DITA Technical Committee

DITA 2.0 proposals

Table of contents

1 Overview

This document contains the text of the stage two and stage three proposals for DITA 2.0 that have been approved by the OASIS DITA Technical Committee (DITA TC). This document is regenerated periodically, as new proposals are approved by the DITA TC.

2 DITA 2.0: Stage two proposals

These proposals have been approved by the DITA TC at the stage two level. For more information about stage two criteria, see DITA 2.0 proposal process.

2.1 Base edition

2.1.1 Stage two: #08 <include> element

A new base element for inclusion of content from external files. This new element will serve as the base element for the existing transclusion elements `<coderef>`, `<svgref>`, and `<mathmlref>`.

Date and version information

Date that this feature proposal was completed
February 25, 2018

Champion of the proposal
Chris Nitchie

Links to any previous versions of the proposal
None

Links to minutes where this proposal was discussed at stage 1 and moved to stage 2
https://lists.oasis-open.org/archives/dita/201607/msg00086.html

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/8

Original requirement or use case

The `<coderef>` element is a transclusion element – it loads the referenced file and places its contents as CDATA at the location of the referencing element. However, `<coderef>` is a specialization of `<xref>`, which is *not* a transclusion element. That is, the specialization-based fallback behavior for `<coderef>` is fundamentally incompatible with the expected processing. The same is true for `<svgref>` and `<mathmlref>`. The only way for people other than the TC to create similar transclusion elements via specialization is to specialize from one of those, or customize their processors to do the right thing with their element.

Use cases

- A company wants to enable the transclusion of some foreign XML markup other than SVG or MathML.

- A company wants to enable the reuse of README textual content from their source code into their DITA content.
- A tool vendor wants to provide feature support for the inclusion of CSV data as DITA tables.
- A tool vendor wants to provide feature support for the inclusion of relational database query results as DITA tables.

New terminology

None.

Proposed solution

A new base vocabulary element called `<include>`. This element will use standard DITA referencing mechanics, but will be used for transclusion rather than simple referencing. It will be the base element for `<coderef>`, `<mathmlref>`, and `<svgref>`. In addition to the standard referencing attributes, the element will carry a `@parse` attribute specifying the processing to apply to the contents of the referenced file. Standard values for this attribute will be `text`, which will imply the conversion of `<`, `>`, and `&` into their equivalent character entity references, and `xml`, which will not. Vendors may provide support for additional processing modes.

Benefits

Who will benefit from this feature?

Developers of DITA processing engines, who can now use a single, unified mechanism for processing transclusion elements, rather than special-casing the current three elements. Organizations needing to support transclusion of non-DITA content other than MathML and SVG, and code examples.

What is the expected benefit?

An extensible mechanism fr transclusion of non-DITA content.

How many people probably will make use of this feature?

Many.

How much of a positive impact is expected for the users who will make use of the feature?

Significant.

Technical requirements

New base element: `<include>` with a `@class` of "- topic/include ". In addition to `%univ-atts;` and `@outputclass`, this element will carry the following attributes:

@href

URI of content to include.

@keyref

Key reference to content to include.

@format

The format of the resource.

@scope

Describes the relationship between this document and the referenced resource. In this case, primarily used by systems such as CCMS tools to determine how or whether to manage the relationship between this document and the target resource.

@parse

Transclusion processing instructions for the processor. Conforming processors must support the following values.

text
Content is treated as plain text. Reserved XML characters are replaced with the equivalent character entity references. This is the default value.

xml
Content is parsed and inserted with XML structure preserved. If a fragment identifier is included in the address of the content, processors must select the element with the specified ID. If no fragment identifier is included, the root element of the referenced XML document is selected. Any grammar processing should be performed during resolution, such that default attribute values are explicitly populated. Prolog content must be discarded. Put another way, processing of XML inclusion should match the processing for XInclude references.

It is an error to use `parse="xml"` anywhere other than within `<foreign>` or a specialization thereof.

Processors *may* support additional values for `@parse` with custom processing expectations. Other standard tokens may be added in future versions of the DITA standard. When an unsupported value for `@parse` is encountered, processors should issue a warning and treat the `<include>` as an unresolvable reference.

@encoding

The character encoding to use when translating the character data from the referenced content when `parse="text"`. The value should be a valid encoding name. If not specified, processors may make attempts to automatically determine the correct encoding, for example using HTTP headers or through analysis of the binary structure of the referenced data, or the `<?xml?>` processing instruction (when including XML as text). The resource should be treated as UTF-8 if no other encoding information is available.

When `parse="xml"`, standard XML parsing rules apply for the detection of character encoding.

It is considered an error to use `<include>` to reference DITA content using `parse="xml"`, and processors should issue an error message when they detect inclusion of DITA content in this way. Authors should use `@conref` or `@conkeyref` instead.

The content model for the element is `ANY`. Content will not be presented in output unless the transclusion fails, in which case the contents of the tag may be used as fallback content. The presence of `translate="yes"` on an `<include>` element *does not* imply that the referenced resource should be translated, only the inline fallback content.

The `<include>` element should generally be used via specialization, and not used directly, similar to guidance on the use of the `<data>` or `<foreign>` elements.

`<include>` will be allowed anywhere `<xref>` is allowed.

The `<coderef>` element will be refactored to be a specialization of `<include>`, with the following default attribute values:

- `parse="text"`
- `format="text"`

The `<svgref>` element will be refactored to be a specialization of `<include>`, with the following default attribute values:

- `parse="xml"`
- `format="svg"`

The `<mathmlref>` element will be refactored to be a specialization of `<include>`, with the following default attribute values:

- `parse="xml"`
- `format="mathml"`

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.

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?
No.

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?
Yes, as described above.

Removing or replacing a processing feature that was defined in DITA 1.3?
No.

Are element or attribute groups being renamed or shuffled?
No.

Migration plan

Existing DITA content with `<coderef>`, `<svgref>`, or `<mathmlref>` elements will not require any migration, as their markup will not change.

DITA implementations that implement support for the `<include>` element will not need to do any extra work to support the refactored inclusion elements from DITA 1.3; correct processing for all three will happen with generic `<include>` processing.

It's unlikely that there are specializations of the existing inclusion elements, but any that do exist will need to be modified to use the new specialization hierarchy.

Costs

This change will have the following impact on the specification:

- A new language reference topic for the `<include>` element.
- Likely a new topic in the architectural spec describing the `<include>` element and its usage.
- Minor updates to the `<coderef>`, `<svgref>`, and `<mathmlref>` element topics in the language reference.

DITA processors will need to add support for the `<include>` element.

CCMS systems will need to be able to index and manage relationships between objects formed using `<include>` when `scope="local"`.

Examples

For the most part, `<include>` will be used as a basis for specialization. The following examples use it directly for purposes of illustration.

```
<!-- Inclusion of XML markup other than SVG or MathML -->
<fig>
  <title>JSP Tag Library Elements and Attributes</title>
  <foreign outputclass="tld">
    <include href="../src/main/webapp/WEB-INF/jsp-tag-library.tld"
             parse="xml" format="tld"/>
  </foreign>
</fig>

<!-- Inclusion of README text into a DITA topic. -->
<shortdesc>
  <include href="../src/README.txt" parse="text" encoding="UTF-8"/>
</shortdesc>

<!-- Vendor-specific handling for CSV tables -->
<fig>
  <title>Data Table</title>
  <include href="data.csv" parse="vnd-csv-to-simpletable"/>
</fig>

<!-- Vendor-specific handling of relationa data as tables -->
<fig>
  <title>Data Table</title>
  <include href="vnd-db://connectionName/queryName?param1=v1&param2=v2"
           scope="external"
           parse="vnd-query-to-simpletable"/>
</fig>
```

2.1.2 Stage two: #15 Relax specialization rules

Specialization rules in DITA 2.0 should be relaxed to permit more flexibility in information modeling.

Date and version information

Proposal version
1

Completion date
June 8, 2017

Proposal Champion
Chris Nitchie

Initial Suggestion
https://lists.oasis-open.org/archives/dita/201703/msg00035.html

GitHub Issue

https://github.com/oasis-tcs/dita/issues/15

Original requirement or use case

Specialization rules in DITA 2.0 should be relaxed such that they allow specialized elements to have attributes unique to that element and not present on the specialization base. Any custom attribute added to any specialized element must be declared in whatever DITA 2.0 will use instead of `@domains` for declaring specialized attributes.

It should also be possible to target a specialization of `@base` (and maybe `@props`) to a specific element or elements. For example, pre-DITA 1.3, it would have been nice to be able to add `@orient` to `<entry>` without having to also make it available on every other element in the DTD.

It should also be possible to introduce new specialized elements into existing content models selectively. For example, it should be possible to introduce a new `<data>` specialization within `<fig>` without adding it globally, all locations that `<data>` is valid. Currently, the only way to accomplish this without running afoul of specialization rules is to add the new specialization everywhere, and then constrain it from every element *except* the ones where you want it to appear. This is overly onerous, and should be simplified to allow affirmative inclusion of the element in specific content models, rather than requiring global inclusion combined with near-global exclusion via constraints.

Use cases

1. A company makes extensive use of complex tables to present product listings. They occasionally highlight individual cells, rows, or columns for various purposes. A content architect wishes to implement a semantically meaningful way to identify the `@cell-purpose` of `<entry>`, `<row>`, and `<colspec>`. They'd prefer to do this with enforceable grammar rules rather than forcing authors to hand-edit `@outputclass`, leading to potential errors. (This could be accomplished via an element domain specializing form `<table>`, but this would break their authoring tool's table editing features.) However, they want to add `@cell-purpose` *only* to table-related tags, rather than using a `@base` specializatin that would have global applicability.
2. A content architect wishes to create a new `<xref>` specialization for legal citations, and want to signify the location within referenced legislation using new attributes (they aren't in DITA, and so specifying an element ID won't work). They want the configuration attributes to appear only on their new specialization. For example:

   ```
   <legislatonref keyref="FamilyLeaveAct" legislationsubset="1.2"
               legislationsubsettype="section"/>
   ```

   These additional attributes will be used both for styling purposes and to generate indices of referenced legislation.
3. A content architect wishes to augment the content model of `<section>` with a new, optional `<section-shortdesc>` element. The `<section-shortdesc>` element would be specialized from `<p>`, but would only be allowed within `<section>`.

Proposed solution

- Specializations of `@base` can be added to attributes lists for specific elements without being added to `%base-attribute-extensions;`. Such `@base` specializations must still be represented in `@domains` and accounted for when generalizing to `@base`.

- Specialized elements can be selectively injected into specific content models without being declared as globally available anywhere its specialization base is used. Such specializations must still be accounted for in the `@domains` attribute.
- When integrated in this way, document type shells must include other `@domains` tokens indicating constraints, since the elements and/or attributes do not follow the default global availability pattern.

Note At the time of this writing, there is talk within the TC (though not yet any formal feature proposal) of radically simplifying the requirements around `@domains`. As such, much of the discussion in this proposal about `@domains` should be considered provisional pending formalization of those changes.

Benefits

- Information architects will gain significant flexibility in their designs for specializations and document type shells.
- Authors will benefit significantly from more intuitive, easier to use content models.

Technical requirements

This change is a relaxation of the rules around how DITA-complient grammars are constructed, and as such, will have essentially no technical impact.

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

DTD and Schema modifications
None.

Processing impact
None. All `@domains`- and `@class`-based processing of specialized elements and attributes will continue to function exactly as it has before.

Overall usability
Current DITA users, and current document type shells, will be unaffected. Usability of authoring with specializations will ultimately be improved due to grammars' being able to more closely model business requirements.

Costs

- This may add perceived complexity to the rules around grammar construction. However, the removal of semi-arbitrary restrictions in the current rules may compensate for this.
- Maintainability of document type shells may suffer, depending on the extent to which information architects make use of these new techniques and the care put into their initial development and organization.
- Because this proposal adds flexibility to the construction of information models, it opens the door to new opportunities for poor decision-making in information design, potentially leading to document type shells that are awkward for authors and/or processors.

Examples

See Use Cases, above.

2.1.3 Stage two #16: Add <titlealts> element to map

Provide a short synopsis of the feature proposal.

Date and version information

Proposal version
1

Completion date

Proposal champion
Kristen James Eberlein, LLC

Initial suggestion
The stage one proposal was sent to the TC list on 20 March 2017 by Chris Nitchie, Oberon Technologies. After discussion on the list, it was discussed and moved to stage two on 2 May 2017.

GitHub Issue
https://github.com/oasis-tcs/dita/issues/16

Original requirement or use case

We should allow `<titlealts>` inside map as well as topic. We should also add a zero-or-more`< titlealt>` element that can be specialized for specific purposes, and consider making `<navtitle>` and `<searchtitle>` specializations of `<titlealt>`

Use cases

Describe how the feature will be used, if ideally implemented. Include any presentation or processing expectations.

New terminology

If any new terminology is needed to define the proposed solution, call out each new term here and provide a definition. Any new terminology will need to be used consistently across the entire specification.

Proposed solution

Provide an overview of how you expect to implement this feature or requirement.

Benefits

Address the following questions:

- Who will benefit from this feature?
- What is the expected benefit?
- How many people probably will make use of this feature? For example, everyone, many, or few.
- How much of a positive impact is expected for the users who will make use of the feature? For example, significant or minor.

Technical requirements

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

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

Adding new elements or attributes

Adding a topic or map specialization
Name of the new module, for example, *Glossary* or *Subject scheme*

Adding a domain
List of topic or map types that will include the domain

Adding an element

- Element name and attributes
- Description of the element, including any processing requirements
- Description of where the element will be permissable
- Is the element translatable? If yes, is it a block or phrase (subflow) element?

Adding an attribute

- Name of the attribute
- Elements for which the attributes will apply, for example, "new global attribute for conref-atts group" or "new specialized attribute for learning topics"
- Processing expectations that are associated with the new attribute
- Does the attribute contain translatable text? This should be avoided, so if the answer is "yes," you must explain why.

Renaming or refactoring elements and attributes

Renaming or refactoring an element
Include the current and new names of the element, the module that defines the element, and (if a domain element) the default document type shells that will be affected. If any other changes will be made to the element (new / removed attributes, modified content model, changed specialization ancestry) list those as well.

Renaming or refactoring an attribute

- Include the current and new names of the attribute.
- If values of the attribute change in any way, list the old and new values.
- If the element applies to a small set of elements, list the elements; if it is part of a large group of attributes (based on groupings in the DITA 1.3 specification section 3.10 Attributes), list the group.
- If the attribute has a new or modified specialization ancestry, specify the change.

Removing elements or attributes

Removing a topic or map specialization
Name of the module to be removed, for example, *Glossary* or *Subject scheme*

Removing a domain entirely, or from select document types
List of topic or map types that will remove the domain

Removing an element
Include the name of the element, the module that defines the element, and (if a domain element) the default document type shells that will be affected.

Removing an attribute

- Include the name of the attribute and the element that declares the element. If the attribute is removed from a larger group, list the group.
- If it is a specialized attribute, list the default document type shells that will be affected.

Processing impact

- How will the feature work?
- Will the feature have an impact on other processing features? For example, will the proposed feature have an impact on key resolution?
- Will the feature have to be evaluated before or after any existing features?
- What edge cases need to be considered?

Overall usability

Discuss the impact to current DITA users. Discuss any set up that users will need to do to make use of this feature.

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?

If this change removes an already-deprecated element, attribute, or feature, please note when it was deprecated.

Removing a document type that was shipped in DITA 1.3?

If so, which?

Removing a domain that was shipped in DITA 1.3?

If so, which? What document types are affected?

Removing a domain from a document type shell was shipped in DITA 1.3?

If a domain is still shipped but is no longer included in a (previously shipped) shell, what document types are affected?

Removing or renaming an element that was shipped in DITA 1.3?

If so, which, and from what module?

Removing or renaming an attribute that was shipped in DITA 1.3?

If so, which, and from what elements?

Changing the meaning of an element or attribute in a way that would disallow existing usage?

If so, how does the definition change?

Changing a content model by removing something that was previously allowed, or by requiring something that was not?

What are the old and new content models?

Changing specialization ancestry?

What are the old and new class attribute values?

Removing or replacing a processing feature that was defined in DITA 1.3?

What is the feature that would be removed?

Are element or attribute groups being renamed or shuffled?

What group is changing? Is the name changing, or is content being added / removed?

Migration plan

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

- Might any existing documents need to be migrated? If so, how would this likely be accomplished:
  - Manual updates or search/replace within files?

    - Script or other automated process to be provided by owner of this proposal?
    - Expecting vendor tools to accommodate migration?
    - Other method?
- Might any existing processors or implementations need to change their expectations?
- Might any existing specialization or constraint modules need to be migrated? For example:
    - Are elements or attributes removed or renamed, such that they also need to be adjusted in specialization / constraint declarations?
    - Do content models change in a way that will impact existing specializations / constraints?
    - If specialization ancestry is changing, how likely is it that specializations of that element exist and need to be updated?
    - Are content model groups (like `basic.block`) or attribute groups (like `topicref.atts` or `univ.atts`) being renamed or shuffled in a way that will affect their usage in specialization or constraint modules? Note: moving a common attribute that is declared individually *into* such a group impacts specializations / constraints, because they will now have to declare the attribute, or the group, but not both.
    - Is an element or attribute moving out of the base and into a new module, so that configured grammar file shells must be updated to make use of it?
- If no migration need is anticipated, why not?

Costs

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

- Maintainers of the grammar files
- Editors of the DITA specification:
    - How many new topics will be required?
    - How many existing topics will need to be edited?
    - Will the feature require substantial changes to the information architecture of the DITA specification? If so, what?
    - If there is new terminology, is it likely to conflict with any usage of those terms in the existing specification?
- Vendors of tools: XML editors, component content management systems, processors, etc.
- DITA community-at-large:
    - Will this feature add to the perception that DITA is becoming too complex?
    - Will it be simple for end users to understand?
    - If the feature breaks backwards compatibility, how many documents are likely to be affected, and what is the cost of migration?
- 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?
    - Will this require an independent white paper or other publication to provide migration details?
    - 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?

Examples

Provide examples of the proposed feature. Include an example for each of the use cases. Be sure to include edge cases, if known.

If this proposal involves migration of existing DITA 1.3 content for compatibility with DITA 2.0, give examples of valid content before and after migration.

2.1.4 Stage two: #17 Make @outputclass universal

Make `@outputclass` a universal attribute.

Date and version information

Include the following information:

- Presented as stage 1 proposal at TC meeting on January 24, 2017: https://www.oasis-open.org/committees/document.php?document_id=59918&wg_abbrev=dita
- First draft of stage 2 proposal: June 12, 2017
- Champion: Robert Anderson
- Links to any previous versions of the proposal: N/A
- Links to e-mail discussion that resulted in new versions of the proposal: N/A

Original requirement or use case

The `@outputclass` attribute is currently a near-universal attribute on content elements. The original definition was:

> Provides a value that will be added to the HTML class attribute for the associated output element, for use by CSS style sheets. May also be used by other output processes.

Effectively the intent for the attribute was to create a simple way to sub-class information. For example, "This is a section, with the subclass of X". Because of the original expectation that it would be used to style content in HTML, it was added universally to content elements, but only sporadically to metadata or other elements. Even with DITA 1.0, architects immediately took advantage of this ability to sub-class elements and used `@outputclass` as a way to mock-up specializations. This doesn't work for metadata elements without the attribute, so each subsequent release of DITA has added it to more elements, but still not to every element. To reduce confusion and enable sub-classing of any element, this proposal adds `@outputclass` to every element (except `<dita>`, which does not even have `@class` and is purely a container).

Use cases

- The lightweight DITA SC initially wanted to use `@outputclass` as part of a template based specialization process. This did not work simply because the attribute is unavailable on a couple of elements. Others have had this same idea and run into the same limitation. Making `@outputclass` universal on elements that can be specialized removes this block and enables template-based processes that have been envisioned since DITA 1.1.
- It is now generally recognized that the specification itself does not control which elements result in content. Any element (even metadata) may need to be rendered. The lack of `@outputclass` means that sub-classing can be used to style most elements, but cannot be used on a few others. Making this available everywhere enables quick styling on all elements.
- In terms of overall complexity, having `@outputclass` available on all content elements + many (but not all) metadata elements just causes confusion when somebody finds it missing. In this

case, having the attribute available globally simplifies both the architecture and the cognitive load on those using it.

New terminology

N/A

Proposed solution

In both maps and topics, add `@outputclass` as an attribute on all elements that do not have it, apart from the `<dita>` container element.

Benefits

Address the following questions:

- Authors who already expect `@outputclass` to be available everywhere will no longer be confused that it is missing on a few elements.
- Tools long envisioned using this attribute (such as template based specialization) can now be realized.
- The DITA specification can be simplified for both readers and maintainers by grouping `@outputclass` among other universal attributes.
- Maintainers of DITA architecture will no longer have to figure out whether the attribute is available on individual elements.
- If tools are developed that depend on a global `@outputclass` attribute, it's possible that many will see benefit from this change. Outside of that, it is likely to be a positive improvement for many DITA users, with a greater impact on information architects / those designing grammar files (who are the ones most likely to be concerned with or confused by this today).

Technical requirements

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

Adding new elements or attributes

Adding an attribute

- `@outputclass` will be added to all elements that do not currently allow it, *except* for the `<dita>` container element. (That element cannot be subclassed / specialized today, and does not have any of the otherwise universal attributes apart from localization attributes.)
- No additional processing is required for this attribute.
- This attribute does not contain translatable text.

Processing impact

- Applications may add processing support for `@outputclass` on the new elements, but no specific processing is required, and any intended processing should match processing already available elsewhere.

