Stage two: #33 Remove `@copy-to`

Remove the `@copy-to` attribute.

Date and version information

Date that this feature proposal was completed	26 Oct 2020
Champion of the proposal	Eliot Kimber, Individual member
Links to any previous versions of the proposal	05 October 2019
Links to minutes where this proposal was discussed at stage 1 and moved to stage 2	30 May 2017
Reviewers for Stage 2 proposal	Chris Nitchie, Oberon Technologies Robert Anderson IBM Kris Eberlein, Eberlein Consulting
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

All of the following requirements, except for the last one, replacement of short descriptions, fall into the real of delivery processing. While valid, they 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.

The imposition of short descriptions was never implemented in DITA Open Toolkit and therefore was almost certainly never used. In any case, it is an edge case that can be satisfied through the use of scoped keys for text variables or content references (for example, authors could use conref push to replace a topic's short description).

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

create links to specific uses of topics	Without keys (DITA 1.1 and earlier), the only way to link to a topic was via direct URI reference and the only way to link to a specific use of a topic was to give that use of the topic an different effective source URI, which is what `@copy-to` does, although weakly (because it cannot, for example, distinguish between two different uses of the same topic that happen to define the same effective source URI, something you can do with scoped keys). With DITA 1.3 keys and key scopes this requirement can be met completely by the use of keys and references to keys in place of direct URI references to source topics. (While keys in DITA 1.2 satisfied this requirement for

	linking to uses of topics from within the same effective root map, it did not satisfy it for references to topics as used in other effective root maps. Keyscopes satisfy the requirement for linking to uses of topics in the context of other root maps.)
control how result artifacts are produced	For multi-part deliverables (i.e., HTML), a topic that is used multiple times can result in either multiple deliverable artifacts (multiple HTML files) or a single artifact. This choice could be made on a per-topic basis, on a per-use basis, or as a matter of deliverable style. Before the addition of DITA Open Toolkit's "ensure unique" option for HTML-based results, `@copy-to` provided a way to determine whether a given topic resulted in a single HTML result or in multiple results: if `@copy-to` was not specified then DITA Open Toolkit produced a single result HTML file to which all uses of the topic pointed. 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.
determine the anchors used in a deliverable	Most deliverables includes anchors that can serve as the targets of links to those deliverables. For many deliverables the anchors should be consistent across different versions in time of the same deliverable. This requires a way to determine what the anchors will be in a given deliverable produced from a given root map. 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 file URIs 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 `@copy-to` attribute provided one way to control the values of some anchors when the deliverable processor used the effective names of topics (as specified by `@copy-to`) to construct result anchors (i.e., the names of HTML files generated from the topics).
treat topicheads as title-only topics	For some deliverable types, such as HTML as produced by DITA Open Toolkit, the HTML result for topicheads can be different than for a reference to the equivalent title-only topic. For example, a topichead might produce only a table of contents entry but not a separate HTML result file that then has links to the child topics of the topoichead, as would be produced for a title-only topic. DITA 1.2 added a specific meaning to the use of `@copy-to` with `@chunk` on topicheads in an attempt to satisfy this requirement.

impose a short description onto the referenced topic	If a topicref that includes a short description also specifies @copy-to the rendered topic should reflect the imposed short description.

As something that is clearly processor-specific and is a weak solution at best to the requirements it was trying to satisfy, @copy-to is not an appropriate feature of the DITA standard and should be removed.

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.
- Highlight the use of the <resourceid> element as a general facility for associating deliverable-specific anchors with topics and uses of topics. This does not change the semantics of <resourceid> in any way but simply highlights its utility for this purpose. Suggest the use of the existing starter set of @deliveryTarget values as values for the @appname attribute in order to indicate the intended deliverable type for a given resource id, i.e., <resourceid appname="pdf" appid="chapter-01"/>. Also observe that normal filtering can be used with the @deliveryTarget attribute to author deliverable-specific <resourceid> elements without depending on @appname, which might be a more convenient or more interchangeable way to associate resource IDs with specific deliverable types.

  Add discussion to the <resourceid> element reference entry for this use of <resourceid> as distinct from the use of <resourceid> specifically for help delivery, emphasizing the general utility of <resourceid> for defining anchors associated with uses of topics or with topics without regard to use context.

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. This change does effectively require processors to provide new features that address the requirements previously addressed by @copy-to.

