note that the definition of introductory context is necessarily dependent on delivery format details, processor behavior, and editorial policy.

The following rules determine how processors render an `<abbreviated-form>` element that references a `<glossentry>` topic:

Location of the `<abbreviated-form>` element	Rendering logic
Introductory context	Render the contents of the `<glossSurfaceForm>` element. If a `<glossSurfaceForm>` element is empty or not present, render the contents of the `<glossterm>` element.
Non-introductory context	Render the contents of the `<glossAcronym>` element. If a `<glossAcronym>` element is empty or not present, render the contents of the `<glossterm>` element.

Specialization hierarchy

The `<abbreviated-form>` element is specialized from `<term>`. It is defined in the abbreviated-form domain module.

Attributes

The following attributes are available on this element: Universal attribute group and `@keyref`.

Example

The following code sample shows the content of a glossary entry topic that defines the "Anti-lock Braking System" term:

```
<glossentry id="abs-definition">
  <glossterm>Anti-lock Braking System</glossterm>
  <glossBody>
    <glossSurfaceForm>Anti-lock Braking System (ABS)</glossSurfaceForm>
    <glossAlt>
      <glossAcronym>ABS</glossAcronym>
    </glossAlt>
  </glossBody>
</glossentry>
```

Note that the topic contains three important elements:

<glossterm>
Specifies a general version of the term. This version is used as fallback if other versions are not specified.

<glossSurfaceForm>
Specifies the term as is appropriate for an introductory context.

<glossAcronym>
Specifies the acronym associated with the term. This version is rendered in non-introductory contexts.

Once the glossary entry topic is assigned a key, the author can reference the glossary entry topic by using the following markup:

```
<section>An <abbreviated-form keyref="abs"/> helps a
driver to stop. For this reason, many find an
<abbreviated-form keyref="abs"/> useful.
</section>
```

The first use of the `<abbreviated-form>` element will be rendered as the surface term, while the second (and following) uses of the term will be rendered as the acronym:

> An Anti-lock Braking System (ABS) helps a driver to stop. For this reason many find an ABS useful.