Overall usability

The attribute is already widely used without problem, so there should not be any usability concerns.

Backwards compatibility

This feature is fully backwards compatible with DITA 1.x.

Migration plan

There is no need for migration.

Costs

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

- Maintainers of the grammar files: minor change to make this available where it is not. Most likely this means placing the attribute into a reused group of attribute definitions, and removing individual definitions of the attribute.
- Editors of the DITA specification:
  - How many new topics will be required: zero.
  - How many existing topics will need to be edited: Many language reference topics will remove an extra link to the `@outputclass` definition, and the definition itself will need to be placed with other universal attributes.
  - The feature will not require substantial changes to the information architecture of the DITA specification, and involves no new terminology.
- Vendors of tools: no cost
- DITA community-at-large:
  - Will this feature add to the perception that DITA is becoming too complex? No: it simplifies DITA by avoiding the arbitrary decisions about what elements need `@outputclass`
  - Will it be simple for end users to understand? Yes
- Producing migration instructions or tools: N/A

Examples

The attribute is not currently available on `<prolog>`; adding it would make the following example a legal way to mock up a specialization of `<topic>`. Note that this would also become a valid mock-up using LwDITA, though does not contain all of the data envisioned for the latest template based specialization process. Both full and LwDITA would currently prohibit this use because `@outputclass` is not valid on `<prolog>`.

```
<topic id="mockup" outputclass="bookreport">
  <title outputclass="booktitle">Sample title</title>
  <prolog outputclass="bookreportdata">
    <data outputclass="pages" value="lots"/>
    <data outputclass="stars" value="many"/>
    <data outputclass="did-i-finish" value="of course"/>
  </prolog>
  <body outputclass="bookreportbody">
    <section outputclass="bookabout">Here is what the book was about</section>
    <section outputclass="bookthoughts">Here is what I thought about it</section>
  </body>
</topic>
```

2.1.5 Stage two: #18 Make audience, platform, product, otherprops into specializations

DITA 2.0 should refactor the existing profiling attributes – `@audience`, `@platform`, `@product`, and `@othermeta` – so that they're defined in domains and specialized from `@props`, as we did for `@deliveryTarget` in 1.3.

Date and version information

Date that this feature proposal was completed
February 27 2018

Champion of the proposal
Robert D. Anderson

Links to any previous versions of the proposal
Original proposal from Chris Nitchie: https://lists.oasis-open.org/archives/dita/201703/msg00033.html

Links to minutes where this proposal was discussed at stage 1 and moved to stage 2
https://lists.oasis-open.org/archives/dita/201705/msg00015.html

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/18

Original requirement or use case

As discussed at the TC May 2, 2017, the goal is to clean up our architecture by making the original four filter attributes into specializations of `@props`. They already function exactly the same way as `@props` (including the fact that they allow the grouping syntax that is exactly equivalent to the generalized attribute syntax). However, they were defined as part of the core before we had specialized attributes, so for backwards compatibility they have remained in the core even though logically they must be treated exactly the same as any specialization of `@props`.

Use cases

1. To clean up the language and simplify processors.

   With DITA 1.0, the four attributes (`@audience`, `@platform`, `@product`, and `@otherprops`) were intended to be the only attributes available for filter / flag metadata processing. When attribute specialization was developed, a new base attribute called `@props` was added, also for filtering / flagging. To fully support the specification around filtering and flagging, processors must now support:

   - The `@props` attribute
   - Any unrecognized attributes that are properly specialized from `@props`
   - Generalized syntax within any of those attributes
   - A general processor must do all of that, but then add in exceptions for the 4 original attributes that function *exactly the same* but are not actually declared as specialized attributes.

   With these four turned into specializations, any processor that supports `@props` + specializations of `@props` is complete.
2. To allow proper specializations of `@audience`, `@platform`, `@product`, and `@otherprops`. Authors have at times requested the ability to specialize these attributes - that need is explicitly

what led to the grouping syntax added in DITA 1.3. However, because of the original DITA 1.0 design, these attributes could not be handled / specialized in the same way. Converting them to specializations of `@props` will allow architects to create new attributes off of these four, such as defining `@appserver` and `@database` as specialized variants of `@product`.

New terminology

N/A; any terminology needed for these already exists for specialized attributes.

Proposed solution

1. Create four new domain modules, one for each of `@audience`, `@platform`, `@product`, and `@otherprops`. Having these as four individual domains (rather than as one group) allows any architect to more easily remove any that do not apply – for example, keeping `@audience` while removing the other three.
2. Remove the existing definitions of these attributes. In OASIS grammars, these are defined as base attributes in the `%filter-atts;` group along with `@props` and a placeholder for extensions of `@props`. This is the only place the attributes are named in OASIS grammars (DTD: `commonElements.mod`; RNG: `commonElementsMod.rng`).
3. Add the domain attributes into the shells of all OASIS-provided grammar files, and add them to the props-extension group. This will restore the attributes to any element that previously defined them.

Note Based on a suggestion from Tom when discussing this stage 2 proposal at TC meeting on 20 March 2018, an updated definition for `@otherprops` should be part of the stage 3 proposal. The updated definition should explain when the attribute should be used (versus creating a new specialized attribute).

Benefits

Who will benefit from this feature?

- Processors can be simplified with one set of rules for all filtering / flagging
- Information architects gain the ability to remove any or all of these 4 without needing a constraint
- Information architects gain the ability to specialize directly from the existing attributes
- Maintainers of DITA specification can simplify language around these attributes and (more importantly) attribute grouping, which becomes an exact match for generalization syntax

How many people probably will make use of this feature?

Realistically, few authors will see a change, as many architects will likely restore all four attributes into existing shells, but this is just a guess.

How much of a positive impact is expected for the users who will make use of the feature?

Minor impact but goes along with the overall goal of simplifying the language.

Technical requirements

Adding new elements or attributes

Adding a domain

All existing shells will get these four new attribute extension domains, with zero overall change to content / attribute models. The domains will exactly match the pattern of the current `@deliveryTarget` attribute, both in how the module is defined and in how it is integrated into shells.

Adding an attribute

The four domains will add a total of four attributes, but with no overall change to the language as a whole.

Removing elements or attributes

Removing an attribute

The attribute definitions will be moved, which will technically remove the attributes from any shells that do not restore them. All DITA document types will be affected, and will need to update their shells as part of any migration to DITA 2.0.

Processing impact

N/A

Overall usability

No change to authors. Architects that maintain shells will need to go through the one-time process of adding these domains.

Backwards compatibility

Was this change previously announced in an earlier version of DITA?

It has been discussed in an unofficial capacity many times, but I'm not sure if it was officially announced.

Removing or renaming an attribute that was shipped in DITA 1.3?

The four attributes in question, from all elements.

Are element or attribute groups being renamed or shuffled?

This is closest in concept to the suggested change; the attributes are not going away, but moving them from the `%filter-atts;` group into their own domains will require updates to shells, but will have no impact on DITA documents.

Migration plan

- No documents will need to be migrated.
- Processors may optionally remove exceptions in any filter/flagging code that explicitly look for these attributes, but as long as the processors correctly support `@props` and specializations, they will continue to work as designed.
- Document types that wish to retain all four attributes will need to add the four new domains into shells.
- Constraints that *remove* any of these attributes today will need to be updated; it may be possible to discard the constraint module and simply not include the relevant domain(s).
- Constraints that explicitly declare or restrict values on these attributes will need to be updated.

Costs

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

- Maintainers of the grammar files: very small cost of removing the attribute definitions, small cost of creating and integrating the domain modules based on the existing `@deliveryTarget` pattern.
- Editors of the DITA specification:
  - No new topics required.
  - Definitions of these attributes will likely remain where they are, together in one topic with `@deliveryTarget`, but need additional language to clarify that they are domain specializations.

    - The topics about filtering and grouping syntax will need editing, and can be simplified to treat all of these attributes in the same way as `@props`.
- Implementations: likely a low cost to remove extra support for these attributes, and allow them to use general support based on `@props`
- DITA community-at-large: those only concerned with authoring should see no change. There will be a small cost associated with updating shells, but we can provide straightforward patterns that allow cut-and-paste updates. Costs for updating constraints are harder to determine, because we do not know how many exist or what they are doing, but should not be large.
- Producing migration instructions or tools:
    - Small cost for instructions on updating shells - basically, documenting the updates we make to our own shells.
    - Small additional cost with (speculative) updates about impacts on constraint modules, with examples of how to update those we consider most likely to exist.
    - No special white paper or publication is needed solely for this feature.
    - Because grammar files can follow different patterns, cut-and-paste updates from the migration document are likely to be more reliable than a tool that updates DTD/XSD/RNG files.

Examples

N/A; content is exactly the same.

2.1.6 Stage two: #21 Resolve inconsistent class values for <shortdesc>, <linktext>, and <searchtitle>

Resolve inconsistent class values for `<shortdesc>`, `<linktext>`, and `<searchtitle>`.

Date and version information

Date that this feature proposal was completed
12 March 2018

Champion of the proposal
Robert Anderson

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/201705/msg00022.htmlhttps://lists.oasis-open.org/archives/dita/201705/msg00022.html

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/21https://github.com/oasis-tcs/dita/issues/21

Original requirement or use case

These elements are the same semantic and thus should have a single consistent `@class` value across all DITA document types.

Use cases

Remove the need to have separate handling for the different map and topic versions of these elements.

New terminology

No new terminology.

Proposed solution

Move the declarations for `<shortdesc>`, `<linktext>`, and `<searchtitle>` to `commonElements`, making them available to both maps and topics.

Benefits

Who will benefit from this feature?

Implementors of DITA processors. Creators of specializations `<shortdesc>`, `<linktext>`, or `<searchtitle>`.

What is the expected benefit?

Simplifies DITA processor implementation by removing the need for special cases for these very common elements. Removes the need for specializers to be careful to select the correct base for their specializations or to be forced to create two different specializations for the same semantic.

How many people probably will make use of this feature?

Affects all DITA processor implementors.

How much of a positive impact is expected for the users who will make use of the feature?

Minor reduction in implementation complexity.

Technical requirements

Renaming or refactoring an element

- Remove all declarations for `<shortdesc>`, `<linktext>` and `<linktitle>` from `mapMod`.
- Move declarations for `<shortdesc>`, `<linktext>` and `<linktitle>` from `topicMod` to `commonElementsMod`.

Processing impact

No change

Overall usability

No change

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?

Change is new to DITA 2.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?
No.

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?
Yes: Removing `map/*` versions of the `<shortdesc>`, `<linktext>`, and `<searchtitle>` elements.

Removing or replacing a processing feature that was defined in DITA 1.3?
No.

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?
Documents that have literal `@class` values for these elements will need to either remove the `@class` attributes from the source or update them to reflect the new `@class` values. Otherwise existing documents will not be affected.

Might any existing processors or implementations need to change their expectations?
Processors that have processing that looks for the `map/*` versions of the `@class` values can remove those checks.

Might any existing specialization or constraint modules need to be migrated?
There are no OASIS-defined specializations of `<shortdesc>`, `<linktext>`, or `<searchtitle>`. Local specializations can be updated with a simple search and replace to change "map" to "topic" for these three element types.

Costs

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

Maintainers of the grammar files
Update grammar files to reflect removal of `map/*` versions of the files. Work done as part of this Stage 2 proposal.

Editors of the DITA specification
No impact.

Vendors of tools
May need to update tools to remove matches on `map/shortdesc`, `map/linktext`, and `map/searchtitle`.

DITA community-at-large

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

  This proposal reduces complexity.
- Will it be simple for end users to understand?

  Most DITA authors should be completely unaware of the change.
- If the feature breaks backwards compatibility, how many documents are likely to be affected, and what is the cost of migration?

  Documents that use default `@class` values will not be affected. Map documents that have literal `@class` values will need to be updated to either remove the `@class` attributes or update the values if the `@class` values must be maintained in the source.

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?

  One or two paragraphs at most.
- 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?

  Might be useful to include as part of a general migration tool but definitely not critical.

Examples

N/A.

2.1.7 Stage two: #27 Multimedia domain

A DITA domain for presentation of multimedia (audio/video) content.

Date and version information

Revision History

August 17, 2017

- Created `<param>` specializations for `<media-autoplay>`, `<media-loop>`, and `<media-muted>`.
- Added proposal to add `<fallback>` to `<object>`.
- Cleaned up and disambiguated language throughout.

June 27, 2017

Fixed 'terminology' section.

2 - June 27, 2017

Fixing typos. Several references to "metadata" should have been "multimedia" or "media".

1 - June 26, 2017

Initial revision.

Completion date

June 26, 2017

Proposal Champion
Chris Nitchie

Initial Suggestion
https://lists.oasis-open.org/archives/dita/201705/msg00048.html

GitHub Issue
https://github.com/oasis-tcs/dita/issues/27

Original requirement or use case

DITA should have an HTML5-compatible multimedia domain to simplify authoring references to audio and video information. Also, since Lightweight DITA will contain multimedia elements, it's important that they be accounted for in full DITA.

Use cases

An author wishes to references audio and/or video content from a DITA topic.

New terminology

None

Proposed solution

Introduce `<audio>` and `<video>` elements specialized from `<object>` and, as much as possible, directly analogous to their HTML5 equivalents.

Also, introduce a new `<fallback>` element as part of the `<object>` content model as well as that of `<audio>` and `<video>`.

Benefits

Authors of content referencing audio and video content will benefit significantly from not having to use the generic `<object>` markup. Stylesheet and processing developers will find it much easier to map these elements to HTML5 than detecting the appropriate HTML5 markup based on available clues in `<object>`.

Technical requirements

This solution will create a new multimedia domain, `media-d`, which will be included in the base DITA 2.0 package and referenced from the Topic document type shell. This domain will consist of the following new elements.

<fallback>
An element containing markup to be presented when the multimedia object represented by the containing `<object>` element (or `<object>` specialization) cannot be presented.

Content Model

```
(#PCDATA | %basic.block.notbfgobj; | %basic.ph; | %data.elements.incl;
| %draft-comment; | %foreign.unknown.incl; | %required-cleanup;)
```

Attributes

- `%univ-atts;`
- `outputclass`

<video>

Represents video content. Specialized from `<object>`.

Content Model

```
(desc?, longdescref?, fallback?, video-poster?, media-controls?, media-autoplay?, media-loop?, media-muted?, media-source*, media-track*, param*, (foreign | unknown)*)
```

Attributes

- `@data`
- `@datakeyref`
- `@width`
- `@height`
- `%univ-atts;`

Note There are a number of important HTML5 attributes that are not represented in this element:

- `@buffered`
- `@crossorigin`
- `@preload`

These attributes specify technical details of the delivery of the media and should be set by the delivery system or HTML transform as necessary. As such, they are not represented in the DITA model. If necessary, they could be configured using `<param>` elements.

<audio>

Represents audio content. Specialized from `<object>`.

Content Model

```
(desc?, longdescref?, fallback?, media-controls?, media-autoplay?, media-loop?, media-muted?, media-source*, media-track*, param*, (foreign | unknown)*)
```

Attributes

- `@data`
- `@datakeyref`
- `%univ-atts;`

Note As with `@video`, a number of important HTML5 attributes go unaccounted for here, including:

- `@autobuffer`
- `@buffered`
- `@preload`

These could be encoded as `<param>` tags.

Other Supporting Elements

The content models of `<video>` and `<audio>` are very similar, and make use of a number of specializations of `<param>` that map closely, though not exactly, to HTML5 equivalents. These are described using loose DTD syntax below.

```
<!ELEMENT media-controls EMPTY>
```

```
<!ATTLIST media-controls
  name CDATA #FIXED 'controls'
  <!-- Processors should default to 'true' -->
  value (true|false|-dita-use-conref-target) #IMPLIED
  class CDATA '+ topic/param media-d/media-controls '
  %univ-atts;
>
<!ELEMENT media-autoplay EMPTY>
<!ATTLIST media-autoplay
  name CDATA #FIXED 'autoplay'
  <!-- Processors should default to 'true' -->
  value (true|false|-dita-use-conref-target) #IMPLIED
  class CDATA '+ topic/param media-d/media-autoplay '
  %univ-atts;
>
<!ELEMENT media-loop EMPTY>
<!ATTLIST media-loop
  name CDATA #FIXED 'loop'
  <!-- Processors should default to 'false' -->
  value (true|false|-dita-use-conref-target) #IMPLIED
  class CDATA '+ topic/param media-d/media-loop '
  %univ-atts;
>
<!ELEMENT media-muted EMPTY>
<!ATTLIST media-muted
  name CDATA #FIXED 'muted'
  <!-- Processors should default to 'false' -->
  value (true|false|-dita-use-conref-target) #IMPLIED
  class CDATA '+ topic/param media-d/media-muted '
  %univ-atts;
>
<!ELEMENT video-poster EMPTY>
<!ATTLIST video-poster
  name CDATA #FIXED 'poster'
  <!-- A URI -->
  value CDATA #IMPLIED
  keyref CDATA #IMPLIED
  type CDATA #IMPLIED
  valuetype (ref) #FIXED 'ref'
  class CDATA '+ topic/param media-d/video-poster '
  %univ-atts;
>
<!ELEMENT media-source EMPTY>
<!ATTLIST media-source
  name CDATA #FIXED 'source'
  <!-- A URI -->
  value CDATA #IMPLIED
  keyref CDATA #IMPLIED
  type CDATA #IMPLIED
  valuetype (ref) #FIXED 'ref'
  class CDATA '+ topic/param media-d/media-source '
  %univ-atts;
>
<!ELEMENT media-track EMPTY>
<!ATTLIST media-track
  name CDATA #fixed 'track'
  <!-- A URI -->
  value CDATA #IMPLIED
  keyref CDATA #IMPLIED
  <!-- HTML5 'kind' Attribute -->
  type (subtitles | captions | descriptions | chapters | metadata | -dita-use-conref-
target) #IMPLIED
  valuetype (ref) #FIXED 'ref'
  class CDATA '+ topic/param media-d/media-track '
  %univ-atts;
>
```

Important The `@xml:lang` attribute on the `<media-track>` element identifies the language of the referenced track. This is analogous to the `@srclang` attribute in HTML5.

Note For the boolean on/off configuration elements - `<media-controls>`, `<media-autoplay>`, `<media-loop>`, and `<media-muted>` - if no `@value` or `@keyref` attributes are specified, the processing default `@value` should is considered "true".

Processing impact

While this domain's markup is not exactly the same as the HTML5 `<audio>` and `<video>` markup, most of the elements and attributes have one-to-one mappings to the equivalent HTML5 markup, making HTML conversions relatively simple. Conversions into other output formats, like PDF, may require extra effort, depending on the capabilities of the target format(s), and should use `<fallback>` when audio or video content is not supported.

Overall usability

This will add to the available tagging in many tag contexts, but will greatly simplify audo and video embedding in DITA-authored content.

Backwards compatibility

No backwards compatibility concerns.

Migration plan

N/A

Costs

Grammar file maintainers will have a new, self-contained domain to deal with and maintain.

The DITA spec should be updated with the following content:

- A new topic describing the embedding of multimedia content using this domain and/or the `<object>` tag.
- Language reference topics for all new elements.

Tool vendors will need to update authoring and publishing stylesheets to simplify authoring of this markup, and proper handling for different output types.

Examples

```
<!-- An audio element with a single source -->
<audio data="mysong.mp3"/>

<!-- An audio element with the same audio content in three different formats -->
<audio data="mysong.mp3">
   <fallback>Audio playback is unavailable</fallback>
   <media-source value="mysong.wav" type="audio/wav" />
   <media-source value="mysong.ogg" type="audio/ogg" />
</audio>

<!-- An audio element with controls turned on, and looping turned off -->
<audio data="mysong.mp3">
   <fallback>Audio playback is unavailable</fallback>
   <media-controls value="true" />
   <media-autoplay value="false" />
   <media-loop value="false" />
</audio>

<!-- A video element with a single source referenced by key -->
<video datakeyref="exampleVideo"/>

<!-- A fuller example. -->
<video width="500px" height="300px">
  <fallback>Video playback not available.</fallback>
```

```
   <media-controls value="false"/>
   <video-poster value="poster.png" />
   <media-source value="myvideo.mp4" type="video/mpeg-4"/>
   <media-source value="myvideo.ogg" type="video/ogg"/>
   <media-track value="myvideo-subtitles-en.vtt" type="subtitles" xml:lang="en"/>
   <media-track value="myvideo-subtitles-fr.vtt" type="subtitles" xml:lang="fr"/>
   <param name="autoplay" value="true"/>
 </video>
```

2.1.8 Stage two: #29 Update <bookmap>

The goal of the redesign is to remediate problems but avoid breaking backward compatibility.

Modify bookmap design

Proposal version
1.1

Completion date
March 27, 2018

Champion of the proposal
Eric Sirois

Initial Suggestion
https://lists.oasis-open.org/archives/dita/201703/msg00019.html
https://lists.oasis-open.org/archives/dita/201807/msg00045.html

Links to minutes where this proposal was discussed at stage 1 and moved to stage 2
https://lists.oasis-open.org/archives/dita/201705/msg00091.html

GitHub Issue
https://github.com/oasis-tcs/dita/issues/29

Original requirement or use case

Overall the bookmap DTD has not been updated as part general DITA releases for a couple releases. It is missing some convenient key features that were added as part of DITA 1.2 and 1.3. The proposal will make it easier to make use of `<ditavalref>` at the root level and make it easier to create and maintain keydefs.