Technical requirements

This proposal involves the following changes:

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 will almost certainly need to add new features to enable appropriate anchor generation based on the use of keys, <resourceid>, 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 or <resourceid> values for each use of a topic where @copy-to was previously used to distinguish different uses of the topic and the use of keys and key references to topics 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 or

	`<resourceid>` values for that where appropriate or necessary (for example, to determine the filename of HTML files resulting from referenced topics). Removing 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). Alternatively, the topic to which the `@copy-to` applied could be literally copied in the source repository and the topicref updated to refer to the new copy. This would allow existing direct URI references to the `@copy-to` value to continue to work as they have. • 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 if `@copy-to` is replaced with `@keys` values. 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`. • Add `<resourceid>` elements to topic references or to topics in order to assign anchors to the uses of a topic (when used within topic references) or to a topic without regard to use context (when used within a topic's prolog).
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, `<resourceid>`, 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. Six 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.
    - `resourceid.dita` to add additional discussion on the use of `<resourceid>` as a general anchor-defining facility.

    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.

In particular, processors will almost certainly need to provide new features by which their deliverable processes can be configured and controlled to do the following:

- Produce deliverable anchors that are consistent for deliverables generated from different versions in time of the same root map (i.e., consistent HTML filenames in HTML-based deliverables, consistent anchors in PDF documents, etc.)
- Produce multiple or single deliverable artifacts for multiple uses of a given topic. This could be a global setting (such as DITA Open Toolkit's "ensure unique" runtime option), could be based on some convention applied to the use of key scopes, could use private metadata set on topicrefs or topics, or some processor-specific configuration facility separate from the DITA source.
- Control how multi-part deliverables handle topicheads: as navigation-only or as though they were title-only topics. Note that in monolithic deliverables

such as PDF there is normally no useful presentation distinction between topicheads and title-only topics because both should contribute to the titled hierarchy reflected in the main content flow of the deliverable.
- Implement the use of `<resourceid>` to define deliverable anchor IDs when `<resourceid>` either specifies no `@appname` value or when the processor associates the `@appname` value with a deliverable being produced.

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 or to simply copy the topics to which `@copy-to` was applied and update the topicrefs to use those new copies. 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.

  Alternatively, topics to which `@copy-to` is applied can simply be literally copied to the effective URIs defined by the `@copy-to` values. This results in content duplication but avoids any need to modify direct URI references to the topics at their `@copy-to`-defined URIs. The duplication can be resolved by using content references to factor out either common content or unique content such that resulting set of topics reflect the minimal practical duplication of content.

  Likewise, `@copy-to` values could be moved to `<resourceid>` elements where the `@copy-to` value is used as the `@appid` value and the appropriate `@appname` value is set.
- Because it will be up to deliverable-producing tools to add new features that satisfy the requirements `@copy-to` addressed, users will need to upgrade

their tools when those tools implement the features they need. However, simply moving to DITA 2.x will require upgrading tools so the effect of this need to upgrade should be part of the larger general cost of moving to DITA 2.x.

- Users that today depend on the use of `@copy-to` to meet specific deliverable requirements will need to understand how to meet those requirements using whatever `@copy-to` replacing features their tools provide. For many users this will likely involve a more sophisticated use of keys or simply a use of keys where they were not using keys before or the use of `<resourceid>` elements. For some users it may require the use of key scopes or a more sophisticated use of key scopes (for example, adding key scopes to ensure that appropriately-distinct deliverable artifacts are produced where key scopes where not previously required simply to ensure key uniqueness or to provide use-specific values for keys).

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?

  Yes. A committee note that discusses the requirements on processors to provide mechanisms for managing the mapping from source components to deliverable artifacts is needed. Such a note should do the following:

  - Outline the use cases for which people are using `@copy-to`.
  - Examine samples of DITA markup that uses `@copy-to` and suggest alternatives.
  - Be clear about the use cases that cannot be met currently.
- 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 and may not be the easiest to understand or use for this purpose. The <resourceid> element provides a direct and more-obvious way to specify deliverable anchors, including deliverable-specific anchors. For many authors, <resourceid> will be a clearer replacement for @copy-to.

For DITA content that already uses keys for all navigation topicrefs and where the key values are carefully designed and maintained, using key names as the basis for anchors is probably the easiest solution because it requires no additional work for authors.

For DITA content that does not already use keys or does not use them consistently, or where there are other requirements for anchor control that make the use of keys difficult (or at least inconvenient), <resourceid> is probably the clearest and easiest solution.

