DITA 2.0 proposed feature #33

Remove copy-to.

Date and version information

Date that this feature proposal was completed	05 Oct 2019
Champion of the proposal	Eliot Kimber
Links to any previous versions of the proposal	N/A
Links to minutes where this proposal was discussed at stage 1 and moved to stage 2	https://lists.oasis-open.org/archives/dita/201706/msg00013.htmlhttps://lists.oasis-open.org/archives/dita/201706/msg00013.html
Reviewers for Stage 2 proposal	Chris Nitchie Robert Anderson
Links to e-mail discussion that resulted in new versions of the proposal	N/A
Link to the GitHub issue	https://github.com/oasis-tcs/dita/issues/33https://github.com/oasis-tcs/dita/issues/33

Original requirement or use case

From the minutes linked to above, Chris Nitchie is recorded as saying:

> The copy-to @ assumes certain things about the way processing is done, specifically the dita-ot way, and with key-scopes that's the wrong way. We should find some other way to address those needs and remove copy-to.

Use cases

General Requirements

The requirements to which `@copy-to` was a response include:

- The ability to unambiguously and reliably link from within DITA source to a specific use of a topic when the topic is used more than once within the same publication. Before DITA 1.2 this requirement was met, weakly, by the `@copy-to` attribute. With DITA 1.2 it can be met completely by the use of keys and references to keys in place of direct URI references to source topics.
- The ability to say, as the author of a map, that a given topic used more than once should produce a single result artifact or should produce multiple result artifacts for different uses of the topic. For example, a topic may be used in multiple chapters but the desire is for a single HTML result to which all links to the topic resolve. Conversely, there may be a desire to have each chapter-specific use of a topic result in a separate result HTML file. While keys enable unambiguous references to specific uses of topics they do not, by themselves, provide a way to indicate the deliverable intent for re-used topics. The use of key scopes and branch filtering effectively require the generation of unique results in some circumstances and may be sufficient to signal author intent. For example, a reference to a use of a topic within a specific scope where the topic is used in a different scope, seems to demand, or least strongly suggest, the need for a unique result artifact in a multi-artifact deliverable (however, just having two uses, even in different scopes is not sufficient to **require** two result artifacts).
- The ability to strongly determine the anchors used in a deliverable independent from the filenames used for the source topics. For example, having published a set of HTML files for a publication and knowing that readers have created links (e.g., bookmarks) to specific HTML files in that publication, when the publication is updated and republished the filenames of the HTML files must be preserved as much as possible, even if the source files have

changed, for example because the source was moved into a CCMS that imposes its own file naming scheme or because the source files were renamed and reorganized to reflect some new general source organization practice.
- The ability to indicate that topicheads should be treated as though they were references to title-only topics.
- Replace the shortdesc of the referenced topic with a short description provided by the referencing topicref.

All of these requirements, while valid, fall into the realm of delivery processing (except for the last one, replacement of short descriptions) and therefore are outside the scope of what the DITA specification can mandate. In particular, the relationship between DITA source files and anything in any kind of deliverable is entirely up to the processor to determine. While the use of keys to refer to specific uses of topics provides an unambiguous identifier for that use, and thus something reliable as the basis for persistent anchors in deliverables, that utility is not sufficient to then *require* that processors use those keys when generating anchors.

The requirement to enable dynamic replacement of short descriptions in referenced topics, while logical, is difficult to justify. Open Toolkit never implemented this feature so it is highly unlikely that anyone ever used it. The requirement to have use-context-specific content in a topic is met more generally by using key scopes and key-scope-specific key definitions or content references.

With the existence of keys the requirement to unambiguously identify and refer to specific uses of a given topic is satisfied. Thus the `@copy-to` attribute is no longer needed.

Proposed solution

- Remove the `@copy-to` attribute.
- The processing requirements for `@chunk` related to the presence of `@copy-to` must be removed or redefined to reflect the appropriate mechanism, if any. This should be addressed in the separate chunking rework proposal. It is likely that the language added in DITA 1.2 around the implications of `@copy-to` on topic heads and the implication for the generation of title-only topics was a Bad Idea and should simply be removed from DITA 2.0.