Use cases

1. A writer wants to create a keydefs for a map in a convenient location in the map.

   The current version of the DITA Bookmap does allow keydefs to be created, but you need to do it inside one of the main top-level elements. like `<frontmatter>`. But it's not convenient because it's buried in an unrelated element.
2. A writer wants to specify a ditaval used for the whole map via a `<ditavalref>`.

   The current version of the DITA Bookmap does not allow `<ditavalref>` as an immediate child of `<bookmap>`. Creating them inside one of the child elements of `<bookmap>`, like `<frontmatter>`, then the `<ditavalref>` only applies to the children of that element.
3. A writer wants to indicate a change history list for a book.
4. A writer wants to reference some amendments in the `<frontmatter>` for a book.
5. A writer wants to use `<glossref>` to reference a glossary topic within `<glosslist>`.

Proposed solution

The purpose of the proposal is to add some enhancements to bookmap, with a minimal amount of impact at the processing end of the pipeline, but make things a bit easier for the end user.

- Add <ditavalref> before <frontmatter>
- Create a <mapresources> as an element that will allow users to reference elements that are specializations of <topicref>.
- Add <amendments> to <booklists>. If <amendments> contains an @href, then process as expected. Otherwise, process the content for revision history
- Add <glossref> to element content model for <glosslist>.

Benefits

- Authors will benefit from the more intuitive use of keydefs.
- Information architects/developers will benefit being able to add a specialization for custom processing of bookmaps.
- Information architects will be able to use the same bookmap for multiple outputs using different set of conditions

Technical requirements

This change is adding two elements, that already supported in DITA 1.3, to the root content model of the bookmap DTD and as such, will have essentially no technical impact. There will be an impact for implementing the logic to support change-history from <amendments>

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

DTD and Schema modifications.

New domain

- Map resources
  - Add a domain specializations that will allow the definition of a wrapper element to define <keydefs> in a <bookmap> or other strictly constrained publication maps. This domain is based on <mapresources>

New element

- <mapresources>
  - This element is a container that allows specializations of <topicref> to be available in a bookmap. The element is defined as resource-only element. The element also allows @keyscope.
  - The element is optional after the <ditavalref>.

Add existing DITA elements to content models in bookmap

- <ditavalref>
  - This is an existing element that is available in the base <map>. We are simply making it available in the top level portion of <bookmap>
  - The element is optional after the <bookmeta>.
- <glossref>
  - This is an existing element that is available in the base <map>. We are simply making it available in the top level portion of <bookmap>

    - The element is optional after the <glosslist>.
- <amendments>
    - This is an existing element that is available in <backmatter>. We are simply making it available in <booklists> and adding behaviour on the processing side.
    - Since the content model for <booklists> is the same in <frontmatter>, then it will also be available in in <frontmatter>

Processing impact

<ditavalref> and <glossref> are already supported features of DITA 1.3. And <mapresources> is a specialization of topicref as a resource-only element.

For <amendments>, if the element does not have a reference to a topic, then process the content in the map as revision history when <change-historylist> element exist.

Overall usability

Current DITA users and current document type shells will be unaffected. Usability of authoring will ultimately be improved due to better management of keydefs and use of ditavalrefs in the bookmap.

Costs

- The main bookmap files need to be updated with the addition of a new domain file to support the solution for this proposal.
- A new topic for documenting the domain and the container element. As well as an update to the bookmap topic.
- Since we are adding elements that are already supported in DITA 1.3, there is no impact to processors and XML tools.
- Adding functionality in the base PDF processing to treat <amendments> without an @href to auto-generate a revision history, based on the <change-historylist>.

Examples

Sample usage in a bookmap:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bookmap PUBLIC "-//OASIS//DTD DITA 2.0 BookMap//EN" "bookmap.dtd">
<bookmap>
    <booktitle>
        <mainbooktitle>Test</mainbooktitle>
    </booktitle>
    <ditavalref href="test.ditaval"/>
    <mapresources>
        <keydef keys="test" href="test.dita"></keydef>
    </mapresources>
</bookmap>
```

Element definition in the bookmap DTD:

```
 <!ENTITY % bookmap.content
 "((%title; | %booktitle;)?,
 (%bookmeta;)?,
 (%ditavalref;)*,
 (%mapresources;)*,
 (%frontmatter;)?,
 (%chapter;)*,
 (%part;)*,
```

```
     ((%appendices;)? |
     (%appendix;)*),
     (%backmatter;)?,
     (%reltable;)*)"
     >
```

Element override in the bookmap shell:

```
<!ENTITY % mapresources  "%mapresources-d-mapresources;
 ">
```

Domain definition:

```
<!ENTITY % mapresources
                      "mapresources"                             >


<!-- ============================================================= -->
<!--                    ELEMENT DECLARATIONS                       -->
<!-- ============================================================= -->


<!ENTITY % mapresources.content
                      "(%topicref;)*"
>

<!ENTITY % mapresources.attributes
              "outputclass
                           CDATA
                                      #IMPLIED
               keyscope
                           CDATA
                                      #IMPLIED
               processing-role
                           CDATA
                                      'resource-only'
               %univ-atts;"
>

<!ELEMENT  mapresources %mapresources.content;>
<!ATTLIST  mapresources %mapresources.attributes;>

<!ATTLIST  mapresources %global-atts;  class CDATA "+ map/topicref  mapresources-d/
mapresources ">
```

Domain entity definition:

```
    <!ENTITY % mapresources-d-mapresources
    "mapresources"
    >

    <!ENTITY mapresources-d-att
    "(map mapresources-d)"
    >
```

Adding amendments to booklists

```
<!-- LONG NAME: Book Lists -->
<!ENTITY % booklists.content
                     "(%abbrevlist; |
                       %amendments; |
                       %bibliolist; |
                       %booklist; |
                       %figurelist; |
                       %glossarylist; |
                       %indexlist; |
                       %tablelist; |
```

```
                        %trademarklist; |
                        %toc;)*"
>
```

Adding glossref to the glossarylist

```
<!--                    LONG NAME: Glossary List                    -->
<!ENTITY % glossarylist.content
                        "((%topicmeta;)?,
                          (%topicref;|
                  %glossref;)*)"
>
```

2.1.9 Stage two: #34 Remove <topicset> and <topicsetref>

Remove topicset and topicsetref.

Date and version information

Date that this feature proposal was completed
11 March 2018

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

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/34https://github.com/oasis-tcs/dita/issues/34

Original requirement or use case

The `<topicset>` and `<topicsetref>` elements seem to provide very little value over ordinary `<topicref>` elements. In addition, they regularly suggest to authors that they should result in some sort of standard special processing (which they do not). The semantic of `<topicset>` was never clearly distinguished from any other use of topic reference elements to group topicrefs (e.g., `<topicgroup>`).

The semantics of the `<topicsetref>` element were never clearly defined in a way that was not tied directly to specific delivery tools (i.e., Eclipse help). Map references and the DITA 1.3 cross-deliverable linking facility appear to satisfy all the requirements implied for `<topicsetref>`.

As far as can be determined, there are no DITA processors in general use that provide any special processing for `<topicset>` or `<topicsetref>`.

Use cases

This removes elements. No usage use case.

New terminology

No new terminology.

Proposed solution

Remove the <topicset> and <topicsetref> elements from the grammars and all specification topics that reference them.

Benefits

Who will benefit from this feature?

All DITA users currently confused by the presence of these elements.
DITA tool implementors who have no idea how to implement these elements correctly or worry about having ignored a DITA feature by not implementing them.

What is the expected benefit?

Reduced complexity.

How many people probably will make use of this feature?

N/A

How much of a positive impact is expected for the users who will make use of the feature?

Hard to quantify impact of removing an element that few if any DITA users are using. Will reduce the set of <topicref> specializations one has to consider when learning about or authoring DITA maps.

Technical requirements

Grammar and specification changes

Remove <topicset> and <topicsetref> from the *Map Group* domain:

- Remove from mapGroupDomain.rng
- Remove from mapGroup.ent
- Remove from mapGroup.mod
- Remove from mapGroupMode.xsd

Update the following topics to remove references to <topicset> and <topicsetref>:

- ditamap-elements.dita
- commonNavLibraryTable.dita
- base-elements-a-to-z.dita
- all-elements-a-to-z.dita
- map-group-elements.ditamap
- Regenerate content model files to reflect removal from grammar files.

Remove the following topics entirely:

- langRef/base/topicsetref.dita
- langRef/base/topicset.dita

Processing impact

Any processor that actually provides special processing for <topicset> or <topicsetref> can remove it.
If a processor provides unique processing for <topicset> or <topicsetref> for which support needs to be retained, the processor will need to either provide its own specializations to which the processing can then be applied or provide some other way to signal the desired behavior, for example, processor-specific values for @outputclass.

Overall usability

No impact.

Backwards compatibility

Was this change previously announced in an earlier version of DITA?

No.

Removing or renaming an element that was shipped in DITA 1.3?

The `<topicset>` and `<topicsetref>` elements will be removed, so any documents currently using them cannot be valid against the DITA 2.0 grammars as provided by OASIS.

Migration plan

Might any existing documents need to be migrated?

Anyone currently using these elements will need to either replace them with normal topicrefs or define their own specializations to replace `<topicset>` or `<topicsetref>`.

Might any existing processors or implementations need to change their expectations?

If a processor provides unique processing for `<topicset>` or `<topicsetref>` for which support needs to be retained, the processor will need to either provide its own specializations to which the processing can then be applied or provide some other way to signal the desired behavior, for example, processor-specific values for `@outputclass`.

As far as the TC can determine, there are no implementations of `<topicset>` or `<topicsetref>` in general use.

Might any existing specialization or constraint modules need to be migrated?

In the unlikely event that there are specializations of `<topicset>` or `<topicsetref>`, those specializations will need to be redefined to either specialize directly from `<topicref>` or another suitable `<topicref>` specialization, such as `<mapref>`, or define a new domain module that provides the equivalent of `<topicset>` and `<topicsetref>` to then serve as the specialization base of the specializations of `<topicset>` and `<topicsetref>`.

Costs

Maintainers of the grammar files

Remove all declarations and comments that mention or use `<topicset>` or `<topicsetref>` (already done as part of this Stage 2 proposal).
Regenerate content model topics and any generated navigation lists to reflect the removal (inherent in work that will have to be done for DITA 2.0 in any case).

Editors of the DITA specification

- How many new topics will be required?

  None
- How many existing topics will need to be edited?

  Five authored topics (see above) plus removal of topicrefs from one map. (Work already done as part of this Stage 2 proposal)
- Will the feature require substantial changes to the information architecture of the DITA specification? If so, what?

  No architectural change required.

- 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 ensure that there are not any hard-coded references to the `<topicset>` and `<topicsetref>` elements. Any processing specific to those elements will need to either be removed or rebound to different elements or ways of signaling the need for the special processing, whatever it might be.

DITA community-at-large

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

  It should not as we are removing a confusing and largely unused element.
- Will it be simple for end users to understand?

  Yes.
- If the feature breaks backwards compatibility, how many documents are likely to be affected, and what is the cost of migration?

  There should be few documents using `<topicset>` or `<topicsetref>`. Those that are can very likely simply replace their use with normal `<topicref>` elements.

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 not require more than a paragraph or two at most.
- 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?

  It might be useful to include a general `<topicset>` and `<topicsetref>` to `<topicref>` transform as part of a general migration tool.

Examples

N/A

2.1.10 Stage two: #36 Remove deprecated elements and attributes

Remove elements and attributes that have been designated as deprecated, "reserved for future use," or defined by mistake and retained only to maintain backwards compatibility

Date and version information

Proposal version

03

Completion date

Stage two proposal sent to DITA TC on 1 February 2019. First revision on 19 February 2018. Second revision on 26 February 2018.

Proposal champion

Kristen James Eberlein, Eberlein Consulting LLC

Links to any previous versions of the proposal

Version 01: https://lists.oasis-open.org/archives/dita/201802/msg00006.html
Version 02: https://lists.oasis-open.org/archives/dita/201802/msg00059.html

Initial suggestion

The stage one proposal was sent to the TC list on 6 June 2017: https://lists.oasis-open.org/archives/dita/201706/msg00016.html. It was discussed and moved to stage two on 6 June 2017.

GitHub issue

https://github.com/oasis-tcs/dita/issues/36

Original requirement or use case

Remove deprecated/reserved for future use items (elements and attributes) from the DITA base

Use cases

Not applicable

New terminology

None

Proposed solution

Modify the grammar files and documentation to remove elements and attributes that have been designated as deprecated, "reserved for future use," or defined by mistake and retained only to maintain backwards compatibility

Benefits

Reduced technical debt

Technical requirements

Figure 1: Element types to be removed

Element type	Module in which the element is defined	Status	Replacement
`<boolean>`	`commonElements.mod`	Deprecated since DITA 1.0	`<state>`
`<indextermref>`	`commonElements.mod`	Reserved for future use	None

Figure 2: Attributes to be removed

Attribute and attribute values	Module in which the attribute is defined	Status	Replacement
`@alt`	`commonElements.mod`	Deprecated since DITA 1.0	`<alt>`
`@chunk="to-navigation`	None	Deprecated since DITA 1.3	None

Attribute and attribute values	Module in which the attribute is defined	Status	Replacement
@collection-type="tree" on <linkpool> and <linklist>	topic.mod	Deprecated. Only present in DTDs, not XSD or RNG.	None
@collection-type on <reltable> and <relcolspec>	The elements are defined in map.mod. Both attribute declaration groups reference the %topicref-atts-no-toc-no-keyscope; entity.	Undefined and reserved for future use	None
@keyref on <navref>	map.mod	"… unintentionally defined for <navref> in the original DITA grammar files. It is retained for backwards compatibility. The attribute will be removed in a future release, and processors are not expected to support it."	None
@locktitle on <topichead> and <topicgroup>	<topichead> and <topicgroup> are defined in mapGroup.mod. In each case, the %topicref-atts;; entity is referenced.	Housekeeping to correct an unintentional mistake; the attribute was present only because attribute definitions were reused between several elements	None
@longdescref on <image>	commonElements.mod	Deprecated since DITA 1.2	<longdescref>
@navtitle	map.mod	Deprecated since DITA 1.2	<navtitle>
@print	map.mod	Deprecated since DITA 1.3	@deliveryTarget
@query	• <topicref>: map.mod • <anchorref>, <mapref>, <topicset>, <topicsetref>, and <keydef>: mapGroup.mod • <link>: topic.mod	Declared but never defined	None
@refcols	The attribute is declared in the %simpletable.attributes; group in commonElements.mod.	Undefined and reserved for future use	None

Attribute and attribute values	Module in which the attribute is defined	Status	Replacement
@role="sample" and @role="external"	%relational-atts; and %rel-atts; in topic.mod	Deprecated since DITA 1.0	None
@title on <map>	mapGroup.mod	Deprecated since DITA 1.1	<title>
@type="internal" and @type="external" on <lq>	commonElements.mod **Note** While @type="internal" is listed as deprecated in the DITA 1.3 spec, it is not a defined attribute value in commonElements.mod	Deprecated since DITA 1.2?	@scope and @format on <lq>

Backwards compatibility

Figure 3: Elements and attributes deprecated in DITA 1.0

- @alt
- <boolean>
- @role="sample" and @role="external"

Figure 4: Elements and attributes deprecated in DITA 1.1

- @title on <map>

Figure 5: Elements and attributes deprecated in DITA 1.2

- @navtitle
- @longdescref on <image>
- @type="internal" and @type="external" on <lq>

Figure 6: Elements and attributes deprecated in DITA 1.3

- @chunk="to-navigation"
- @print

Figure 7: Impact to specialization modules

If implementations have specialized from <boolean> or <indextermref>, they will need to redefine the specialization base.

Figure 8: Impact to constraint modules

The following constraint modules will need to be refactored:

- Constraint modules that include <boolean> or <indextermref> in a content model.
- Constraint modules that list any of the deprecated attributes. This will most likely to be encountered in regard to @alt, @navtitle, and @print.

Migration plan

For most cases, migration could be handled by any of the following methods:

- Search and replace across the body of DITA topics and maps
- Prebuilt XLST scripts

Implementation that use the `@print` attribute will need to do more comprehensive rework of their information architecture. This might include the following:

- Determining values for use with the `@deliveryTarget` attribute
- (Optional) Developing a subjectScheme map to control values for the `@deliveryTarget` attribute
- Developing or modifying DITAval files to include or exclude content tagged with specific values for the `@deliveryTarget` attribute

Costs

This feature will have an impact on the following:

Maintainers of the grammar files

The following files must be modified:

DTDs

- `commonElements.mod`
- `map.mod`
- `mapGroup.mod`
- `topic.mod`

RNG

- `commonElementsMod.rng`
- `mapMod.rng`
- `mapGroupMod.rng`
- `topicMod.rng`

Editors of the DITA specification

The following topics will need to be removed:

- `<boolean>`
- `<indextermref>`

This proposal will affect all of the following topics (and probably others that I missed). The `@navtitle` attribute appears in **MANY** examples.

- Edits to the following element-reference topics:
  - 3.1 Base DITA elements, A to Z
  - 3.2.1.5 `<navtitle>`
  - 3.2.2.1 `<alt>`
  - 3.2.2.17 `<image>`
  - 3.2.2.23 `<lq>`
  - 3.2.2.25 `<object>`

  - 3.2.4.1 `<link>`
  - 3.2.4.2 `<linklist>`
  - 3.2.4.3 `<linkpool>`
  - 3.3.1.5 `<navref>`
  - 3.3.1.1 `<map>`
  - 3.3.1.6 `<reltable>`
  - 3.3.1.10 `<relcolspec>`
  - 3.3.2.4 `<topicgroup>`
  - 3.3.2.5 `<topichead>`
  - 3.4.1.22`<resourceid>`
  - 3.5.1.3 `<hazardsymbol>`
  - 3.6.1.2 `<schemeref>`
  - 3.6.1.15 `<relatedSubjects>`
  - 3.6.2.2 `<topicapply>`
  - 3.10.1.2 Metadata attribute group
  - 3.10.3 Attributes common to many map elements:
  - 3.10.10 Simpletable attribute group
  - 3.10.12 Topicref element attributes group
  - 3.10.13.5.1 Using the -dita-use-conref-target value
  - 3.10.13.12 The `@role` and `@otherrole` attributes
- Edits to the examples in the following element-reference topics
  - 3.3.1.4 `<anchor>`
  - 3.3.2.3`<mapref>`
  - 3.4.1.14 `<metadata>`
  - 3.6.1.1 `<subjectScheme>`
  - 3.6.1.3 `<hasInstance>`
  - 3.6.1.4 `<hasKind>`
  - 3.6.1.5 `<hasNarrower>`
  - 3.6.1.6 `<hasPart>`
  - 3.6.1.7 `<hasRelated>`
  - 3.6.1.8 `<enumerationdef>`
  - 3.6.1.10 `<attributedef>`
  - 3.6.1.14 `<subjectdef>`
  - 3.6.2.1 `<subjectref>`
  - 3.6.2.3 `<topicsubject>`
  - 3.6.2.4 `<topicSubjectTable>`
- Edits to the following appendix topics:
  - "Element-by-element recommendations for translators: Base edition"
- Edits to the following architectural specification topics:
  - 2.2.2.4 DITA map attributes
  - 2.2.2.5.4 Example: How the @cascade attribute functions
  - 2.2.3.6 Scaling a list of controlled values to define a taxonomy
  - 2.2.3.8.1 Example: How hierarchies defined in a subject scheme map affect filtering
  - 2.2.3.8.2 Example: Extending a subject scheme

    - 2.2.3.8.3 Example: Extending a subject scheme upwards
    - 2.2.4.2.1 Conditional processing attributes
    - 2.2.4.4 Cascading of metadata attributes in a DITA map
    - 2.2.4.5 Reconciling topic and map metadata elements
    - 2.2.4.6.1 Cascading of attributes from map to map
    - 2.3.4.9 Processing key references to generate text or link text [`@alt`]
    - 2.4.1.1 Table of contents
    - 2.4.3.4 Conditional processing to generate multiple deliverable types
    - 2.4.5.2 Chunking examples
    - 2.6.3.3 DTD: Coding requirements for element type declarations
    - 2.6.4.3 RELAX NG: Coding requirements for element type declarations
- No changes to the basic information architecture of the specification
- No new terminology

Vendors of tools: XML editors, component content management systems, processors, etc.

If tool vendors have built features around any of the deprecated items, those features will need to be migrated to use alternate markup.

DITA community-at-large

Removing the `@navtitle` and `@print` attributes is likely to have the greatest impact. While DITA maps that contain `@navitle` attributes could be automatically rewritten to use `<navitle>` elements, reworking maps that use the `@print` attribute will require more fundamental architectural rework.

DITA migration procedures or tools

The DITA TC will deliver a separate document that contains migration information.

It is possible that the TC should provide some basic scripts to make migration easier.

Examples

The following table contains examples of code that contains deprecated elements and attributes; it also provides example of how the code could be constructed in DITA 2.0.

Deprecated item	DITA 1.3 markup	DITA 2.0 markup
`@alt`	`<image href="bike.gif" alt="Two-wheeled bicycle"/>`	`<image href="bike.gif" <alt>Two-wheeled bicycle</ alt> </image>`
`<boolean>`	`She said "<boolean state="yes"/>" when I asked her to marry me!`	`<step><cmd>Verify the presence of an "on" or high condition at the input gate (ie, <state name="inflag" value="high"/>)</cmd></step>`
`@print`	`<topicref href="foo.dita" print="no">`	`<topicref href="foo.dita" deliveryTarget="Web-only">`

Deprecated item	DITA 1.3 markup	DITA 2.0 markup
@navtitle	<subjectScheme> <subjectdef keys="os" navtitle="Operating system"> <subjectdef keys="linux" navtitle="Linux"> <subjectdef keys="redhat" navtitle="RedHat Linux"/> <subjectdef keys="suse" navtitle="SuSE Linux"/> </subjectdef> <subjectdef keys="windows" navtitle="Windows"/> <subjectdef keys="zos" navtitle="z/OS"/> </subjectdef> <enumerationdef> <attributedef name="platform"/> <subjectdef keyref="os"/> </enumerationdef> </subjectScheme>	<subjectScheme> <subjectdef keys="os"> <topicmeta> <navtitle>Operating systems</navtitle> </topicmeta> <subjectdef keys="linux"> <topicmeta> <navtitle>Linux</navtitle> </topicmeta> <subjectdef keys="redhat"> <topicmeta> <navtitle>RedHat Linux</navtitle> </topicmeta> </subjectdef> <subjectdef keys="suse"> <topicmeta> <navtitle>SUSE Linux</navtitle> </topicmeta> </subjectdef> </subjectdef> <subjectdef keys="windows"> <topicmeta> <navtitle>Windows</navtitle> </topicmeta> </subjectdef> <subjectdef keys="zos"> <topicmeta> <navtitle>z/OS</navtitle> </topicmeta> </subjectdef> </subjectdef> <enumerationdef> <attributedef name="platform"/> <subjectdef keyref="os"/> </enumerationdef> </subjectScheme>
@longdescref on < image>	<image href="puffin.jpg" longdescref="http://www.example.org/birds/puffin.html"> <alt>Puffin pigure</alt> </image>	<image href="puffin.jpg"> <alt>Puffin pigure</alt> <longdescref href="http://www.example.org/birds/puffin.html" scope="external" format="html"/> </image>
@title on <map>	<map id="mybats" title="Bats> <topicref	<map id="mybats"> <title>Bats</title> <topicref

Deprecated item	DITA 1.3 markup	DITA 2.0 markup
	href="bats.dita" type="topic"> <topicref href="batcaring.dita" type="task"/> <topicref href="batfeeding.dita" type="task"/> <topicref href="batsonar.dita" type="concept"/> <topicref href="batguano.dita" type="reference"/> <topicref href="bathistory.dita" type="reference"/> </topicref> </map>	href="bats.dita" type="topic"> <topicref href="batcaring.dita" type="task"/> <topicref href="batfeeding.dita" type="task"/> <topicref href="batsonar.dita" type="concept"/> <topicref href="batguano.dita" type="reference"/> <topicref href="bathistory.dita" type="reference"/> </topicref> </map>

2.1.11 Stage two: #46: Remove @xtrf and @xtrc

The `@xtrf` and `@xtrc` attributes were added in DITA 1.0 purely for processing purposes, to add a standard way for processors to store mid-conversion debug information while (in theory) retaining validity against an original DTD. This is a legacy concern outside the scope of the standard / outside of the usual interoperability concern; they should be removed as part of the effort to clean up DITA 2.0 and remove obsolete or unnecessary markup.

Date and version information

Date that this feature proposal was completed
January 30, 2018

Champion of the proposal
Robert D Anderson

Links to any previous versions of the proposal

- Stage 1 details stored in tracking issue https://github.com/oasis-tcs/dita/issues/46
- Latest SVN version of the proposal: https://tools.oasis-open.org/version-control/browse/wsvn/dita/trunk/DITA-2.0/stage-2/Issue46-removeXtrfXtrc.dita

Links to e-mail discussion that resulted in new versions of the proposal
N/A

Link to Github issue
https://github.com/oasis-tcs/dita/issues/46

Original requirement or use case

The `@xtrc` and `@xtrf` attributes were added to all elements in DITA 1.0. They are entirely intended for processing purposes, with a DITA-OT style processing model in mind. Specifically, the expectations were that:

1. Assumption (often but not always true): Processors would read/write DITA files as they evaluated features like `@conref` (as DITA-OT has done since early days in the temporary directory).
2. Assumption (often but not always true): Processors needed to a place to track the source each element to enable later debugging (this was done by some processors with `@xtrc` and `@xtrf` even in DITA beta days).

3. Assumption (not a valid assumption!): Each DITA document had to remain valid during every read/write operation in the temp directory, including with this debug information. DITA-OT has never followed this and will not in the future; I only know of one tool in beta DITA days that had this requirement.
4. Result: Because processed DITA with debug info had to be valid, every DITA element needed to declare `@xtrc` and `@xtrf` as part of the standard. (At the time, there was no way to specialize attributes, so they had to be in the standard to be valid.)

The DITA specification (in the last couple of releases) has moved away from adding functionality or markup based on what is done with processed DITA, in particular what might be done by one single processor but not others. Instead it uses DTD / RNG to place rules on the source, but once processors start working with it, something like storing debug information is completely up to the processor.

The standard cannot and should not try to design this aspect of any DITA implementation, particularly a feature like this that presents no interoperability concern. Based on item #3 above, even the implementations that resulted in these attributes consider them obsolete as part of the standard.

Use cases

The use cases for removing these are:

- Cleaner language; when attributes are unnecessary, there is no reason for them to clutter up the language, the specification, and the attribute list within an authoring tool.
- Cleaner implementation; there is not (and cannot be) a requirement for implementations to use these attributes as designed, but their presence forces implementors to consider them and may guide implementations down an unnecessary path.
- The attribute names themselves are a point of confusion when first encountered, adding to DITA's "perception of complexity" problem.
- There is not a requirement for specialized elements to declare these attributes, so the idea that any implementation can assume they are "global attributes" is already false. At the same time, those designing specializations are often afraid to remove them because it's not clear what they're doing (again adding to the complexity problem).

New terminology

N/A

Proposed solution

Remove the `@xtrf` and `@xtrc` declarations from grammar files and from the specification.

Benefits

Who will benefit from this feature?

Authors, information architects working on specializations, DITA implementations, maintainers of the DITA specification.

What is the expected benefit?

Simplified language leads to reduced complexity & less noise in the language.

How many people probably will make use of this feature?

N/A; nearly all audiences will have the *very slight* benefit of not running into or having to think about the attributes at some point.

How much of a positive impact is expected for the users who will make use of the feature?

Minor impact to wide audience.

Technical requirements

Removing elements or attributes

Removing an attribute

- The `@xtrf` and `@xtrc` attributes are currently declared for every element that is formally part of the standard, as part of the `%global-atts;` entity. This entity would be removed from all MOD files that list it (this includes nearly every MOD file that declares elements).

Processing impact

Should not have any processing impact. Tools like DITA-OT that declare this attribute in intermediate files can continue to do so, but this is not required.

Overall usability

N/A

Backwards compatibility

Was this change previously announced in an earlier version of DITA?

No, though it has been discussed infrequently in various forums. (Some have complained at the absurdity of such a processing feature being part of the core specification.)

Removing or renaming an attribute that was shipped in DITA 1.3?

Yes, removing two attributes from every element.

Removing or replacing a processing feature that was defined in DITA 1.3?

Technically yes, removing these attributes is removing a specification-defined method to store debugging information. However, that usage is still allowed by the specification (and could not be prevented in any case).

Migration plan

For documents that may use the attributes:

- A tool could be provided to explicitly scan for and delete any instances of these attributes in source files.
- If any script exists to help migrate documents to DITA 2.0 (as is generally anticipated), this is an easy addition to that tool.

Might any existing processors or implementations need to change their expectations?

Yes, if cases exist where these attributes were not used as intended. Specifically, it's possible that somebody has noticed that these attributes 1) exist on every element, 2) have no meaningful definition with regards to interoperability of source material, and 3) used them as a handy place to drop some other form of annotation or metadata. In such cases, there are a few alternatives:

1. Create an attribute domain that declares `@xtrf` and `@xtrc` as specializations of `@base`, add that to local shells, and continue as-is.
2. Preferably, do the same, but give the attributes names related to their purpose.
3. If "no specialization or extension" is a requirement, use `@base` directly as the generic "extensible for any purpose" attribute. Alternatively, a recent proposal from Eliot for a global metadata attribute (also specializable) may serve the purpose.

Might any existing specialization or constraint modules need to be migrated?

Yes - most existing element specialization modules will need an update. It's possible that constraints will need an update, but much less likely.

The two attributes are added to all OASIS based modules using a consistent pattern for each grammar (always using the `global-atts` name). While this is not required, that pattern is followed in all of the specialization modules I've encountered, and I suspect it is true of most modules:

- In DTDs, the attributes are added with the `%global-atts;` parameter entity:

```
<!ATTLIST  steps        %global-atts;  class CDATA "- topic/ol task/steps ">
```

- In RNG, the attributes are also added with the class attribute:

```
<define name="steps.attlist" combine="interleave">
      <ref name="global-atts"/>
      <optional>
        <attribute name="class" a:defaultValue="- topic/ol task/steps "/>
      </optional>
    </define>
```

- In XSD, the attributes are added with an attribute group:

```
<xs:attributeGroup name="steps.attributes">
      <xs:attributeGroup ref="univ-atts"/>
      <xs:attribute name="outputclass" type="xs:string"/>
      <xs:attributeGroup ref="global-atts"/>
   </xs:attributeGroup>
```

If a utility is created to help scan and update specialized grammar files for DITA 2.0, removing these items is a trivial addition.

If that utility does not exist, removing these references can be done with a search/replace operation that matches the exact sequences above and replaces them with an empty value. I would expect this to work for nearly all specialization modules. If modules exist that independently declare `@xtrf` or `@xtrc`, then the same search-and-replace process can be used with a search for those attribute declarations.

Costs

Maintainers of the grammar files

Minor cost, primarily search and replace, to remove the attribute group and its usage.

Editors of the DITA specification:

How many new topics will be required?

None

How many existing topics will need to be edited?

- The global attribute topic will need to be removed: http://docs.oasis-open.org/dita/dita/v1.3/errata01/os/complete/part1-base/langRef/attributes/debugAttributes.html
- Element definitions that explicitly list and link to these attributes will need to remove the reference (based on DITA 1.3, assumes these elements remain in DITA 2.0): `<attributedef>`, `<dita>`, `<elementdef>`, `<enumerationdef>`, `<no-topic-nesting>`, `<ux-window>`

Will the feature require substantial changes to the information architecture of the DITA specification?

No

f there is new terminology, is it likely to conflict with any usage of those terms in the existing specification?

No

Vendors of tools:

Only cost is to tools that make use of these attributes in ways not intended; see the "migration plan" section above.

DITA community-at-large:

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

No, it should reduce that perception.

Will it be simple for end users to understand?

N/A

If the feature breaks backwards compatibility, how many documents are likely to be affected, and what is the cost of migration?

- Few DITA maps or topics, as this attribute is not intended for use in source documents
- Specialization modules will need to be updated to remove these attributes; this is a straightforward process with easy migration instructions
- Tools that used these attributes in a way not intended may have a higher cost; that cost will depend on how the attributes were used

Producing migration instructions or tools:

How extensive will migration instructions be:

Minimal - full instructions covered above

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, though it would be helpful. Migration tools for source should not be very complex. Migration tools for grammars could be more difficult because there are many ways these could be defined. A good search/replace tool is likely the fastest way to migrate specializations.

Examples

N/A.

If the attribute is used as intended, it will not exist in source files before or after migration.

If the attribute is used in some other way (not as defined in the spec), before/after migration examples will depend on that usage.

2.1.12 Stage two: #73 Remove delayed conref domain

Given the complexity of this feature, lack of implementation support, and lack of evidence that it is actually used by DITA adopters, the TC should remove this feature from DITA 2.0.

Date and version information

- Champion: Alan Houser
- Stage 2 proposal revised 12 February 2018 by Alan Houser (current version)
- Stage 2 proposal completed 6 February 2018 by Alan Houser: https://www.oasis-open.org/apps/org/workgroup/dita/download.php/62461/delayed_conref.html
- Stage 1: Proposed by Robert Anderson on 5 September 2017: https://github.com/oasis-tcs/dita/issues/73
- E-mail discussion: https://lists.oasis-open.org/archives/dita/201708/msg00056.html

Original requirement or use case

As this is a proposal to remove a feature from DITA 2.0 there is no original use case. Proposal is based on the complexity of the feature, lack of implementation support, and lack of evidence that the feature is used.

DITA 1.3 specification: http://docs.oasis-open.org/dita/dita/v1.3/errata01/os/complete/part3-all-inclusive/langRef/containers/delayconref-d.html#delayconref

Use cases

Reduce complexity of DITA 2.0 specification; reduce complexity for implementors.

New terminology

Not applicable

Proposed solution

Removing this feature would require:

- Remove delayedResolution domain content from DITA 2.0 specification.
- Remove delayedResolution domain, and references to the domain, from DITA 2.0 grammar files.

Benefits

The expected benefits of this proposal are:

- Some simplification of the DITA 2.0 specification.
- Simplification for implementers of DITA processing tools and applications.

While there has not been a formal audit of usage of this feature, TC-member perception is that few adopters use this feature.

Current DITA adopters who do use this feature, if any, would presumably need to replace it with some other mechanism before migrating to DITA 2.0.

Technical requirements

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

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

Removing elements or attributes

Removing a domain entirely, or from select document types

"delayResolutionDomain.ent" and "delayResolutionDomain.mod" are to be removed. The domain is included in the basemap DTD shell, and is available in all map types.

Removing an element

All elements are defined in "delayResolutionDomain.mod":
anchorid
anchorkey
exportanchors

Processing impact

Removing this feature should yield zero processing side-effects for processors that do not use this feature. Maintainers of processors that have implemented this feature may choose to remove

support, support the feature as a DITA domain specialization, or migrate support to some other markup.

Overall usability
Near-zero impact to current DITA users who are not using this feature. Small reduction in the content of the specification.

Backwards compatibility

This feature was introduced with DITA 1.2. The feature is bound to the delayed conref resolution domain. Removing this feature will break backwards compatibility *for users of the domain*.

Any current users of this domain could add this domain to their DITA 2.0 shells; if their tools continue to support the markup then the feature will continue to work.

Migration plan

Current users of this feature could add the domain to ther DITA 2.0 shells or implement the feature in some other way.

Costs

- Costs to editors of the DITA grammar files: Removing this feature would require removing references to the delayedConrefResolution domain from the three DITA schema expressions (RNG, DTD, and W3C Schema) and supporting catalog files. Estimated total effort: one hour.
- Costs to editors of the DITA specification: Removing this feature would require removing references to the delayedConrefResolution domain and component elements (exportanchors, anchorid, anchorkey) from specification map files, and removal of references to the three delayed conref resolution elements where mentioned in the specification content (example: Element-by-element recommendations for translators). Estimated total effort: one hour.
- Costs to tools vendors: Zero. DITA-OT/Eclipse is the only known publishing toolchain that supports this feature. No known commercial tools support this feature.
- Costs to DITA community-at-large: Zero cost. Very minor possible positive benefit (admitedly negligible) with respect to simplifying the DITA Specification.
- Migration costs: Current users of this feature (if any) will need to update their DITA 2.0 shells to include the domain or identify/implement an alternate solution.

Examples

Not applicable.

2.1.13 Stage two: #105 Redesign chunking

Simplify how the `@chunk` attribute is defined to 1) make it easier to use, and 2) make implementation easier and more reliable.

Date and version information

Include the following information:

Date that this feature proposal was completed
14 March 2018

Champion of the proposal
Robert D Anderson

Links to any previous versions of the proposal

1. Stage 1 proposal 28 Feb 2018: https://lists.oasis-open.org/archives/dita/201802/msg00106.html
2. Stage 2 proposal 26 March 2018: https://lists.oasis-open.org/archives/dita/201803/msg00071.html

Links to minutes where this proposal was discussed at stage 1 and moved to stage 2
https://www.oasis-open.org/committees/download.php/62726/minutes20180313.txt, with Eliot and Stan as reviewers

Reviewers for Stage 2 proposal
Stan Doherty, Eliot Kimber

Links to e-mail discussion that resulted in new versions of the proposal
Discussed at TC 3 April 2018, resulting in updated Stage 2 proposal

Link to the GitHub issue
https://github.com/oasis-tcs/dita/issues/105

Original requirement or use case

Redesign the chunk attribute for the following reasons / benefits:

- Make it easier to use (rename useful tokens to intuitive values)
- Make it easier to implement (discard operations that are not useful or are edge cases)

Use cases

- Make it easier to use: the current tokens are not obvious. To someone who is not already very familiar with the values, the most common values (`chunk="to-content"` to combine content and `chunk="by-topic"` to split content) are not intuitive and require frequent use of DITA reference documentation. It's also not clear from the value-names whether those values would apply to the referenced document, child documents in a map hierarchy, or both. Replacing these two tokens
- Make it easier to use and implement: the current attribute tries to do too many things at one time, resulting in complex and difficult-to-implement attribute values. The attribute values in DITA 1.3 attempt to do three things at once (select how much of a single document should be published, decide how to combine multiple documents, and decide how to render those combinations). To use these effectively an author must use multiple tokens in the single attribute. All 7 original tokens for these functions are non-obvious, making it very difficult to know which and what combination of these tokens are necessary. Most of the additional behaviors are rarely (if ever used), resulting in little or no benefit from the additional values.
- Make the spec (and implementations) easier to maintain: the original 7 values were defined in a topic on chunking in DITA 1.1. The values were not rigorously explained, and there was no explanation of how they interacted. Later versions of the specification attempted to clarify some of the missing information, but this topic has been very difficult to work with given the need to support all interpretations of what came in with the first version. In addition, implementations have often been unclear about how to handle combinations; using DITA-OT as an example, the need to handle many (often non-sensical) possible combinations has resulted in very complex, error-prone, and hard to maintain code.

New terminology

N/A

Proposed solution

The overall goal with this solution is to preserve (mostly intact) the two most useful existing cases for chunking.

Important

1. The chunking function, as with features like `@conref`, is a DITA-defined operation related to processing DITA documents. As such, the specification can only declare the before and after state of all DITA documents that implement the feature, in the context of processing the documents for some other purpose. For example, a DITA document `many.dita` might be chunked into many topic documents during rendering, but (again like `@conref`) the before/after state still deals with the DITA content. Any examples that make use of published HTML file names are purely for illustration / ease of understanding.
2. Because the chunking operation is defined in terms of processing, the values below are not meant as *tool operations on the source*, such as "refactor my source to reflect these new chunks". The result of evaluating `@chunk` is no longer a source file, and does not need to exist as an actual file (it may be an object in memory somewhere).
3. This entire function is intended for situations where splitting or combining content is relevant, & where authors need control over how that happens. In nearly all cases, chunking will be irrelevant for monolithic publishing formats like PDF or EPUB. Likewise, published HTML is often multi-file and so typically makes use of chunking. However, neither of these is always the case – local style may dictate that PDFs are split at some level, or that HTML is always generated as a single file. As such, we need to be careful that the specification allows `@chunk` to be ignored when needed. This also means that the specification itself cannot know in advance when this is the case or for what formats this is the case – the best we can do is give examples of common cases.

These are the two operations people already think of or look for when they ask about chunking: the ability to publish many documents as if they were one, and the ability to publish one document as if it was many. To that end, the proposed solution is:

1. Remove all of the current `@chunk` token values (one value, `to-navigation`, is already deprecated).
2. Define one new value `combine` to handle the most common scenario, combining multiple DITA documents from a map into one while preserving the overall hierarchy of the map.
    - When specified on a map, it means that all documents referenced by the map should be combined into one DITA document.
    - When specified on a branch of a map, it means that all documents referenced within that branch should be combined into one. This is true regardless of whether the element that specifies `@chunk` refers to a topic or specifies a heading. In cases such as `<topicgroup>` where a grouping element specifies `chunk="combine"`, the result is likely to be a single DITA document with a `<dita>` root element containing peer topics.
    - When `chunk="combine"` is specified on a reference to a map, it indicates that all documents within the scope of the referenced map should be combined into one DITA document.
    - Once `chunk="combine"` is specified on a map, branch, or map reference, all documents in scope are combined into a single resource. Any additional `@chunk` attributes on elements within the hierarchy are ignored. (This is based on a response to my original stage 1 proposal.)

3. Define one new value `split` to handle the second most common scenario of splitting one DITA document into many.
    - When specified on a `<topicref>`, it indicates that all topics *within the referenced document* should be split into multiple documents. ***For example***, in a context where each individual DITA document is published as a single HTML file, specifying `chunk="split"` on a reference to a document that contains five topics will result in 5 documents + 5 output files.
    - When specified on an element such as `<topicgroup>` that does not refer to a topic or have content that is treated as a topic, the value has no meaning.
    - In each of the above cases (`chunk="split"` specified on a `<topicref>` rather than on a map), there is no cascading for the `@chunk` value; if contained topic references do not specify any `@chunk` attributes, they will use whatever default chunking style is in operation for the rest of the map.
    - When specified on the root map, it indicates that `chunk="split"` is the default operation for *all documents in the map, outside the context of relationship tables*. The `split` value is used until / unless a `"combine"` value is encountered, in which case `"combine"` takes over.
    - When specified on a submap, it indicates that `chunk="split"` is the default operation for *all documents within the scope of that map, outside the context of relationship tables*, until / unless a `"combine"` value is encountered, in which `"combine"` takes over.
    - I don't think it makes sense for documents to be split *inside* of a relationship table; doing so would result in far more links than expected. I think that if you do want links between specific topics, those can and should be specified in the source relationship table. Basically, what I'm going for here is "do what people would expect", while also trying not to overcomplicate things.
    - In all cases, when a DITA document is split into multiple documents, the hierarchy of the topics in that document must be preserved in the resulting `<topicref>` hierarchy that references each generated document. Any nested `<topicref>` elements within the original `<topicref>` should be treated as follows:
        a. If the referenced topic document to be split has a root DITA topic, any nested `<topicref>` elements within the original `<topicref>` *SHOULD* be treated as if they are nested directly within the root topic, after any other nested topics. For example, if the document to be split uses the following hierarchy, any nested topic references from the map should appear nested inside of topic "c":