Whether using <resourceid>, keys, or some other mechanism, processors should provide ways to manage the source-to-anchor mapping. Other ways to capture the original @copy-to distinction include using processor-specific <data> or a specialization of <data> within the topicrefs' metadata. Processors could also use values of @outputclass or @base to allow authors to indicate details of how the deliverable should be delivered, such as whether or not the use of a topic should always or never result in a new deliverable artifact or whether or not a topichead should be treated as though it were a title-only topicref.

For the following example the original documents use @copy-to. There are two solutions. In the first solution, documents that use @copy-to are updated to use @keys instead, using the name part of the @copy-to filenames as the keys. In the second example, documents that use @copy-to are updated to use <resourceid> instead.

The use of keys in the first solution 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.

In addition to ensuring distinct result HTML files, the information architect's intent is also to ensure that correct "Parent topic", "Previous topic", and "Next topic" links are created in the HTML result for the "Installation Instructions" portion of the test map shown below.

```
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>
  <topichead collection-type="sequence">
    <topicmeta>
```

```
      <navtitle>Installation instructions for Windows, Linux, and macOS</
navtitle>
    </topicmeta>
    <topicref href="installing-windows.dita>
      <topicref href="install-info.dita/>
      <topicref href="installing-dbase-windows.dita/>
      ...
    </topicref>
    <topicref href="installing-linux.dita>
      <topicref href="install-info.dita copy-to="install-info-linux.dita"/>
      <topicref href="installing-dbase-linux.dita/>
      ...
    </topicref>
    <topicref href="installing-macos.dita>
      <topicref href="install-info.dita copy-to="install-info-macos.dita"/>
      <topicref href="installing-dbase-macos.dita"/>
      ...
    </topicref>
  </topichead>
</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-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>
```

Figure 1: Before: DITA 1.x source using direct URI references to topics with `@copy-to`

```
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>
  <topichead keys="install-win-linux-macos" collection-type="sequence">
    <topicmeta>
      <navtitle>Installation instructions for Windows, Linux, and macOS</
navtitle>
    </topicmeta>
    <topicref keyscope="install-win" keys="installing" href="installing-
windows.dita>
      <topicref keys="install-info" href="install-info.dita/>
      <topicref keys="installing-dbase" href="installing-dbase-windows.dita/
>
      ...
    </topicref>
    <topicref keyscope="install-linux" keys="installing" href="installing-
linux.dita>
      <topicref keys="install-info" href="install-info.dita"/>
      <topicref keys="installing-dbase" href="installing-dbase-linux.dita/>
      ...
    </topicref>
    <topicref keyscope="install-macos" keys="installing" href="installing-
macos.dita>
      <topicref keys="install-info" href="install-info.dita copy-
to="install-info-macos.dita"/>
      <topicref keys="installing-dbase" href="installing-macos-linux.dita/>
      ...
    </topicref>
  </topichead>
</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>
```

Figure 2: After: Solution 1: DITA 2.0 source with `@copy-to` replaced with keys, `@href` on `<xref>` replaced by `@keyref`, and keys and key scopes added to the installation information topicrefs

Note that for the installation instructions, the new version uses distinct key scopes for each platform's installation instructions, allowing the keys for the subordinate topics to be the same in each scope. The presence of the key scopes provides a unique name to each group of topicrefs and would enable a processor to generate unique deliverable anchors for each use of the same topic `install-info.dita`, i.e., "install-windows_install-info.html", "install-linux_install-info.html", etc. These deliverable anchors (the HTML filenames) are determined from the key names, not the filenames, which could be changed without affecting the keys. For example, even if the content was migrated to a CCMS that replaced all the original filenames with some kind of opaque object identifier, the key names would

be unchanged and a processor that used the key names to determine deliverable anchors would produce a consistent result.