Benefits

Who will benefit from this feature?	• Tool vendors who no longer need to account for the effect of `@copy-to`. • Authors who no longer need to use `@copy-to` simply to achieve a processor-specific, deliverable-specific result.
What is the expected benefit?	• Simplification of the DITA specification by removing a problematic and redundant feature. • Providing, through guidance to implementors, richer and more consistent facilities for managing the anchors in deliverables generated from DITA source.
How many people probably will make use of this feature?	Not relevant (removing an existing feature).
How much of a positive impact is expected for the users who will make use of the feature?	This should be a significant positive impact for DITA users who currently depend on the use of `@copy-to` or otherwise struggle to manage the anchors in their generated deliverables.

Technical requirements

Important: This section must be complete in order for the proposal to be approved.

Remove the declaration of the `@copy-to` attribute from the following groups:

- topichead.attributes (`mapGroupDomain.rng`)
- anchorref.attributes (`mapGroupDomain.rng`)

- mapref.attributes (`mapGroupDomain.rng`)
- keydef.attributes (`mapGroupDomain.rng`)
- topicref.attributes (`mapMod.rng`)

Processing impact	The removal of `@copy-to` should not require a change to any processor. Processors that currently handle `@copy-to` can remove or disable that code if desired. Processors may need to add new features to enable appropriate anchor generation based on the use of keys or other author-provided hints (`@outputclass` values, new run time parameters, etc.).
Overall usability	Documents that currently use `@copy-to` will need to be migrated to replace `@copy-to` with the appropriate replacement, i.e., the use of unique keys for each use of a topic where `@copy-to` was previously used to distinguish different uses of the topic and for which there are direct URI references to the effective source file defined by `@copy-to`.

Backwards compatibility

DITA 2.0 is the first DITA release that is open to changes affecting backwards compatibility. To help highlight any impact, does this proposal involve any of the following?

Was this change previously announced in an earlier version of DITA?	No. The `@copy-to` attribute was not marked as "deprecated" in DITA 1.x.
Removing a document type that was shipped in DITA 1.3?	No.
Removing a domain that was shipped in DITA 1.3?	No.
Removing a domain from a document type shell was shipped in DITA 1.3?	No.
Removing or renaming an element that was shipped in DITA 1.3?	No.
Removing or renaming an attribute that was shipped in DITA 1.3?	Yes: `@copy-to`.
Changing the meaning of an element or attribute in a way that would disallow existing usage?	No.
Changing a content model by removing something that was previously allowed, or by requiring something that was not?	No.
Changing specialization ancestry?	No.
Removing or replacing a processing feature that was defined in DITA 1.3?	This change removes the ability to *directly* define the effective filename of a referenced topic. However, processors are encouraged to use `@keys` values for that where appropriate or necessary (for example, to determine the filename of HTML files resulting from referenced topics).

	May remove the current (likely unused) ability to impose short descriptions onto effective copies of topics.
Are element or attribute groups being renamed or shuffled?	No.

Migration plan

If the answer to any question in the previous section is "yes":

Might any existing documents need to be migrated?	Maps that use `@copy-to` will need to be migrated. Migration actions may include: • The `@copy-to` attributes must be removed. • If a topicref that used `@copy-to` does not already have a unique key associated with it, it will likely be necessary to assign a unique key to the topicref, especially if the topic is a target of a direct URI reference to the effective filename defined by the `@copy-to` attribute. For example, a migration tool can use the `@copy-to` value as a new or additional value for `@keys`, possibly removing any extension in the `@copy-to` value (i.e., removing ".dita" and then using the result as a new `@keys` value). • Cross references or content references that make direct URI references (`@href`, `@conref`) to the effective filenames defined by `@copy-to` attributes must be updated to address the appropriate resource, normally the unique key of the topicref. For example, a migration tool could simply use the target filename as the `@keyref` or `@conkeyref` value, assuming that the migration tool also uses the `@copy-to` value as a new value for `@keys`.
Might any existing processors or implementations need to change their expectations?	Processors that depend on or expect the use of `@copy-to`, for example to signal the generation of distinct artifacts from that use of a topic, will need to provide other ways to provide that signal, such as rules associated with the use of keys or `@outputclass` values.
Might any existing specialization or constraint modules need to be migrated?	Existing specialization or constrain modules that declare the `@copy-to` attribute will need to remove the attribute declaration.

Costs

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

Maintainers of the grammar files	Trivial.
Editors of the DITA specification	• How many new topics will be required? At least one topic to document the new processing expectations. Possibly more for an explanatory appendix.