```
<topic id="root"><title>Root topic</title>...
  <topic id="a"><title>First child topic</title>...content, maybe nested
topics</topic>
  <topic id="b"><title>Second child topic</title>...content, maybe
nested topics</topic>
  <topic id="c"><title>Last child topic</title>
    ...content...
    ...maybe nested topic...
    <!-- nested map hierarchy goes here, after (but not within) other
topics -->
  </topic>
</topic>
```

        b. If the referenced topic document to be split has a root `<dita>` element, any nested `<topicref>` elements within the original `<topicref>` *SHOULD* be treated as if they are nested directly within the final child topic of the `<dita>` element, after any other nested topics. For example, if the document to be split

uses the following hierarchy, any nested topic references from the map should appear nested inside of concept "c":

c.
```
<dita>
  <topic id="a"><title>First topic</title>...content, maybe nested
topics</topic>
  <task id="b"><title>Second topic, a task</title>...content, maybe
nested topics</task>
  <concept id="c"><title>Last child topic, a concept</title>
    ...content...
    ...maybe nested topics...
    <!-- nested map hierarchy goes here, after (but not within) other
topics -->
  </concept>
</dita>
```

- In all cases, when a DITA document is split into multiple DITA documents, file names are up to the implementation. (We could suggest that file names be taken from topic IDs, but implementations must be free to choose naming schemes that make sense in their context, and would regardless have to handle conflicting IDs.)

4. When links exist to a topic that is chunked, applications will need to handle the link so that it resolves to the new combined or split context. If a chunking operation results in multiple instances of a result topic (either chunked separately, or some chunked and some not), applications may determine which result topic to target with the link.

   Note As discussed in the TC meeting on 3 April 2018, there is no specification-defined way to associate context-sensitive keys with each nested topic in a document that will be split. This is not a change from previous version of DITA, but was not previously called out in the specification. This means that if a document appears *more than one time* in a map, and is split into multiple documents, there is no standard mechanism to associate keys or URIs with one instance of the nested topic versus another instance of the nested topic. As a result, there is no standard mechanism that allows links or relationship tables to those topics.

   This is only an issue when the document appears more than once in a navigation context, because if the topic only appears once then key or URI based links can be updated as needed. Applications can come up with methods to associate context-sensitive keys or URIs with these topics, although those methods should still follow the normal rules of the DITA language (in other words, they should not create a new mechanism where `@keyref` resolves in a non-standard fashion).

5. This attribute should still be defined as CDATA, which would allow applications to define additional tokens, although I expect those will be rare. One potential advantage to this approach is that DITA 1.x tokens would still remain valid according to the parser (but ignored by 2.0 processors). I propose that we avoid some of the DITA 1.x confusion by stating that the attribute can only contain a single token (note this would mean some potential DITA 1.x values are no longer valid).
6. All remaining behaviors associated with DITA 1.x chunking are no longer supported by this attribute. The original tokens declared several unrelated behaviors using a single attribute. I suggest that *if any of those other behaviors are still required*, alternate attributes be defined to handle them. I do not intend to define those attributes as part of this proposal. That work should only be done if somebody has a strong need for the attributes.

Benefits

Who will benefit from this feature?

1. Authors wishing to combine or split documents
2. Those trying to implement chunking in a processor

3. Maintainers of the DITA specification and of DITA tools who can now provide a clear explanation of the function

What is the expected benefit?

1. Chunking is easier to use
2. Chunking is easier to implement
3. Improved documentation (in the spec and elsewhere)
4. DITA is simplified by making the feature more intuitive and by removing features that are not used + make the simple case difficult

How many people probably will make use of this feature?

Many, based on my own experience and based on the number of open defect reports against DITA-OT chunking

How much of a positive impact is expected for the users who will make use of the feature?

Significant improvement over the current feature

Technical requirements

Renaming or refactoring elements and attributes

Renaming or refactoring an attribute

The current attribute values are not defined in the grammar file, so the grammar definition does not change.

In the specification, the current definition for all 6 valid values and 1 deprecated value will be removed. The will be replaced with the two values "combine" and "split".

This applies to all uses of the `@chunk` attribute; no elements will get the attribute that did not have it before, and no elements that had the attribute will have it removed.

Processing impact

- The `chunk="combine"` attribute value will be equivalent to the current `chunk="to-content"` value when no other chunk tokens are present, so implementations will need to adjust for the new value.
- The `chunk="split"` attribute value will be equivalent to the current `chunk="by-topic"` value when no other chunk tokens are present, so implementations will need to adjust for the new value.
- Removing other tokens, removing the ability to combine unrelated tokens, and defining these new values in a way that is clear & simple will all allow applications to remove the many conditions required to support old values.
- The feature may have an impact on how result documents are named, but that should generally be left up to processors.

Overall usability

The chunking feature today is hard to use and hard to implement. This should address both concerns, resulting in a much more usable experience in all aspects of DITA chunking.

Backwards compatibility

Was this change previously announced in an earlier version of DITA?

No, although I have personally described this in public venues as one feature sure to be redesigned in DITA 2.0.

Changing the meaning of an element or attribute in a way that would disallow existing usage?
Yes; for the most common uses (possibly for the only real-world uses), the migration path is clear.

Migration plan

Documents
The easiest path is likely to use search/replace across DITA maps, and update chunking tokens to use the new value.

Processors
Processors will need to be updated to handle the new tokens. As a first approach they could simply treat the new values the same way as older equivalent values, but I would expect that over time many tools will want to replace older chunking processes with new ones.

Might any existing specialization or constraint modules need to be migrated?
Unlikely, although possible in theory. The only case where this could happen is if a module was designed to explicitly enumerate values for `@chunk`. In that case, the same modules would need to allow for the new tokens.

Costs

Maintainers of the grammar files
N/A

Editors of the DITA specification

- How many new topics will be required? 0
- How many existing topics will need to be edited? Two topics will need significant editing (Using the `@chunk` attribute and chunking examples. We may wish to revise the current definition of `@chunk` in the attribute details topic, although the current definition actually works better for the new design than for the old.
- Will the feature require substantial changes to the information architecture of the DITA specification? No.
- If there is new terminology, is it likely to conflict with any usage of those terms in the existing specification? N/A

Vendors of tools
Tools that implement chunking can take a quick approach (interpret the new values exactly the same as old ones), which should have minimal cost. Alternatively, they may wish to rewrite the chunking process, which will have a larger cost (hard to specify exactly due to widely different tool scenarios).

DITA community-at-large

- This should not add to the perception that DITA is too complex (it should do the opposite).
- It should be simple for end users to understand - that is the primary goal. This should be much simpler than the current design.
- Backwards compatibility: any documents that use chunking today will require migration, but with low cost (search-and-replace which can be done before or after the switch to a fully DITA 2.0 environment).

Producing migration instructions or tools

- Migration instructions (as part of a larger migration document) will be minimal, likely only a few paragraphs with small code examples.
- No independent publication for this feature migration will be needed.
- If other tools exist to migrate DITA documents, this would be an easy addition to those tools, but absent that tool this would be more easily done with search-and-replace routines.

Examples

Figure 9: Creating a single monolithic result document from a root map

```
<map chunk="combine">
   <title>Previously this would have been chunk="to-content"</title>
   <topicref href="hello.dita">...</topicref>
   <topicref href="world.dita">...</topicref>
   ...
</map>
```

Figure 10: Creating multiple result documents from a single document

In the case where `hello.dita` contains 5 topics (either nested or peers within a `<dita>` element), the following markup would result in `hello.dita` being split into 5 individual documents. How the documents are handled at that point is up to the processor (in HTML5 output where one input file generally = one output file, this would turn `hello.dita` into five output files, presumably named after topic IDs within the original document). Note that the `chunk="split"` value has *no impact* on content within the nested reference `notchunked.dita`. In the resulting hierarchy, the reference to `notchunked.dita` should end up as the final nested topic within the root topic of `hello.dita` (when that file has a root topic), or as the final nested topic within the final child of `<dita>` (when that file has a root `<dita>` element).

```
<map>
   <title>Previously this would have used chunk="by-topic"</title>
   <topicref href="hello.dita" chunk="split">
     <topicref href="notchunked.dita"/>
   </topicref>
   <topicref href="world.dita">...</topicref>
   ...
</map>
```

Figure 11: Creating multiple result documents from every source DITA document

In the case where `hello.dita` and `world.dita` *each* contain 5 topics each (either nested or peers within a `<dita>` element), the following markup would result in the two original documents being split into 10 individual documents, with the same handling caveats as above.

```
<map chunk="split">
   <title>Previously this would have used chunk="by-topic"</title>
   <topicref href="hello.dita">
     <topicref href="world.dita"></topicref>
    </topicref>
</map>
```

Figure 12: Explicit example of split topic with resulting hierarchy

Assume the very simple map below with a single topic `simple.dita`, and the contents of `simple.dita` are also shown.

```
<map>
   <title>Very simple "split" example</title>
   <topicref href="simple.dita"/>
</map>

simple.dita:
<topic id="a">
  <title>Root topic</title>
  <body>...</body>
```

```
  <topic id="b">
    <title>Sub-topic</title>
    <body>...</body>
    <topic id="c">
      <title>sub-sub-topic</title>
      <body>...</body>
    </topic>
  </topic>
  <topic id="jumpup">
    <title>another sub-topic</title>
    <body>...</body>
  </topic>
</topic>
```

The document `simple.dita` contains four topics; the chunking operation `split` effectively results in the following map, with each document containing only one topic. For this sample the file names are taken from the topic IDs for clarity but this is not required.

```
<map>
  <title>Very simple "split" example</title>
  <topicref href="a.dita">
    <topicref href="b.dita">
      <topicref href="c.dita"/>
    </topicref>
    <topicref href="jumpup.dita"/>
  </topicref>
</map>
```

Figure 13: "split" when used on a grouping element

Assume the following map, where `chunk="split"` is used on grouping elements:

```
<map>
  <title>Groups are split</title>
  <topicgroup chunk="split">
    <topicref href="ingroup1.dita">...</topicref>
    <topicref href="ingroup2.dita">...</topicref>
  </topicgroup>
  <topichead chunk="split">
    <topicmeta><navtitle>Heading for a branch</navtitle></topicmeta>
      <topicref href="inhead1.dita">...</topicref>
      <topicref href="inhead2.dita">...</topicref>
  </topichead>
</map>
```

- In the case of the `<topicgroup>` element, the `@chunk` value is ignored; it does not cascade, and there is no referenced topic, so it has no effect.
- In the case of the `<topichead>` element, in many applications, the title is equivalent to a single title-only topic. In this case the `@chunk` value also has no effect; if the `<topichead>` is treated as a title-only topic, it cannot be split further, and if it is ignored for the current processing context, it is treated no differently than `<topicgroup>`.

Figure 14: "combine" when used on a grouping element

Assume the following map, where `chunk="combine"` is used on grouping elements:

```
<map>
  <title>Groups are combined</title>
  <topicgroup chunk="combine">
    <topicref href="ingroup1.dita">...</topicref>
    <topicref href="ingroup2.dita">...</topicref>
  </topicgroup>
  <topichead chunk="combine">
    <topicmeta><navtitle>Heading for a branch</navtitle></topicmeta>
      <topicref href="inhead1.dita">...</topicref>
      <topicref href="inhead2.dita">...</topicref>
```

```
  </topichead>
</map>
```

- In the case of the <topicgroup> element, the @chunk value results in a single DITA document that includes the contents of both ingroup1.dita and ingroup2.dita. A literal, DITA-valid file representation of the resulting content would presumably include each of those as peers within a <dita> containter).
- In the case of the <topichead> element, the @chunk value also results in a single DITA document. Again, in many applications, the title is equivalent to a single title-only topic. In that case, a file representation of the resulting content would include the contents of inhead1.dita and inhead2.dita as children of the topic with "Heading for a branch" as the title. If <topichead> is ignored for the current processing context, the result would be the same as with <topicgroup>, where the contents of inhead1.dita and inhead2.dita become peers within a <dita> element..

Figure 15: Edge case: "split" becomes "combine"

Assume the following map, where chunk="split" on the root element means that all topics within this map structure are split by default, but a branch within the map sets chunk="combine":

```
<map chunk="split">
  <title>Split most, but not one branch</title>
  <topicref href="splitme.dita">...</topicref>
  <topicref href="exception.dita" chunk="combine">...</topicref>
  <topicref href="splitmetoo.dita">...</topicref>
</topicref>
```

Assume as well that no other @chunk attributes are specified in this map. The following is true:

1. The document splitme.dita and all documents within that branch will be split apart if they contain more than one topic
2. Because of the chunk="combine" setting, the second branch with exception.dita at the root will result in a single result document
3. The document splitmetoo.dita and all documents within that branch will be split apart if they contain more than one topic

Figure 16: Edge case: ignoring "split" values within a combined branch

Assume the following map, where a branch is combined, but a nested <topicref> specifies "split":

```
<map>
  <title>Ignoring split value</title>
  <topicref href="bigBranch.dita" chunk="combine">
    <topicref href="iamhappy.dita"/>
    <topicref href="iamconfused.dita" chunk="split"/>
    <topicref href="happyagain.dita"/>
  <topicref>
  ...
</map>
```

In this case:

1. The branch beginning with bigBranch.dita results in a single, combined document
2. In the combined document, the contents of iamhappy.dita, iamconfused.dita, and happyagain.dita are all peers within the final topic of bigBranch.dita
3. The chunk="split" value within the branch is ignored

2.1.14 Stage two #107: Add and

Proposal for the inclusion of the and elements under a new domain, while redefining the <b> and <i> elements in a more semantic manner.

Date and version information

Date that this feature proposal was completed
8 October 2018

Champion of the proposal
Keith Schengili-Roberts

Links to any previous versions of the proposal
https://lists.oasis-open.org/archives/dita/201803/msg00012.html

Links to minutes where this proposal was discussed at stage 1 and moved to stage 2
https://www.oasis-open.org/apps/org/workgroup/dita/download.php/62726/minutes20180313.txt

Links to e-mail discussion that resulted in new versions of the proposal
https://lists.oasis-open.org/archives/dita/201808/msg00058.html

Link to the GitHub issue
https://github.com/oasis-tcs/dita/issues/107

Original requirement or use case

Many new people coming to DITA have expressed confusion as to the supposed semantic nature of DITA when they notice the existence of the <b> (bold) and <i> (italics) elements.

HTML has long supported (since the "HTML+" specification from 1993) the additional and elements as more descriptive, semantic equivalents for <b> and <i>. In late 2014 the HTML5 standard took this one step further by fully defining <b> and <i> as semantic elements, distinct from and .

In keeping with HTML5, a standard that many coming to DITA have more than a passing familiarity with, this proposal suggests that and be added as elements under a new domain called "emphasis". At the same time, the existing <b> and <i> elements within the highlighting domain will be re-defined within the DITA 2.0 specification in a more semantic manner. This will bring them more in-line with their equivalent elements in HTML5; they are otherwise unchanged.

Use cases

For users seeking a semantic equivalent for the <b> and <i> elements, and could now be used instead.

The retention and redefining of the <b> and <i> elements would also make it clear as to the situations for which and should be used, and the scenarios where <b> and <i> are more appropriate.

New terminology

The element would inherit from topic/ph emphasis, and could be defined as follows:

“The element can be used to indicate content that is considered to be important, serious, or has some form of urgency (without being a specific warning). Typically, its content will be rendered in boldface at output. Use this element only when a more semantically appropriate element is not available. For example, for a specific warning, consider using an appropriate element from the hazard statement domain, such as <hazardstatement>.”

The element would also inherit from topic/ph emphasis, and could be defined as follows:

“The element can be used to indicate emphasis. A stress emphasis is designed to change the meaning of a phrase or sentence, or stressing the importance of a particular noun, verb or adjective. Typically, its content will be rendered in italics at output. Use this element only when a more semantically appropriate element is not available. For example, when indicating a different mood or voice, the <i> element may be more relevant.”

The <b> element description would also change, making it more semantically descriptive, and aligning with its equivalent in HTML5. This could look like the following:

“The <b> element should be used to draw attention to a word or phrase for utilitarian purposes without implying that there is any extra importance. There is also no implication of an alternate voice or mood, or that its content should be actionable. For example, it can be used to indicate product names within a review, highlighting roles within a process, or for use in spans of text where the typical presentation is expected to be in a boldface.”

Similarly, the <i> element would also be redefined to make it more semantically descriptive, and aligning with its equivalent in HTML5. It could look like the following:

“The <i> element should be used for a word or phrase indicating either an alternate voice or mood, or to otherwise offset it from the content around it to indicate a different quality of text, such as a taxonomic designation, an idiomatic phrase from another language, technical term, or a ship name.”

Proposed solution

1. Create a new emphasis domain.
2. Create two new phrase-level elements within this domain: and .
3. Add new descriptions plus example code illustrating the intended usage for these elements.
4. Change the descriptions for the <b> and <i> elements within the highlighting domain and include example code illustrating their intended usage.

Benefits

Who will benefit from this feature?

Authors seeking a more semantic element for encapsulating content that should either be or emphasized. The redefinition of the <b> and <i> elements will also make it plain when and where these highlighting elements should be used. It will also benefit DITA trainers who will now be able to point to more semantic equivalents to the existing <b> and <i> elements.

What is the expected benefit?

Authors working within DITA will have a more clear-cut choice on when to use and , and when to use <b> and <i>, in keeping with how these elements are currently defined within HTML5.

How many people probably will make use of this feature?

There are cases where technical writing teams have constrained out the highlighting domain because of its lack of semantic elements. Similarly, there are DITA authoring groups that have either specialized <ph> to create their own equivalent of and , or, more awkwardly, use @outputclass with <ph> to achieve the same ends. The redefinitions proposed for <b> and <i> may convince the former to retain the highlighting domain, while providing the new, semantically-described and elements ought to take care of the latter group.
While this proposal is not sufficient to draw people to use DITA 2.0, it will likely be welcomed by the user community.

How much of a positive impact is expected for the users who will make use of the feature?

Likely minimal; in many ways this is less a feature than a long-overdue tweak to the specification. However, those who will use this feature are likely to be pleased with its addition.

Technical requirements

Adding new elements or attributes

Two new elements, `<strong>` and `<em>`, will be added under a new domain.

Adding a domain

The new `emphasis` domain would fall under the set of general-purpose Domain elements. It is possible that other elements may fit into this domain in the future; for example, active-low signals in electrical engineering/semiconductor documentation are typically rendered with an overline, indicating logical negation. This is currently handled using the `<overline>` element from the highlight domain. Similarly, instead of using subscript letters to indicate voltage nodes, these could be more specifically and semantically described within the emphasis domain, which can then be formatted according to the style for that industry sector.

Adding an element

Two new elements will be added under the emphasis domain: `<strong>` and `<em>`.

Inheritance:

```
+ topic/ph emphasis/strong
+ topic/ph emphasis/em
```

DTDs:

(Please note that the following is based on DITA 1.3 and does not include any proposed changes for phrase-level elements that may have already been proposed for DITA 2.0).