The following solution uses <resourceid> to explicitly define anchors for the re-use topics, separate from any keys they may have. In this version of the solution, in addition to defining the keys, which are needed to simply enable cross references to the topics as used, the map now includes <resourceid> elements that specify @appid values, which serve to define the intended anchors for the topics in each place they are referenced (each use context of each topic).

```
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">
      <topicmeta>
        <resourceid appid="topic.A"/>
      </topicmeta>
    </topicref>
    <topicref href="topic_b.dita" keys="topic_b">
      <topicmeta>
        <resourceid appid="topic.B"/>
      </topicmeta>
    </topicref>
    <topicref href="topic_c.dita" keys="topic_c">
      <topicmeta>
        <resourceid appid="topic.C"/>
      </topicmeta>
    </topicref>
    <topicref href="topic_a.dita" keys="topic_a-use-02" >
      <topicmeta>
        <navtitle>Topic A Second Use</navtitle>
        <resourceid appid="topic.A2"/>
      </topicmeta>
    </topicref>
    <topicref href="topic_a.dita" keys="topic_a-use-03" >
      <topicmeta>
        <navtitle>Topic A Third Use</navtitle>
        <resourceid appid="topic.A3"/>
      </topicmeta>
    </topicref>
    <topicref href="topic_a.dita" keys="topic_a-use-04" >
      <topicmeta>
        <navtitle>Topic A Fourth Use</navtitle>
        <resourceid appid="topic.A4"/>
      </topicmeta>
    </topicref>
  </topicref>
  <topichead keys="install-win-linux-macos" collection-type="sequence">
    <topicmeta>
      <navtitle>Installation instructions for Windows, Linux, and macOS</
navtitle>
      <resourceid appid="installing-all-os"/>
    </topicmeta>
    <topicref keyscope="install-win" keys="installing" href="installing-
windows.dita>
      <topicmeta>
        <resourceid appid="installing-windows"/>
      </topicmeta>
      <topicref keys="install-info" href="install-info.dita>
        <topicmeta>
          <resourceid appid="install-info-windows"/>
        </topicmeta>
```

```
      </topicref>
      <topicref keys="installing-dbase" href="installing-dbase-windows.dita>
        <topicmeta>
          <resourceid appid="installing-dbase-windows"/>
        </topicmeta>
      </topicref>
    </topicref>
      ...
    </topicref>
    <topicref keyscope="install-linux" keys="installing" href="installing-
linux.dita>
      <topicmeta>
        <resourceid appid="installing-linux"/>
      </topicmeta>
      <topicref keys="install-info" href="install-info.dita">
        <topicmeta>
          <resourceid appid="install-info-linux"/>
        </topicmeta>
      </topicref>
      <topicref keys="installing-dbase" href="installing-dbase-linux.dita>
        <topicmeta>
          <resourceid appid="installing-dbase-linux"/>
        </topicmeta>
      </topicref>
      ...
    </topicref>
    <topicref keyscope="install-macos" keys="installing" href="installing-
macos.dita>
      <topicmeta>
        <resourceid appid="installing-macos"/>
      </topicmeta>
      <topicref keys="install-info" href="install-info.dita copy-
to="install-info-macos.dita">
        <topicmeta>
          <resourceid appid="install-info-macos"/>
        </topicmeta>
      </topicref>
      <topicref keys="installing-dbase" href="installing-macos-linux.dita>
        <topicmeta>
          <resourceid appid="installing-dbase-macos"/>
        </topicmeta>
      </topicref>
    </topicref>
      ...
    </topicref>
  </topichead>
</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>
```

Figure 3: After: Solution 2: Use of <resourceid> to replace @copy-to

A processor generating HTML could use the <resourceid> values as the filenames for the HTML files generated from each use of the topic. A processor generating PDF could use the <resourceid> values as the PDF named anchor names. Note that the <resourceid> @appid values do not necessarily match the key names used for the topicrefs: <resourceid> lets you completely separate the key names from anchor IDs, making the source more flexible at the cost of additional markup within the map.

In this example the <resourceid> elements only specify @appid, as that is the minimum needed to associate an anchor with a given topic or use of a topic. If there was a need to have different anchors for different deliverables, for example, to maintain consistency with previously published versions or to accommodate the anchor details of a given deliverable type, you could include @appname or use @deliveryTarget and filtering to have different anchors for different deliverables, i.e.:

```
<topicref keys="installing-dbase" href="installing-macos-linux.dita>
  <topicmeta>
    <resourceid appid="installing-dbase-macos" appname="html"/>
    <resourceid appid="install:dbase:macos" appname="pdf"/>
  </topicmeta>
</topicref>
```

or

```
<topicref keys="installing-dbase" href="installing-macos-linux.dita>
  <topicmeta>
    <resourceid appid="installing-dbase-macos" deliveryTarget="html"/>
    <resourceid appid="install:dbase:macos" deliveryTarget="pdf"/>
  </topicmeta>
</topicref>
```