- Which existing topics will need to be edited?

  Eight topics in the architecture spec:

  - `chunkingdetails.dita` has rules involving `@copy-to` in the discussion of rules for chunking. To the degree that these rules survive the separate chunking rework, this topic will need to be updated to remove references to `@copy-to`.
  - `chunkingexamples.dita` examples include those with `@copy-to`. They will need to be reworked as appropriate.
  - `ditamap-attributes.dita` has a definition of the `@copy-to` attribute. It will need to be removed.
  - `dtd-coding-element-types.dita` and `reconciling-topic-and-map-metadata.dita` show example attribute list declarations that includes `@copy-to`.
  - `metadata-in-maps-and-topics.dita` has a statement about maps being allowed to (MAY) override topic short descriptions if `@copy-to` is specified. Remove this language.
  - `processing-key-references-general.dita` mentions `@copy-to` under the section title "Reusing a topic in multiple key scopes". This statement needs to be revised to remove mention of `@copy-to`.
  - `reconciling-topic-and-map-metadata.dita` has an entry for `<shortdesc>` that refers to the same implication for shortdesc replacement when `@copy-to` is specified similar to the statement in `metadata-in-maps-and-topics.dita`. Remove this language.

  Five topics in the language reference (not counting topics that reflect generated attribute lists):

  - `dvrResourcePrefix.dita` and `dvrResourceSuffix.dita` use the reusable phrase "ditavalref-copyto" from `conref-file.dita`. The statement is not relevant to these elements with the removal of `@copy-to`. However, it is probably appropriate to say something about how prefix and suffix can affect anchor generation (namely, that the prefix and suffix should be used as appropriate when

constructing deliverable anchors). These topics also refer to the renaming rules for `@copy-to`.
    - `topicrefElementAttributes.dita` defines the `@copy-to` attribute.
    - `abstract.dita` refers to the potential for `@copy-to` to impose a short description.
    - `shortdesc.dita` refers to the implication for `@copy-to` on the imposition of short descriptions.

    The non-normative appendix `interoperability-considerations.dita` has a section on the implications for `@copy-to`. That section can be removed.
- Will the feature require substantial changes to the information architecture of the DITA specification? If so, what?

    No substantial change.
- If there is new terminology, is it likely to conflict with any usage of those terms in the existing specification?

    No new terminology.

Vendors of tools

Tool vendors will need to adjust their processors to not depend on the use of `@copy-to` and, if necessary, provide additional features that give users the appropriate control over deliverable anchors.

DITA community-at-large

- Will this feature add to the perception that DITA is becoming too complex?

    Since we are removing a confusing attribute, it should reduce the perceived complexity.
- Will it be simple for end users to understand?

    Hard to say as the implications of reuse are always challenging and this change exposes some inherent challenges around managing references to and anchors for reused content. The challenges have always been present (they are inherent in any system that provides DITA's level of reuse) but have not always been obvious.
- If the feature breaks backwards compatibility, how many documents are likely to be affected, and what is the cost of migration?

    There are probably a fairly large number of documents that use `@copy-to`. They will all need to be migrated. In the simple case the migration is a simple use of the `@copy-to` value as a `@keys` value with a corresponding change to any references to the topic. Some migration scenarios will be more involved, but in those cases it is likely that a deeper consideration of the information architecture was required in any case.

Producing migration instructions or tools	• How extensive will migration instructions be, if it is integrated into an overall 1.3 → 2.0 migration publication or white paper? Migration instructions should be fairly short, as evidenced above. They can be included in a migration whitepaper. • Will this require an independent white paper or other publication to provide migration details? No. • Do migration tools need to be created before this change can be made? If so, how complex will those tools be to create and to use? No.

Examples

A general requirement for DITA processors that produce deliverables (HTML, PDF, online help, etc.) is to provide a reliable way to map from aspects of the DITA source to "anchors" in a deliverable generated from the source, where by "anchor" is meant any uniquely-identified thing in the deliverable that can be linked to in some way. Types of anchors include HTML filenames, IDs on elements in HTML, named anchors in PDF, and help IDs. An obvious use of this is the generation of HTML for a publication: once published to the web, users may bookmark specific HTML pages or even specific HTML elements with `@id` values. If the HTML filenames or ID values change when the HTML is republished it can be very disruptive to readers who have previously bookmarked those pages. Thus the processor that produces the HTML should do its best to consistently generate result filenames and ID values. The `@copy-to` attribute was an early attempt to satisfy this requirement.