```
<!ENTITY % emphasis-d-ph
"strong | em"
>

<!-- ============================================================= -->
<!--                    DOMAIN ENTITY DECLARATION                  -->
<!-- ============================================================= -->

<!ENTITY emphasis-d-att
"(emphasis-d-ph)"
>

<!-- ============================================================= -->
<!--                  ELEMENT NAME ENTITIES                        -->
<!-- ============================================================= -->
<!ENTITY % strong       "strong"                                    >
<!ENTITY % em           "em"                                        >


<!--                    LONG NAME: Strong                           -->
<!ENTITY % strong.content
"(#PCDATA |
%basic.ph; |
%data.elements.incl; |
%draft-comment; |
%foreign.unknown.incl; |
%required-cleanup;)*"
>
<!ENTITY % strong.attributes
"%univ-atts;
outputclass
CDATA
#IMPLIED"
>
<!ELEMENT  strong % strong.content;>
<!ATTLIST  strong % strong.attributes;>
```

```
<!--                    LONG NAME: Em                                -->
<!ENTITY % em.content
"(#PCDATA |
%basic.ph; |
%data.elements.incl; |
%draft-comment; |
%foreign.unknown.incl; |
%required-cleanup;)*"
>
<!ENTITY % em.attributes
"%univ-atts;
outputclass
CDATA
#IMPLIED"
>
<!ELEMENT  em % em.content;>
<!ATTLIST  em % em.attributes;>
```

Renaming or refactoring elements and attributes

Only the description of the <b> and <i> elements need to be updated in the DITA 2.0 specification. See the "New Terminology" section for the proposed changes in wording.

Renaming or refactoring an attribute

N/A

Removing elements or attributes

N/A

Processing impact

Expected to be minimal.

Overall usability

Users will have a choice between using and vs. the <b> and <i> elements. There may be some confusion as to when to best use vs. <b> and vs. <i>, but this can be mitigated by providing numerous, relevant code examples in the specification for each element.

Backwards compatibility

Changing the meaning of an element or attribute in a way that would disallow existing usage?

As the <b> and <i> elements are not being removed, going forward DITA 2.0 users can continue to use these elements if they choose, opt to use and as their replacements, or to use both sets of elements in parallel.

Migration plan

Might any existing documents need to be migrated?

Use of and is optional as <b> and <i> are still present, so there is no need to update all instances of <b> to , and <i> to , though there will undoubtedly be some technical documentation teams that choose to do so.

Might any existing processors or implementations need to change their expectations?

Not in terms of expectations, though output processors (such as the DITA-OT) will need to accommodate the formatting of the two new elements, though for compatibility it is suggested that copies the default output behavior of <b>, and that copies that of <i>.

Might any existing specialization or constraint modules need to be migrated?

Groups that have previously constrained out the highlighting domain, or who have specialized <ph> for creating equivalents for and , are likely to drop their modifications with this

proposal. There may still be groups that choose to constrain out the highlighting domain despite the revised semantic descriptions for <b> and <i>, but if so that would be their choice.

Costs

DITA community-at-large

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

Maintainers of the grammar files
Minor cost in adding the new domain and its associated elements.

Editors of the DITA specification: How many new topics will be required?
Three. One to describe the intent of the new emphasis domain, and one for each new element (and). There also ought to be changes to the default document shells to include references to the new domain.

Editors of the DITA specification: How many existing topics will need to be edited?
Two. The topics for <b> and <i> ought to be updated to be more semantically descriptive, which will align them with their equivalent elements in HTML5.

Will the feature require substantial changes to the information architecture of the DITA specification? If so, what?
Only the addition of the new domain. Other than that, no significant architectural change is required. A possible advantage of having a new emphasis domain is that it opens the possibility to include other, more semantically-descriptive elements that lend themselves to specific formatting styles. For example, active-low signals in electrical engineering/semiconductor documentation are typically rendered with an overline, indicating logical negation. This is currently handled using the <overline> element from the highlight domain. Similarly, instead of using subscript letters to indicate voltage nodes, these could be more specifically and semantically described within the emphasis domain, which can then be formatted according to the style for that industry sector.

Vendors of tools
Low cost is expected. Again, this is less a significant new feature than an overdue "tweak".

Will this feature add to the perception that DITA is becoming too complex?
Any additional element adds to the total number of elements available in DITA. However, the intent is to bring DITA more in line with current HTML5 practice, something that will likely be welcomed by the community.

Will it be simple for end users to understand?
Yes. As mentioned earlier, this is less of a wholly new feature than a long-overdue tweak. It seems likely that the community is likely to embrace these new tags, along with the alignment with their equivalents in HTML5.

Producing migration instructions or tools
If there are teams that decide to migrate all instances of <b> and <i> to and , there are already tools capable of doing this one-for-one switch. It is unlikely that there will be new tools needed to do this.
A white paper to describe the correct usage of the new and revised elements would be overkill, especially if sufficient code examples explaining the context for usage are provided within the specification.

If there is new terminology, is it likely to conflict with any usage of those terms in the existing specification?

The new definitions for and , plus <b> and <i>, will make it clear as to their scenarios for use, along with a good set of code examples to demonstrate best practices for when they should be used.

Examples

(The following is a draft description of the element intended for use in the DITA 2.0 specification. It includes several examples).

The element can be used to indicate content that is considered to be important, serious, or has some form of urgency (without being a specific warning). Typically, its content will be rendered in bold at output.

Examples

The following examples show how it can be used.

Emphasizing an important detail:

```
<p>Your doctor prescribed this medicine to treat an infection. It is important that you
<strong>take all of
the medicine</strong> as described.</p>
```

Another example:

```
<p>When starting a car with a keyless ignition, you must <strong>step on the brake pedal</
strong> before
pressing the start button.</p>
```

Underscoring a serious point:

```
Use the word <em>very</em> <strong>sparingly</strong>. Where emphasis is necessary, use words
strong in
themselves.
```

Pointing out a critical/urgent detail:

```
<p>SERVICE HEADLIGHT—<strong>Black</strong> wire with <strong>red tracer</strong> from
handlebar toggle switch
to large terminal screw; <strong>red</strong> wire with <strong>yellow tracer</strong> from
handlebar toggle
switch to small terminal screw.</p>
```

This element is part of the emphasis domain. The addition of this element brings DITA more into alignment with its equivalent in the current HTML specification.

(The following is a draft description of the element intended for use in the DITA 2.0 specification. It includes several examples).

The element can be used to indicate emphasis. A stress emphasis is designed to change the meaning of a phrase or sentence, or stressing the importance of a particular noun, verb or adjective. Typically, its content will be rendered in italics at output. Use this element only when a more semantically appropriate element is not available. For example, when indicating a different mood or voice, the <i> element may be more relevant.

Examples

The following examples show how it can be used.

Emphasizing meaning within a sentence:

```
<p>What was previously called <em>block-level</em> content up to HTML 4.1 is now called
<em>flow</em> content in HTML5.</p>
```

Stressing the importance of a noun within a sentence:

```
<p>A <em>condenser</em> is an apparatus for condensing a large quantity of electricity
on a comparatively small surface.</p>
```

Stressing the importance of a verb or actions within a sentence:

```
To remove a message from a pigeon, first <em>catch</em> the bird, then <em>hold</em> it
in one hand, <em>extend</em> its leg, and <em>remove</em> the message holder with the other
hand.
```

Stressing the importance of an adjective or adjectival phrase within a sentence:

```
<p>A good plan once adopted and put into execution <em>should not be abandoned</em> unless it
becomes clear that it can not succeed.</p>
```

This element is part of the emphasis domain. The addition of this element brings DITA more into alignment with its equivalent in the current HTML specification.

(The following is a draft description of the <b> *element intended for use in the DITA 2.0 specification. It includes several examples).*

The <b> element is used to draw attention to a word or phrase for utilitarian purposes without implying that there is any extra importance. There is also no implication of an alternate voice or mood, or that its content should be actionable. For example, it can be used to indicate product names within a review, highlighting roles within a process, or for use in spans of text where the typical presentation is expected to be in a boldface.

Examples

The <b> element can be used to indicate a product name within a review:

```
<p>One of the best features of <b>Mr. Flip-it</b> is its ability to manipulate objects within
a
three-dimensional space so that you can see the other side.</b>
```

The <b> element can be used to highlight related concepts within a topic:

```
<p>The <b>Solid Waste Operations Manager</b> plans and manages the countywide transfer
station and
landfill operations, coordinates solid waste processing operations with the planning and
engineering staff, and performs related duties as required.</p>

[...lots of intervening text]

<p>The <b>Sanitation Engineer</b> creates strategies for landfill sites that minimize the
impact on the environment.</p>
```

The <b> element can also be used in situations where boldfaced text is expected for stylistic purposes, such as when the house style for an article lede is to be rendered in boldface:

```
<p><strong>Know where to get help.</strong> Before proceeding to wrangle your first ostrich,
ensure
you know the location of the closest first aid station.</p>
```

The redefining of this element brings DITA more into alignment with the equivalent element in the current HTML specification.

(The following is a draft description of the `<i>` *element intended for use in the DITA 2.0 specification. It includes several examples).*

The `<i>` element is used to indicate either an alternate voice or mood, or to otherwise offset it from the content around it to indicate a different quality of text, such as a taxonomic designation, an idiomatic phrase from another language, technical term, or a ship name.

Examples

The `<i>` element can be used for indicating text in a different voice, such as when foreign words or phrases are used:

```
<note type="caution">Even highly experienced operators of heavy machinery should remain alert
for dangerous situations. Having a <i>laissez-faire</i> attitude is a recipe for
disaster.</note>
```

The `<i>` element can also be used to indicate different character voices:

```
<p><i>Edgar</i>: I know thee well—a serviceable villain, as duteous to the vices of thy
mistress
as badness would desire.</p>
```

```
<p><i>Gloucester</i>: What, is he dead?</p>
```

It can also be used to indicate a taxonomic designation:

```
<p>When wrangling ostriches (<i>Struthio camelus</i>) people are advised that while they are
a type of bird
(Class: <i>Aves</i>), they are thought to be descendants of their extinct dinosaur (Suborder:
<i>Theropoda</i>) relatives and share the same type of temperament.</p>
```

The `<i>` element can also be used to designate the name of a ship:

```
<p>The MV <i>Rena</i> was a container ship that ran aground near Tauranga, New Zealand,
resulting in an
oil spill.</p>
```

It can also be used to indicate a new or technical term the first time it is introduced:

```
<p>Immediately prior to undergoing an MRI, a doctor may inject a contrast agent called the
<i>gadolinium
contrast medium</i> into the patient. This 'dye' highlights the part of the body being
scanned and can
provide more information to the radiologist who is assessing the patient's problem.</p>
```

The redefining of this element brings DITA more into alignment with the equivalent element in the current HTML specification.

2.1.15 Stage two: #164 Redesign <hazardstatement>

Redesign the hazard statement domain to better support current standards, authoring requirements, and rendering requirements.

Date and version information

This proposal contains the following information:

Date that this feature proposal was completed
21 September 2019

Champion of the proposal
Kristen James Eberlein, Eberlein Consulting LLC

Links to any previous versions of the proposal

16 September 2019

This version of the stage two proposal, dated 18 September 2019, contains the following changes from the version submitted to the DITA TC on 16 September 2019:

- Acknowledgment of feedback from Amber Swope, Individual member
- Acknowledgment of review from Dawn Stevens, Comtech Services, Inc.
- Correction of a typographic error and a markup error
- Added the following sentence to the "Use cases" section: “Enabling hazard images to be associated with `<messagepanel>` and its child elements makes it possible to produce grouped safety messages that comply with ANSI Z535.6.”

Links to minutes where this proposal was discussed at stage 1 and moved to stage 2
20 August 2019

In addition, this proposal was discussed (requests for early feedback) at the following TC meetings:

- 03 September 2019
- 10 September 2019

Reviewers for Stage 2 proposal

This proposal was reviewed by the following voting TC members

Robert Anderson, IBM
Eliot Kimber, Individual member
Scott Hudson, The Boeing Company
Dawn Stevens, Comtech Services, Inc.

In addition, the following individuals were asked for feedback on this proposal:

Jang Graat, Former individual member (sent personal e-mail approving the proposal)
Amber Swope, Individual member (sent personal e-mail approving the proposal)

Links to e-mail discussion that resulted in new versions of the proposal
Not applicable

Link to the GitHub issue
Issue #64

Original requirement or use case

In May 2018, the following TC members suggested changes to the hazard statement domain for DITA 2.0:

- Jang Graat, Individual member (no longer an OASIS member)
- Dawn Stevens, Comtech Services, Inc.
- Amber Swope, Individual member

All three TC members were motivated to improve the hazard statement domain by increasing its alignment with various safety standards; in addition, they wanted to make it easier for content developers to use and better able to support rendering requirements.

In September 2019, after a careful study of existing safety standards and the changes suggested by DITA TC members and the larger DITA community, I presented various options to the TC and developed consensus on the solution that is outlined in this proposal.

Use cases

This proposal will enable hazard statements authored in DITA to meet the following objectives:

Better comply with the ANSI Z535.6 standard

While the ANSI Z535.6 standard provides great flexibilty for companies to implement its recommendations, it does provide strict requirements for safety alert symbols and signal words.

ANSI Z535.6 defines a *signal word* as a “word that calls attention to a safety message or messages or a property damage message or messages, and designates a degree or level of hazard seriousness.” The standard only permits four signal words: DANGER, WARNING, CAUTION, and NOTICE.

The `@type` attribute on the `<hazardstatement>` element specifies the signal word. For DITA 1.2 and 1.3, the values of the `@type` attribute are identical to those permitted on `<note>`. That allows a lot of values that are not germane for hazard statements and so can introduce authoring dissonance.

Improve the semantic nature of the specialization base

Currently, `<hazardstatement>` is specialized from `<ul>`, and its child elements are specialized from `<li>`. This specialization hierarchy does not make sense.

Enable hazard images to be associated with <messagepanel> and its child elements

Currently, the hazard images are part of the content model for the `<hazardstatement>` element. This presents a problem when the `<hazardstatement>` element contains multiple `<messagepanel>` elements; there is no way to determine which hazard images are associated with specific message panels – nor can it be determined *what* the hazard image represents: `<typeofhazard>`, `<consequence>`, or `<howtoavoid>`.

Enabling hazard images to be associated with `<messagepanel>` and its child elements makes it possible to produce grouped safety messages that comply with ANSI Z535.6.

Simplify the authoring experience

Currently, the fixed order of the child elements within `<messagepanel>` forces some authors to author content in a different order than it will be rendered and read by consumers. This can be counter intuitive.

Also, often authors must use a value for the `@outputclass` attribute in order to indicate how a list in the `<howtoavoid>` element should be rendered. This makes the authoring process harder than it needs to be.

Simplify the work of CSS and stylesheet developers

Many implementations must use transformations, CSS, and stylesheets to swap the order of the `<consequence>` and `<howtoavoid>` elements. This can be burdensome to small implementations, and it can easily be avoided by relaxing the content model of `<messagepanel>`.

Currently, many implementations rely on CSS and style sheets to render the simple list allowed within `<howtoavoid>` as either an ordered or unordered list. Allowing `<ol>` and `<ul>` within `<howtoavoid>` will eliminate this extra work.

New terminology

Not applicable

Proposed solution

- Restrict the values of the `@type` attribute on `<hazardstatement>` to danger, caution, warning, notice, and -dita-use-conref-target
- Require the `@type` attribute on `<hazardstatement>`
- Improve the element-reference topic for `<hazardstatement>` to include definitions of the values for the `@type` attribute that match those in the ANSI Z535.6 standard. The following definition descriptions are taken verbatim from ANSI Z535.6:

  DANGER
  Indicates a hazardous situation that, if not avoided, will result in death or serious injury. This signal word is to be limited to the most extreme situations.

  WARNING
  Indicates a hazardous situation that, if not avoided, could result in death or serious injury.

  CAUTION
  Indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.

  NOTICE
  Indicates information considered important, but not hazard-related (e.g. messages relating to property damage).

  Note that the DITA spec will not use these definitions verbatim. They will be edited to match our style guidelines.
- Expand the content model of `<messagepanel>` to enable greater flexibility in the order of the child elements. While `<typeofhazard>` will remain required and the first child of `<messagepanel>`, it now can be followed by either of the following:
  - `<consequence>` (zero or more), `<howtoavoid>` (one or more)
  - `<howtoavoid>` (one or more), `<consequence>` (zero or more)
- Expand the content model of `<howtoavoid>` to permit `<ol>` and `<ul>`, in addition to `<sl>`
- Change the specialization base of `<messagepanel>`, `<typeofhazard>`, `<consequence>`, and `<howtoavoid>` to `<div>`
- Add `<hazardsymbol>` to the content models of `<messagepanel>`, `<typeofhazard>`, `<consequence>`, and `<howtoavoid>`
- Remove `<hazardsymbol>` from the content model of `<hazardstatement>`

Benefits

This proposal addresses the following questions:

Who will benefit from this feature?

The following will benefit from this redesign:

- All DITA implementations that use the hazard statement domain
- DITA implementations that do not use the hazard statement domain due to concerns that the hazard statement domain is inadequate, difficult to use, or difficult to render
- Companies whose documentation requires hazard statements but who have not migrated to DITA due to concerns that the hazard statement domain is inadequate, difficult to use, or difficult to render

What is the expected benefit?
A hazard statement domain that is easier to use and and easier to render

How many people probably will make use of this feature?
Unknown

How much of a positive impact is expected for the users who will make use of the feature?
Significant

Technical requirements

This proposal involves the following changes:

Refactoring elements

- `<hazardstatement>`
  - Remove several values for the `@type` attribute
  - Make the `@type` attribute required
  - Remove `<hazardsymbol>` from the content model
- `<messagepanel>`
  - Modify content model to enable better flexibility regarding the order of elements
  - Add `<hazardsymbol>` (zero or more) to the content model
  - Change specialization base to `<div>`
- `<typeofhazard>`
  - Add `<hazardsymbol>` (zero or more) to the content model
  - Change specialization base to `<div>`
- `<consequence>`
  - Add `<hazardsymbol>` (zero or more) to the content model
  - Change specialization base to `<div>`
- `<howtoavoid>`
  - Add `<ol>` and `<ul>` to the content model
  - Add `<hazardsymbol>` (zero or more) to the content model
  - Change specialization base to `<div>`

Processing impact
Not applicable

Overall usability
Improved usability for current and future users.

Backwards compatibility

This proposal addresses the following questions:

Was this change previously announced in an earlier version of DITA?
No

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?

No

Changing the meaning of an element or attribute in a way that would disallow existing usage?

Yes – Removing values for the `@type` attribute on `<hazardstatement>` that were allowed in DITA 1.2 and 1.3

Changing a content model by removing something that was previously allowed, or by requiring something that was not?

Yes – Removing `<hazardsymbol>` from the content model of `<hazardstatement>`

Yes – Making the `@type` attribute required on `@hazardstatement`

Changing specialization ancestry?

Yes – As follows:

Element	DITA 1.3	DITA 2.0
<consequence>	"+ topic/li hazard-d/consequence "	"+ topic/div hazard-d/ consequence "
<howtoavoid>	"+ topic/li hazard-d/howtoavoid "	"+ topic/div hazard-d/ howtoavoid "
<messagepanel>	"+ topic/ul hazard-d/messagepanel "	"+ topic/div hazard-d/ messagepanel "
<typeofhazard>	"+ topic/li hazard-d/typeofhazard "	"+ topic/div hazard-d/ typeofhazard "

Removing or replacing a processing feature that was defined in DITA 1.3?

No

Are element or attribute groups being renamed or shuffled?

No

Migration plan

This proposal addresses the following questions:

Might any existing documents need to be migrated?

Yes, implementations will need to rework `<hazardstatement>` elements to relocate `<hazardsymbol>` elements.

Yes, if existing DITA topics use values of `@type` on `<hazardstatement>` that are removed.

Yes, if existing DITA topics contain `<hazardstatement>` elements that do not set the `@type` attribute.

Might any existing processors or implementations need to change their expectations?

Authoring tools might need to change the CSS or XSLT that is used to render hazard statements.

Implementations might need to adjust their stylesheets for rendering output.

Might any existing specialization or constraint modules need to be migrated?
(Unlikely) If an implementation has specialized elements from the hazard statement domain, it might need to modify their specialization modules.

If no migration need is anticipated, why not?
Not applicable

Costs

This proposal has a (time and effort) impact on the following groups:

Maintainers of the grammar files

- (DTD) Edits to `hazardstatementDomain.mod` and `hazardstatementDomain.ent`
- (RNG) Edits to `hazardstatementDomain.rng`

Editors of the DITA specification

The changes to the DITA specification will be minor to medium:

- Editorial changes to the `<hazardstatement>` topic to specify values for `@type`
- Updates to the "Specialization hierarchy" sections for the `<messagepanel>`, `<typeofhazard>`, `<consequence>`, and `<howtoavoid>` topics
- Changes to the "Example" sections in the `<hazardstatement>`, `<messagepanel>`, `<typeofhazard>`, `<consequence>`, and `<howtoavoid>` topics
- Updates to the "Usage information" section in the `<messagepanel>` topic, to explain meaning of `<hazardsymbol>` as direct child of `<messagepanel>`

The entire topic collection for the hazard statement domain is overdue for an edit, but that should be handled separately from this proposal. We should consider providing more information about formatting expectations for elements in the hazard statement domain.

No changes to the information architecture or specification terminology will be required.

Vendors of tools

Authoring tools that render the `<hazardstatement>` element will need to modify the CSS or XSLT that renders the element.

Applications that style the `<hazardstatement>` element in output formats will need to modify their stylesheets.

DITA community-at-large
The changes to the hazard statement domain should not add to a perception of DITA complexity. The changes should be simple and intuitive for end users to understand.

Producing migration instructions or tools

Migration instructions will be straight forward.

Examples

This section provides examples of the new markup.

Figure 17: Loosening content model of <messagepanel> and <howtoavoid>

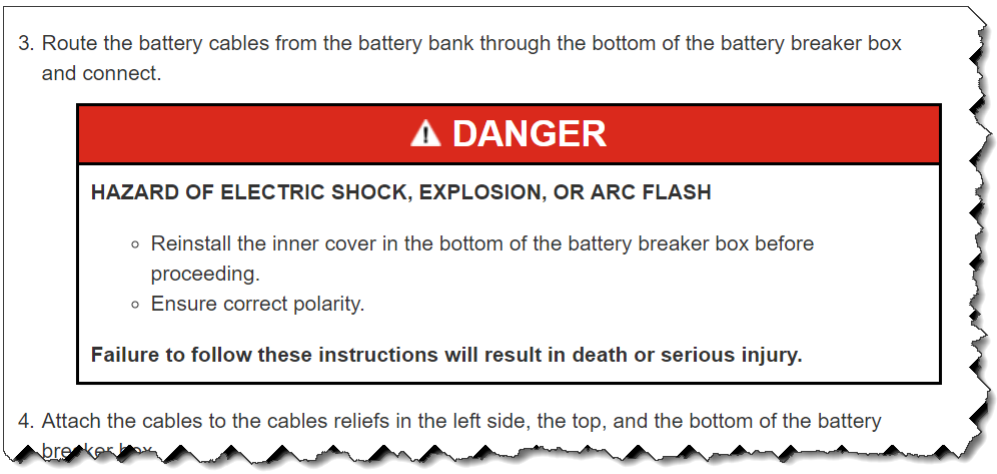
The following code sample illustrates the following:

- Increased flexibility for the content model of `<messagepanel>`

- Use of an unordered list within the `<howtoavoid>` element