The relationship between any aspect of the DITA source and the anchors in any deliverable generated from that source is entirely processor dependent. While keys provide a good base for generating anchors (because they have precise uniqueness rules and are controlled entirely by the map author) they are only one of many possible ways of generating reliable deliverable anchors. Processors should provide ways to manage the source-to-anchor mapping.

In the following example, documents that use `@copy-to` are updated to use `@keys` instead, using the name part of the `@copy-to` filenames as the keys. This retains the topicref-specific distinctions that `@copy-to` was providing. However, it is up to processors to use the `@keys` values in some way when generating deliverables. Other ways to capture the original `@copy-to` distinction include using the `<resourceid>` element or processor-specific `<data>` or a specialization of `<data>` within the topicrefs' metadata.

Before:

```
Root map:

<map>
 <title>Reused Topics Test 01</title>
  <topicref href="reuse_with_copy_to_01.dita">
    <topicref href="topic_a.dita"/>
    <topicref href="topic_b.dita"/>
    <topicref href="topic_c.dita"/>
    <topicref href="topic_a.dita" copy-to="topic_a-use-02.dita" >
      <topicmeta>
        <navtitle>Topic A Second Use</navtitle>
      </topicmeta>
    </topicref>
    <topicref href="topic_a.dita" copy-to="topic_a-use-03.dita" >
      <topicmeta>
        <navtitle>Topic A Third Use</navtitle>
      </topicmeta>
    </topicref>
```

```
    <topicref href="topic_a.dita" copy-to="topic_a-use-04.dita" >
      <topicmeta>
        <navtitle>Topic A Fourth Use</navtitle>
      </topicmeta>
    </topicref>
  </topicref>
</map>

Topic that links to copy-to versions of topics:

<topic id="topic_b">
  <title>Topic B</title>
  <body>
    <p>Link to URI "topic_a.dita":
      <xref href="topic_a.dita"/>
    </p>
    <p>Link to URI "topic_a-use-02.dita":
      <xref href="topic_a-use-02.dita"/>
    </p>
    <p>Link to URI "topic_a-use-02.dita (no fragment ID)":
      <xref href="topic_a-use-02.dita"/>
    </p>
    <p>Link to URI "topic_a-use-03.dita":
      <xref href="topic_a-use-03.dita#topic_a"/>
    </p>
    <p>Link to URI "topic_a-use-04.dita":
      <xref href="topic_a-use-04.dita#topic_a"/>
    </p>
  </body>
</topic>
```

After (`@copy-to` replaced with keys, `@href` on `<xref>` replaced by `@keyref`):

```
Root map:

<map>
 <title>Reused Topics Test 01</title>
  <topicref href="reuse_with_copy_to_01.dita">
    <topicref href="topic_a.dita" keys="topic_a"/>
    <topicref href="topic_b.dita" keys="topic_b"/>
    <topicref href="topic_c.dita" keys="topic_c"/>
    <topicref href="topic_a.dita" keys="topic_a-use-02" >
      <topicmeta>
        <navtitle>Topic A Second Use</navtitle>
      </topicmeta>
    </topicref>
    <topicref href="topic_a.dita" keys="topic_a-use-03" >
      <topicmeta>
        <navtitle>Topic A Third Use</navtitle>
      </topicmeta>
    </topicref>
    <topicref href="topic_a.dita" keys="topic_a-use-04" >
      <topicmeta>
        <navtitle>Topic A Fourth Use</navtitle>
      </topicmeta>
    </topicref>
  </topicref>
</map>

Topic that links to specific uses of topic_a:

<topic id="topic_b">
  <title>Topic B</title>
```

```
  <body>
    <p>Link to key "topic_a":
      <xref keyref="topic_a"/>
    </p>
    <p>Link to key "topic_a-use-02":
      <xref keyref="topic_a-use-02"/>
    </p>
    <p>Link to key "topic_a-use-03":
      <xref keyref="topic_a-use-03"/>
    </p>
    <p>Link to key "topic_a-use-04":
      <xref keyref="topic_a-use-04"/>
    </p>
  </body>
</topic>
```