```
<hazardstatement type="danger">
  <messagepanel>
    <typeofhazard>HAZARD OF ELECTRIC SHOCK, EXPLOSION, OR ARC FLASH</typeofhazard>
    <howtoavoid>
      <ul>
        <li>Reinstall the inner cover in the bottom of the battery breaker box
            before proceeding.
        </li>
        <li>Ensure correct polarity.</sli>
      </ul>
    </howtoavoid>
    <consequence>Failure to follow these instructions will result in death or serious
                 injury.
    </consequence>
  </messagepanel>
</hazardstatement>
```

This markup might be rendered as the following:



This is an actual screen capture from the Schneider Electric installation instructions for Gallery XV Battery Breaker Box, taken in August 2019.

Note that the company currently must override the HTML transformation to get the desired rendering; the changes that we are implementing for DITA 2.0 will make that override unnecessary.

Figure 18: Adding <hazardsymbol> to <typeofhazard>, <consequence>, and <howtoavoid>

The following code sample illustrates how a company could use a `<hazardstatement>` element to generate what ANSI Z535.6 calls a "grouped safety message":

```
<hazardstatement type="warning">
  <messagepanel>
    <typeofhazard>
      <hazardsymbol keyref="electric-shock-hazard"/>
      ELECTRIC SHOCK HAZARD</typeofhazard>
      <consequence>The equipment must be grounded. Improper grounding, setup, or usage of
                   the system can cause electric shock
      </consequence>
      <howtoavoid>
        <hazardsymbol keyref="ground-power-source"/>
        <ul>
          <li>Turn off and disconnect power at main switch before disconnecting any
              cables or before servicing or installing any equipment.</li>
          <li>Connect only to grounded power sources.</li>
          <li>All electric wiring must be done by a qualified electrician and comply
```

```
              with all local codes and regulations.</li>
        </ul>
      </howtoavoid>
    </messagepanel>
    ...
    <messagepanel>
      <typeofhazard>
        <hazardsymbol keyref="burn-hazard"/>
        BURN HAZARD</typeofhazard>
        <consequence>Electric surfaces and fluid that's heated can become very hot during
                    operation.</consequence>
        <howtoavoid>
          To avoid burns:
          <ul>
            <li>Do not touch hot fluid or equipment.</li>
          </ul>
        </howtoavoid>
  </messagepanel>
</hazardstatement>
```

This markup might be rendered as the following:

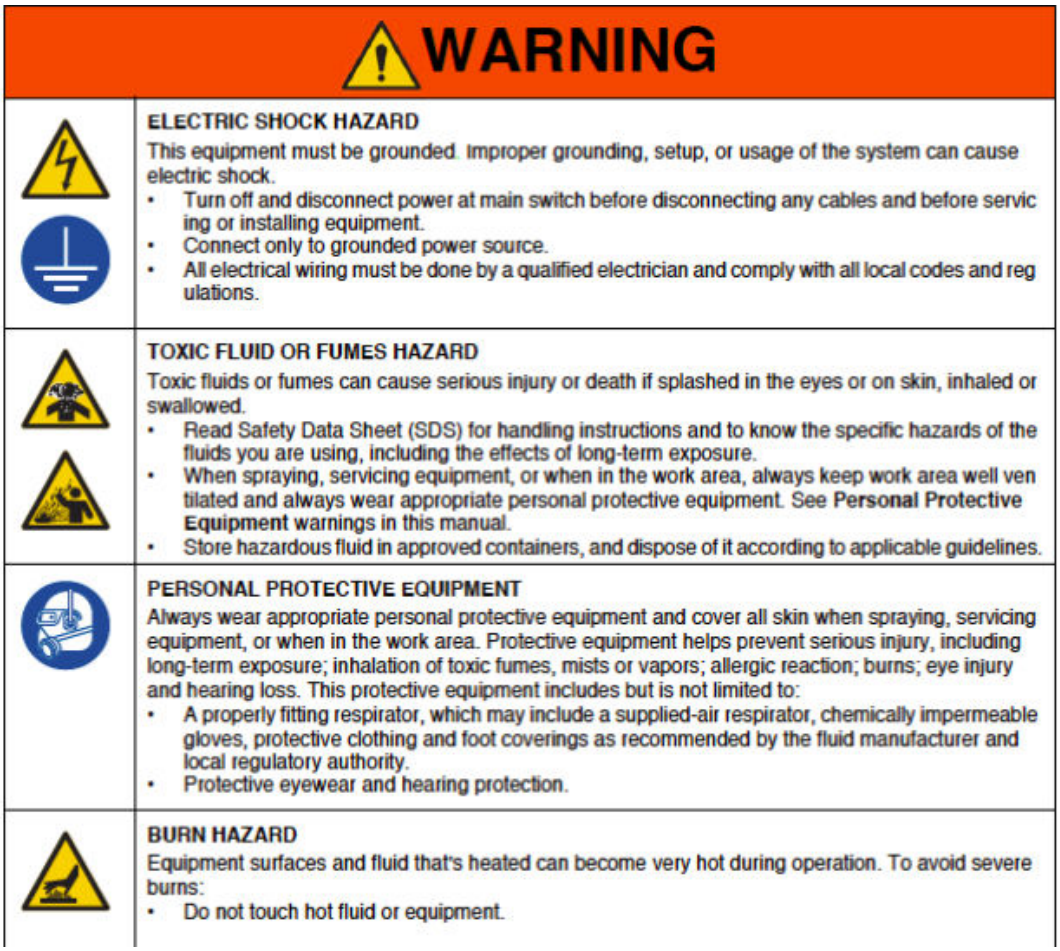


Figure 19: Adding <hazardsymbol> to <messagepanel>

The following code sample illustrates how `<messagepanel>` will be able to contain `<hazardsymbol>`. This will be useful for companies as they migrate from the old to new content models.

```
<hazardstatement type="warning">
  <messagepanel>
```

```
    <typeofhazard>GENERAL HAZARDS</typeofhazard>
    <consequence>Overriding or defeating the interlocks will expose personnel to
                 hazardous conditions.</consequence>
    <howtoavoid>DO NOT override or defeat the interlocks unless specifically directed to
                do so in the procedures. When directed to override an interlock, follow all
                safety procedures and apply HEI (Lockout/Tagout) procedures as necessary.
    </howtoavoid>
    <hazardsymbol keyref="general-warning"/>
  </messagepanel>
</hazardstatement>
```

2.1.16 Stage two: #217 Remove @domains attribute

Remove the domains attribute, and the tokens used for the domains attribute; for specialized attributes, replace the existing parenthetical syntax with a simpler token syntax..

Date and version information

Include the following information:

Date that this feature proposal was completed
June 2019

Champion of the proposal
Robert D. Anderson

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
May 14 2019

Reviewers for Stage 2 proposal
Chris Nitchie, Bill Burns, Eliot Kimber

Links to e-mail discussion that resulted in new versions of the proposal
xxx

Link to the GitHub issue
https://github.com/oasis-tcs/dita/issues/217

Original requirement or use case

Discussed May 14 2019 based on email to the DITA TC and extended follow-up discussion. Given the limited utility of the current domains attribute design, the high level of complexity needed to use it properly, and the mostly-theoretical benefits that have never been realized in 15 years of use, we should drop `@domains` attribute requirements that place a high level of burden on DITA architects and Implementors.

Use cases

This proposal removes extraneous features in order to simplify DITA design and implementation.

- For information architects: removing `@domains` will remove steps from creating specializations or constraints, simplifying the process for each.
- For specification readers and editors: this will remove some of the most technically complex portions of the specification (about how to set up grammar file tokens for `@domains`), particularly for specialization modules that combine structural and domain tokens.
- For DITA implementations: today there are complex rules about how to use `@domains` tokens when evaluating `@conref`; those rules are technically complex, and can confuse tool users when

implemented properly. Removing those rules will simplify DITA implementations and avoid those confusing results.

In addition, this proposal replaces `@domains` with a new attribute for the one domain token that is necessary for processing and generalization. This gives a clearer purpose for the attribute (simplifying the spec for readers and simplifying DITA in general). Along with moving the value to a new attribute, this proposal simplifies the syntax for attribute domain tokens, making it easier to create that value and easier for implementations to process that value.

New terminology

N/A

Proposed solution

1. Remove definitions of `@domains`
2. Remove specification sections and topics about how to properly define tokens for `@domains`
3. Remove the grammar file definition of `@domains`
4. Define a new attribute for declaring specializations of `@props` and `@base`
5. Define a simpler syntax for declaring specializations of `@props` and `@base`

Benefits

Address the following questions:

Who will benefit from this feature?

- Information architects creating specializations or constraints
- Readers of the specification
- Maintainers of the specification
- Implementors of DITA processing tools

What is the expected benefit?

Removes one of the most technically complex portions of the specification, and also simplifies the process for evaluating domain tokens with specialized attributes.

How many people probably will make use of this feature?

- All those creating or maintaining specialization and constraint modules will have fewer steps to go through.
- Editors, reviewers, and readers of the specification will no longer have to edit / review / read the extremely complex topics about domain tokens.
- Tools that strictly comply with the specification will no longer have to work with the complex rules around `@domains` tokens.
- No impact on most DITA authors who do not create or maintain grammar file modules.

How much of a positive impact is expected for the users who will make use of the feature?

Medium level impact to those listed above. The rules today are relatively straightforward but require following a complicated list of rules for little or no perceived benefit, so avoiding those rules will reduce that burden for all who encounter them today.

Technical requirements

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

Adding an attribute

- Name of the attribute: `@specializations`

  Syntax for the new attribute: today's syntax will be simplified. Today's tokens use the syntax `a(props newthing)` to indicate that `@newthing` is a specialization of `@props`; similarly, `a(props newthing newerThing)` indicates that `@newerThing` is specialized from `@newthing` which is specialized from `@props`. Instead, that syntax will be simplified to use a single token without spaces. Each attribute name is separated from its ancestor by a slash: `a(props newthing)` is simplified to `@props/newthing`, and `a(props newthing newerThing)` is simplified to `@props/newthing/newerThing`

  The same syntax is used for specializations of `@base`. While that attribute does not normally provide any inheritance based processing, the token must still be defined; without it, generalization processors would not have any way to recognize and generalize the specialized attribute back into `@base`. So, where a specialization of base named `@myInfo` would define a token today using the syntax `a(base myInfo)`, with this proposal the syntax would match the one above for `@props`: `@base/myInfo`

  The method for integrating these tokens into the `@specializations` attribute matches the current method for adding tokens to `@domains` using the configured grammar file shell.

  Note Originally I considered using an attribute name that explicitly limited this to attribute specializations, such as `@attribute-extensions` or `@special-atts`. Based on review feedback, I've gone with a more generic name `@specializations`, which will help future proof us if DITA 2.x eventually needs to provide additional specialization tokens unrelated to attributes.

  Comment by RobertAnderson
  Originally proposed several alternatives for the attibute name: `@extendedAtts` (alternatives for discussion: `@special-atts`, `@specialized-attributes`, `@specializedAttributes`, `@attributeExtensions`, or any other name suggested by a reviewer)

  Chris Nitchie suggeseted `@specializations`, under the theory that this keeps open the idea that future specialization info *could* be added without changing the name or adding a new attribute. Chris also suggested the leading @ for this reason.

- Elements that will get the new attribute: all elements that currently take `@domains` (that is, `<map>`, `<topic>`, and their specializations)
- Processing expectations that are associated with the new attribute: same as those associated with attribute specialization tokens in `@domains` with DITA 1.3
- The attribute does not contain translatable text

Removing an attribute

- Removing `@domains` from `<topic>`, `<map>`, and all specializations of those.

Processing impact

Processors can remove support for most aspects of `@domains` processing, while processing for attribute specialization tokens must be updated to account for a new attribute name and simpler syntax.

Overall usability

No impact to authors, and much simpler for those working with domain modules.

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.

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. This will impact all specialization modules and configuration shells.

Removing or replacing a processing feature that was defined in DITA 1.3?
Removing `@domains` and most associated processing (the only critical aspect of processing, for attribute domains, is moved to a new attribute).

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?
The `@domains` attribute is intended to be used as a default attribute value retrieved from grammar files; wherever topics or maps that have added the attribute into a DITA document, the attribute will need to be removed.

Might any existing processors or implementations need to change their expectations?

- Processors can remove support for processing that is no longer defined in the specification, such as generalization-during-conref and differing support for loose versus strict constraints.
- Processors will need to be updated to use the new syntax in the new attribute for specialized attribute tokens.

Might any existing specialization or constraint modules need to be migrated?
Yes:

- Specializations of `<topic>` or `<map>` will need to remove declarations of `@domains`, and add a declaration for the new attribute
- Declarations of `@domains` tokens can be removed from non-attribute modules
- Declarations of `@domains` tokens for `@props` and `@base` attribute specializations will need to be modified to use a new syntax, and configuration shells might need to use a new syntax to add those tokens to the new attribute

Costs

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

Maintainers of the grammar files

Minor impact to remove the old attribute declaration and create the new attribute + update to use the new syntax

Editors of the DITA specification

- How many new topics will be required? *None*
- How many existing topics will need to be edited? *Several, mostly by removing content*
- Will the feature require substantial changes to the information architecture of the DITA specification? *No*

Vendors of tools

Minor impact (removing extraneous processing and simplifying parsing rules for attribute domain tokens)

DITA community-at-large

- Will this feature add to the perception that DITA is becoming too complex? *No, should have the opposite effect*
- Will it be simple for end users to understand? *Yes*
- If the feature breaks backwards compatibility, how many documents are likely to be affected, and what is the cost of migration? *Few documents should be affected, most specialization modules and configuration shells will need an update.*

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? *Minor addition to existing migration instructions*
- 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? *Migration tools for this cannot cover all cases, because specializations / shells do not have to use the same format. Tools may catch some cases but some minor updates will likely be required.*

Examples

With DITA 1.3, a specialization of `@props` must declare a token beginning with the letter `a` followed by a `(props` followed by a space followed by the name few the new attribute (followed by attribute names for further specializations, if present) followed by `)`. For example, if `@props` is specialized to `@newthing` which is specialized to `@newerThing` which is specialized to `@finalThing`, the DITA 1.3 domain token is `a(props newthing newerThing finalThing)`

With DITA 2.0 that syntax is simplified, removing the `a` and parenthetical grouping, as well as all spaces. The new declaration will be a single token without spaces, including the full ancestry of the specialized element – starting with `@props` and ending with the attribute name. For example, if `@props` is specialized to `@newthing` which is specialized to `@newerThing` which is specialized to `@finalThing`, the updated domain token will be `@props/newthing/newerThing/finalThing`

When a grammar file is parsed, a configured shell that defines three attribute extensions – `@deliveryTarget`, a `@props` specialization named `@newThing`, and a `@base` specialization named `@myInfo` – would include the following three tokens (not necessarily in this order):

```
specializations="@props/deliveryTarget @props/newThing @base/myInfo"
```

2.1.17 Stage two: #252 Add @outputclass to DITAVAL

For flagging purposes, DITAVAL today allows you to associate a variety of styles with a specific property or revision. Today, the `@outputclass` attribute is frequently used to associate a much broader range of processing or even CSS styling with an element or group of elements; allowing DITAVAL to associate that same token as a flag will give DITAVAL based flagging the same full range of possibilities.

Date and version information

Include the following information:

Date that this feature proposal was completed
September 4, 2019

Champion of the proposal
Robert D Anderson

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
June 11 2019

Reviewers for Stage 2 proposal
Kris Eberlein

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/252

Original requirement or use case

Presented June 11 2019:

Today, DITAVAL allows you to flag content based on property attributes or revision attributes in the file. A limited set of flag styles are available (including images, color, a specific set of text styles, change bars for revisions). For example, you can set up your DITAVAL file to flag everything with `platform="mac"`, and define the flag so that any of those styles are applied: every element with `platform="mac"` can be flagged with an image, or presented in bold red text, or given a yellow background, or all of the above. The intent with flagging is to provide a way to style annotated elements so that they stand out in a publication.

At the same time, `@outputclass` (now a universal attribute for DITA 2.0) is a general attribute that gives you a full range of processing possibilities. For any output, a processor can use the attribute to provide any type of custom styling. For many known HTML processors, the value is passed directly into HTML's `@class` attribute, making it directly available for styling by CSS.

The original suggestion leading to this requirement was: with my DITAVAL conditions, I should have access to the same full range of CSS based processing that I already have with `@outputclass`. At the same time, we shouldn't need to create a whole new processing feature. `@outputclass` already provides this full capability on its own, and is already widely used for CSS or other custom processing. We should be able to flag content by setting up an `@outputclass` attribute on flagged content, which gives us the full range of formatting already available with that mechanism.

Use cases

1. Today, I have a lot of my content marked up with `rev="v7"` for the upcoming v7 release.
2. I also have marketing material that shows off new features. It uses `outputclass="exciting"` for exciting new features. That value goes through to HTML `@class` attributes, where it is picked up by my CSS and JS so that 1) it renders with a distinct font, 2) the text expands and contracts, and 3) a "ta-da" sound plays when you hover over the feature description.
3. When I publish my documentation for V7, I'd like all of those custom rendering features -- which I already have working thanks to `@outputclass` -- to show up with each V7 change in the documentation.
4. Allowing me to associate an `@outputclass` token with any revision or property in the DITAVAL gives me access to all of my existing custom formatting, without needing to implement anything new.

New terminology

N/A

Proposed solution

Add `@outputclass` to the `<prop>` and `<revprop>` elements in the DITAVAL format. The attribute is CDATA, allowing one or more tokens (exactly the same syntax as `@outputclass` in DITA).

When a property evaluates to "flag" based on existing DITAVAL rules, and `revprop/@outputclass` or `prop/@outputclass` is specified, processors should treat the flagged element as if the full value `@outputclass` value in the DITAVAL was specified on the full element.

If `@outputclass` is already specified on the flagged element, the DITAVAL based `@outputclass` comes *first* -- in CSS cascading order, this means that the explicit value on the element has higher precedence for any conflicting styles.

If two properties each evaluate to "flag", and each associate an `@outputclass` value with the flag, the order in which they are applied is determined by the processor.

Benefits

Address the following questions:

Who will benefit from this feature?
Anyone who wants more advanced flagging capabilities associated with existing content metadata attributes.

What is the expected benefit?
Allows content to make use of `@outputclass` based styling capabilities for flagging.

How many people probably will make use of this feature?
Anyone who already uses DITAVAL based flagging; those who do not use DITAVAL flagging because the flag capabilities are too limited; those who already use `@outputclass` based styling and would like to apply similar styling based on existing metadata.

How much of a positive impact is expected for the users who will make use of the feature?
For those making use of the feature, allows a broad range of new capabilities with little learning curve.

Technical requirements

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

Adding new elements or attributes

Adding an attribute

- Adds `@outputclass` to `<revprop>` and `<prop>` in the DITAVAL format.
- When one of those elements evaluates to "flag" using existing DITAVAL based rules, in addition to any flagging applied by existing attributes, processors will treat the flagged element as if the `@outputclass` value is also specified on the element.
- The attribute does **not** contain translatable text.

Processing impact

- There is a processing impact for any processor that handles DITAVAL based flagging.
- When a property on an element in DITA content evaluates to "flag", and the matching DITAVAL rule for flagging specifies `@outputclass`, processors will treat the DITA element as if that `@outputclass` attribute is specified in the content.
- If processors do not provide any capability for custom styling or processing based on `@outputclass` in DITA, the net effect is zero - if there is no way to handle custom processing for the attribute in DITA, then adding the equivalent via flagging has no meaning.
- For other processors, any ability to customize processing or styling based on existing `@outputclass` values becomes available on the flagged element.
- What edge cases need to be considered?

  1. What happens when the `@outputclass` attribute in DITAVAL has more than one value? Both are used (the full value is made available to the flagged DITA element)
  2. What happens when the flagged element already specifies `@outputclass`? It is combined with the value from the DITAVAL, with the flag value coming first (so that CSS based selectors give priority to the local value over the flag value)
  3. What happens when more than one flag value provides `@outputclass`? Both are added. The order is determined by the processor: there is no requirement for the order in which property tokens are evaluated, and requiring an order in order to handle this edge case seems to add far more complexity than is warranted.

Overall usability

Should be fairly straightforward to use; it does not provide any new terminology, and the `@outputclass` attribute is already familiar from earlier versions of DITA.

Backwards compatibility

N/A

Migration plan

N/A

Costs

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

Maintainers of the grammar files

Minor

Editors of the DITA specification

- How many new topics will be required? 0
- How many existing topics will need to be edited? 2 (element reference topics for `<prop>` and `<revprop>`)
- Will the feature require substantial changes to the information architecture of the DITA specification? No substantial changes
- If there is new terminology, is it likely to conflict with any usage of those terms in the existing specification? N/A

Vendors of tools

XML editors, component content management systems, processors, etc: Small impact; this makes use of existing DITA features in a new way, so would not expect a large cost.

DITA community-at-large

- Will this feature add to the perception that DITA is becoming too complex? No
- Will it be simple for end users to understand? Yes
- If the feature breaks backwards compatibility, how many documents are likely to be affected, and what is the cost of migration? N/A

Producing migration instructions or tools

N/A

Examples

Assuming the following DITAVAL properties are provided to the processor:

```
<prop action="flag" att="otherprops" val="simple" outputclass="SimpleCSS"/>
<prop action="flag" att="otherprops" val="twovals" outputclass="addColor addFont"/>
<prop action="flag" att="platform" val="mac" outputclass="appleFont"/>
<revprop action="flag" val="v7" outputclass="exciting"/>
```

- The result of evaluating flags on my paragraph `<p otherprops="simple">` is equivalent to `<p otherprops="simple" outputclass="SimpleCSS">`
- The result of evaluating flags on my paragraph `<p otherprops="twovals">` is equivalent to `<p otherprops="simple" outputclass="addColor addFont">`
- The result of evaluating flags on my paragraph `<p platform="mac" outputclass="shinymac">` is equivalent to `<p platform="mac" outputclass="appleFont shinymac">`
- The result of evaluating flags on my paragraph `<p platform="mac" rev="v7" outputclass="shinymac">` is *determined by the processor*, because `@platform` and `@rev` can be evaluated in any order. The result is equivalent to *either* `<p platform="mac" rev="v7" outputclass="appleFont exciting shinymac">` *or* `<p platform="mac" rev="v7" outputclass="exciting appleFont shinymac">`

2.1.18 Stage two: #253 Indexing changes

Refactor indexing to remove redundant elements and reduce complexity

Date and version information

This proposal includes the following information:

Date that this feature proposal was completed

14 June 2019

Champion of the proposal
Kristen James Eberlein

Links to any previous versions of the proposal
Not applicable

Links to minutes where this proposal was discussed at stage 1 and moved to stage 2
11 June 2019

Reviewers for Stage 2 proposal
Robert Anderson
Deb Bissantz

Links to e-mail discussion that resulted in new versions of the proposal
Not applicable

Link to the GitHub issue
https://github.com/oasis-tcs/dita/issues/253

Original requirement or use case

Clean up indexing elements:

- Remove the indexing domain, and add <index-see> and <index-see-also> to the base
- Remove `<index-base>`
- Remove `<index-sort-as>`

See https://lists.oasis-open.org/archives/dita/201906/msg00017.html and https://lists.oasis-open.org/archives/dita/201906/msg00022.html.

Note The `<indextermref>` element was removed as as result of proposal #36: Remove deprecated elements and attributes.

Use cases

- Simplify the indexing experience for authors
- Simplify the experience for DITA practitioners who create document-type shells
- Simplify the experience for maintainers of the DITA grammar files by removing unnecessary elements and domains

New terminology

Not applicable

Proposed solution

Simple changes to grammar files and specification

Benefits

This proposal addresses the following questions:

Who will benefit from this feature?
Everyone using DITA 2.0 who indexes content

What is the expected benefit?
Decrease in complexity; removal of a domain

How many people probably will make use of this feature?
Everyone using DITA 2.0 who indexes content

How much of a positive impact is expected for the users who will make use of the feature?
Minor

Technical requirements

This proposal involves the following changes:

Renaming or refactoring elements and attributes
Adding `<index-see>` and `<index-see-also>` to `commonElementsMod.rng`; redefining the value of the `@class` attribute for those elements

Removing a domain

- Removing `indexingDomain.rng`
- Removing the indexing domain from `basemap.rng` and `basetopic.rng`

Removing an element

- Removing `<index-base>` from `commonElementsMod.rng`
- `<index-sort-as>` will be removed as a result of removing the indexing domain

Overall usability
Simplifies the current authoring experience by removing `<index-base>`; reduces the number of domains that need to be considered for document-type shells; removes a redundant element

Backwards compatibility

This proposal addresses the following questions:

Was this change previously announced in an earlier version of DITA?
No

Removing a document type that was shipped in DITA 1.3?
No

Removing a domain that was shipped in DITA 1.3?
Yes – Removing the indexing domain

Removing a domain from a document type shell was shipped in DITA 1.3?
Yes – Removing the indexing domain from base map and topic

Removing or renaming an element that was shipped in DITA 1.3?
Yes – Removing `<index-base>` and `<index-sort-as>`

Removing or renaming an attribute that was shipped in DITA 1.3?
No

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?
Yes – Removing `<index-base>` from the content model of `<indexterm>`

Changing specialization ancestry?
Yes – As follows:

Element	DITA 1.3	DITA 2.0
<index-see>	"+ topic/index-base indexing-d/index-see "	"- topic/index-see "
<index-see-also>	"+ topic/index-base indexing-d/index-see-also "	"- topic/index-see-also "

Removing or replacing a processing feature that was defined in DITA 1.3?

Yes – If processors did not implement support for <sort-as> within <indexterm>

Are element or attribute groups being renamed or shuffled?

Yes – Adding <index-see> and <index-see-also> to base

Migration plan

This proposal addresses the following questions:

Might any existing documents need to be migrated?

- Implementations will need to remove any instances of <index-base> from their DITA source.
- Implementations will need to change any instances of <index-sort-as> to <sort-as>. This can be accomplished by search-and-replace or a simple script.
- If the DITA source has instances <index-see> or <index-see-also> with values of the @class attribute explicit, that will need to be modified.

Might any existing processors or implementations need to change their expectations?

- Processor should no longer expect to encounter <index-sort-as>; if they have not implemented support for <sort-as>, they will need to do so.
- Processors will need to use new values for @class attribute for <index-see> and <index-see-also>.

Might any existing specialization or constraint modules need to be migrated?

- (Removal of elements) If anyone has specialized <index-base>, they will need to redo their specialization modules.
- (Specialization ancestry changes) If anyone has specialized <index-see> or <index-see-also>, they will need to change the values of the @class attributes in the specialization modules.

Costs

This proposal has a (time and effort) impact on the following groups:

Maintainers of the grammar files

- Delete indexingDomain.rng
- Edits to basemap.rng and basetopic.rng
- Edits to commonElementsMod.rng
- Edits to catalog files

Editors of the DITA specification

The changes to the DITA specification will be minor:

- No new topics.

- Removal of `<index-base>` from "Specialization elements" cluster.
- Removal of `<index-base>` and `<index-sort-as>` topics from the "DITA elements, A to Z" topic.
- Removal of the "Indexing elements group" topic cluster from the element-reference topics.
- Addition of `<indexterm>`, `<index-see>`, and `<index-see-also>` topics to an "Indexing elements" cluster. I suggest that it be a child of "Metadata elements".
- Moving content from `<index-sort-as>` topic into other topics about sorting.

Vendors of tools

Tools will need to:

- Remove any legacy processing for `<index-sort-as>`
- Handle the changed `@class` attributes for `<index-see>` and `<index-see-also>`

DITA community-at-large

Overall, this proposal will reduce complexity and might make indexing easier for authors.

Since this proposal removes elements, there is a migration cost. It affects the following:

- DITA source that contains `<index-base>`, which should NOT be in DITA source
- DITA source that contains `<index-sort-as>`: Not common for DITA source authored in English, but common for (indexed) DITA source authored in Japanese
- Implementations that have specialized `<index-base>` and `<index-sort-as>`
- Stylesheets that format `<index-see>` and `<index-see-also>`?

Producing migration instructions or tools

Migration instructions can be integrated into the *MIgrating to DITA 2.0* committee note. They will be straight-forward.

Examples

This section contains examples of DITA 1.3 markup and how that markup needs to be modified in DITA 2.0.

DITA 1.3	DITA 2.0

`<indexterm>`
`  <data>`
`  <index-sort-as>data</index-sort-as>`
`</indexterm>`
	`<indexterm>`
	`  <data>`
	`  <sort-as>data</sort-as>`
	`</indexterm>`

2.1.19 Stage two: #277 Change specialization base for <imagemap>

Change the specialization base for `<imagemap>` to enable image maps to be treated as images.

Date and version information

Date that this feature proposal was completed

17 July 2019; updated on 12 August 2019

Champion of the proposal

Kristen James Eberlein, Eberlein Consulting LLC

Links to any previous versions of the proposal

E-mail to list on 17 July 2019

Links to minutes where this proposal was discussed at stage 1 and moved to stage 2
16 July 2019

Reviewers for Stage 2 proposal

Robert Anderson, IBM
Eliot Kimber, Individual member

Links to e-mail discussion that resulted in new versions of the proposal
Not applicable

Link to the GitHub issue
https://github.com/oasis-tcs/dita/issues/277

Original requirement or use case

“Originally, <imagemap> was a specialization of <fig>. This means you cannot use an <imagemap> directly inside of a <fig>, which makes it complicated to add a title to your image. … If we change the specialization base of <imagemap> from <fig> to something slightly more generic such as , that may simplify the adding of a title (or other content) around an <imagemap> in a <fig>.” Zoe Lawson, 09 July 2019.

Use cases

Changing the specialization base for `<imagemap>` will enable authors to treat imagemaps in the same way as images:

- Insert `<imagemap>` in a figure with an associated title
- Associate a description with the `<imagemap>`
- Include `<imagemap>` in rendered lists of figures

New terminology

None

Proposed solution

Change the specialization base for `<imagemap>` and `<area>` to `<div>`

Benefits

This proposal addresses the following questions:

Who will benefit from this feature?
Authors will benefit from consistency in markup for images and imagemaps.

What is the expected benefit?
Increased consistency in authoring markup.

How many people probably will make use of this feature?
Few.

How much of a positive impact is expected for the users who will make use of the feature?
Minor.

Technical requirements

This section contains a detailed description of how the solution will work.

Renaming or refactoring elements and attributes

This proposal suggests changing the specialization base for `<imagemap>` and its child element `<area>`.

Processing impact

None

Overall usability

Positive effect on usability; authors will be able to treat imagemaps in the same way as images.

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

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?

No

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?

Yes – As follows:

Element	DITA 1.3	DITA 2.0
`<area>`	"+ topic/figgroup ut-d/area "	"+ topic/div ut-d/area "
`<imagemap>`	"+ topic/fig ut-d/imagemap "	"+ topic/div ut-d/imagemap "

Removing or replacing a processing feature that was defined in DITA 1.3?

No

Are element or attribute groups being renamed or shuffled?

No

Migration plan

This proposal addresses the following questions:

Might any existing documents need to be migrated?

Only if CCMS or other applications make the `@class` attribute explicit in the source.

Might any existing processors or implementations need to change their expectations?

(Very unlikely) If any stylesheets are matching on the full value of the @class attribute, they will need to be modified.

(Very unlikely) If any stylesheets are expecting fallback processing for `<fig>`, they might get unexpected results.

Might any existing specialization or constraint modules need to be migrated?

The "Domains extensions" section of all document-type shells that integrate the utilities domain will need to be modified.

It is very unlikely that anyone has specialized `<imagemap>` or `<area>`. But if implementations have done so, they will need to update their specialization modules.

(An extreme edge case) If an implementation has applied a constraint module to remove `<div>` – or limit where it can be used, the constraint modules might need to be updated to ensure image maps can still be placed in the pre-DITA 2.0 locations.

If no migration need is anticipated, why not?

Not applicable.

Costs

This proposal has a (time and effort) impact on the following groups:

Maintainers of the grammar files

The following grammar files will need to be modified:

- `basemap.dtd`
- `basemap.rng`
- `basetopic.dtd`
- `basetopic.rng`
- `utilitiesDomain.ent`
- `utilitiesDomain.mod`
- `utilitiesDomain.rng`

Editors of the DITA specification

The following specification topics will need to be modified:

- `area.dita`
- `imagemap.dita`

Vendors of tools

Not applicable

DITA community-at-large

Not applicable

Producing migration instructions or tools

MIgration instructions will be simple and can be handled in the planned committee note, *Migrating to DITA 2.0*.

Examples

Changing the specialization base of `<imagemap>` (and its child element `<area>`) will allow authors to insert `<imagemap>` within a figure.

```
<fig>
<title>Map of the world</title>
<imagemap>
 <image href="imagemapworld.jpg">
   <alt>Map of the world showing 5 areas</alt>
 </image>
 <area><shape>rect</shape><coords>2,0,53,59</coords>
  <xref href="d1-s1.dita">Section 1 alternative text</xref>
 </area>
 <area><shape>rect</shape><coords>54,1,117,60</coords>
  <xref href="d1-s2.dita"><!-- Pull title from d1-s2.dita --></xref>
 </area>
 <area><shape>rect</shape><coords>54,62,114,116</coords>
  <xref href="#inline" type="topic">Alternative text for this rectangle</xref>
 </area>
 <area><shape>circle</shape><coords>120,154,29</coords>
  <xref format="html" href="test.html">Link to a test html file</xref>
 </area>
 <area><shape>poly</shape>
  <coords>246,39,200,35,173,52,177,86,215,90,245,84,254,65</coords>
  <xref format="pdf" href="test.pdf">Link to a test PDF file</xref>
 </area>
</imagemap>
</fig>
```

2.1.20 Stage two: #279 Remove @lockmeta attribute

The `@lockmeta` attribute will be removed from the `<topicmeta>` and `<bookmeta>` elements.

Date and version information

Date that this feature proposal was completed

TBD

Champion of the proposal

Bill Burns

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://wiki.oasis-open.org/dita/Agenda-16-July-2019

Reviewers for Stage 2 proposal

TBD

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/278

Original requirement or use case

The `@lockmeta` attribute is ambiguous and poorly defined. While it was present in the DITA 1.0 DTDs, it was not defined in the spec. An ambiguous definition was added in the DITA 1.1 timeframe. Rather than work to define an attribute that we do not see the need for, the DITA TC recommends that it be removed.

Modified grammar files

The following files must be modified:

DTDs

map.mod
bookmap.mod

RNGs

mapMod.rng
bookmapMod.rng (appears to be missing already)

Use cases

N/A

New terminology

N/A

Benefits

Who will benefit from this feature?

Users who are unsure about what metadata will be applied for a particular topic. The ambiguity of the specification leaves the question open about how `@lockmeta` impacts metadata assignment.

What is the expected benefit?

Removing the attribute eliminates ambiguity. The assumption is that metadata in the map overrides when only one metadata element or value is allowed but ssupplementswhat is in the referenced topicwhen more than one element or value is permitted.

How much of a positive impact is expected for the users who will make use of the feature?

Minor

Technical requirements

Removing elements or attributes

Removing an attribute

- `@lockmeta` would be removed from `<topicmeta>` and `<bookmeta>` elements.

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

Removing or renaming an attribute that was shipped in DITA 1.3?

Yes.

Migration plan

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

Might any existing documents need to be migrated?
Yes. This attribute would need to be removed from any maps on which it is set. Authors or organizations would also need to rethink their metadata strategy to ensure that they don't override topic metadata values where doing so is not desired.

Might any existing processors or implementations need to change their expectations?
Possibly but doubtful because the processing expectations were never clear from the start. Since the attribute is simply going away, the only changes needed would be for processors that did not comply with the default processing anyway.

Might any existing specialization or constraint modules need to be migrated?
Possibly, if a specialization includes this attribute.

Costs

Maintainers of the grammar files
Minimal

Editors of the DITA specification
Minimal:

- No new topics would be required.
- Three topics would need to be edited: 2.2.4.3, 3.3.1.3, and 3.10.6.2.1.
- No changes would be required for the information architecture of the DITA specification.
- No new terminology is necessary.

Vendors of tools
Any impact should be minimal.

DITA community-at-large

We expect any impact to be small as community use of this attribute is expected to be low.

2.2 Technical Content edition

2.2.1 Stage two: #85 Add <sub> and <sup> to glossary elements

The `<sub>` and `<sup>` elements are not allowed in several elements from glossentry, which means that terms cannot be used properly in certain contexts. Since `<sub>` and `<sup>` are specialized from `<ph>`, adding the `<ph>` element solves the issue.

Date and version information

Date that this feature proposal was completed
January 29, 2018

Champion of the proposal
Scott Hudson

Links to any previous versions of the proposal
https://www.oasis-open.org/apps/org/workgroup/dita/download.php/62449/Issue85-AddSupSubToGlossElements.html

Initial proposal
https://lists.oasis-open.org/archives/dita/201710/msg00042.html

Links to minutes where this proposal was discussed at stage 1 and moved to stage 2

http://markmail.org/message/h7x7dmilcatu6klj?q=Add+%3Csup%3E+and+%3Csub%3E+to+glossary+elements+list:org%2Eoasis-open%2Elists%2Edita

Links to e-mail discussion that resulted in new versions of the proposal

- http://markmail.org/message/duqczthaqgh2a5oq?q=Add+ph+to+glossentry+models+list:org%2Eoasis-open%2Elists%2Edita
- https://www.oasis-open.org/committees/download.php/62511/minutes20180213.txt

Link to the GitHub issue

https://github.com/oasis-tcs/dita/issues/85

Original requirement or use case

Authors need to be able to use superscript and subscript in several elements:

- `<glossSurfaceForm>`
- `<glossAcronym>`
- `<glossSynonym>`
- `<glossShortForm>`
- `<glossAbbreviation>`

Use cases

When implemented, authors will be able to include subscript and superscript content as part of most glossary elements. For example:

```
<glossentry id="gl_DesignManeuveringSpeed">
    <glossterm>design maneuvering speed</glossterm>
    <glossBody>
        <glossPartOfSpeech value="noun"/>
        <glossSurfaceForm>design maneuvering speed (V<sub>A</sub>)</glossSurfaceForm>
        <glossUsage>Spell out on first use. Use V<sub>A</sub> in subsequent references. <dl>
                <dlentry>
                    <dt>Examples</dt>
                    <dd>The pilot's operating handbook is the best source for determining your
                        aircraft's design maneuvering speed (V<sub>A</sub>). The cockpit
placard
                        might also include V<sub>A</sub> information.</dd>
                </dlentry>
            </dl><p>See also <xref href="" <xref
                    href="" speeds</xref>.</p></glossUsage>
        <glossAlt id="alt_gl_DesignManeuveringSpeed">
            <glossAcronym>V<sub>A</sub></glossAcronym>
        </glossAlt>
    </glossBody>
</glossentry>
```

Proposed solution

Modify the content models to allow the `<ph>` element within the following elements:

- `<glossSurfaceForm>`
- `<glossAcronym>`
- `<glossSynonym>`
- `<glossShortForm>`
- `<glossAbbreviation>`

Benefits

Who will benefit from this feature?

Glossary term authors

What is the expected benefit?

Glossary authors will benefit significantly with the ability to include subscript and superscript content as part of a glossary term.

How many people probably will make use of this feature?

many

How much of a positive impact is expected for the users who will make use of the feature?

Any industries dealing with chemical or physics-related content can benefit from this change, including Aerospace and Aviation.

Technical requirements

Renaming or refactoring elements and attributes

Content model name / location	Original model	Proposed model
glossSurfaceForm.content (`glossentry.mod`)	`<!ENTITY % glossSurfaceForm.content "(#PCDATA | %keyword; | %term; | %text; | %tm;)*" >`	`<!ENTITY % glossSurfaceForm.content "(#PCDATA | %keyword; | %term; | %text; | %tm; | %ph;)*" >`
glossSurfaceForm.content (`glossentryMod.rng`)	`<define name="glossSurfaceForm.content"> <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> </choice> </zeroOrMore> </define>`	`<define name="glossSurfaceForm.content"> <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> <ref name="ph" dita:since="2.0" /> </choice> </zeroOrMore> </define>`
glossAcronym.content (`glossentry.mod`)	`<!ENTITY % glossAcronym.content`	`<!ENTITY % glossAcronym.content "(#PCDATA |`

Content model name / location	Original model	Proposed model
	"(#PCDATA | %keyword; | %term; | %text; | %tm;)*" >	%keyword; | %term; | %text; | %tm; | %ph;)*" >
glossAcronym.content (glossentryMod.rng)	<define name="glossAcronym.content" > <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> </choice> </zeroOrMore> </define>	<define name="glossAcronym.content" > <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> <ref name="ph" dita:since="2.0""/> </choice> </zeroOrMore> </define>
glossSynonym.content (glossentry.mod)	 <!ENTITY % glossSynonym.content "(#PCDATA | %keyword; | %term; | %text; | %tm;)*" >	<!ENTITY % glossSynonym.content "(#PCDATA | %keyword; | %term; | %text; | %tm; | %ph;)*" >
glossSynonym.content (glossentryMod.rng)	<define name="glossSynonym.content" > <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> </choice> </zeroOrMore> </define>	<define name="glossSynonym.content" > <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> <ref name="ph" dita:since="2.0" /> </choice> </zeroOrMore> </define>
glossShortForm.content (glossentry.mod)	 <!ENTITY %	<!ENTITY % glossShortForm.content

Content model name / location	Original model	Proposed model
	glossShortForm.content "(#PCDATA | %keyword; | %term; | %text; | %tm;)*" >	"(#PCDATA | %keyword; | %term; | %text; | %tm; | %ph;)*" >
glossShortForm.content (glossentryMod.rng)	<define name="glossShortForm.content"> <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> </choice> </zeroOrMore> </define>	<define name="glossShortForm.content"> <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> <ref name="ph" dita:since="2.0" /> </choice> </zeroOrMore> </define>
glossAbbreviation.content (glossentry.mod)	<!ENTITY % glossAbbreviation.content "(#PCDATA | %keyword; | %term; | %text; | %tm;)*" >	<!ENTITY % glossAbbreviation.content "(#PCDATA | %keyword; | %term; | %text; | %tm; | %ph;)*" >
glossAbbreviation.content (glossentryMod.rng)	<define name="glossAbbreviation.content"> <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> </choice> </zeroOrMore> </define>	<define name="glossAbbreviation.content"> <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> <ref name="ph" dita:since="2.0" /> </choice> </zeroOrMore> </define>

Processing impact

Minimal. All `@domains`- and `@class`-based processing of specialized elements and attributes will continue to function exactly as it has before. Processors may need to be adjusted to account for the additional allowed children and specializations of `<ph>`, such as `<xref>`.

Overall usability

Current DITA users, and current document type shells, will be unaffected. Usability of authoring with specializations will ultimately be improved due to content models that more closely model business requirements.

Note Adding <ph> will also allow the following elements to be available (particularly from the equation, highlight, programming, software, user interface, and XML mention domains):

- <abbreviated-form>
- <apiname>
- <b>
- <cmdname>
- <codeph>
- <equation-inline>
- <filepath>
- <i>
- <line-through>
- <markupname>
- <menucascade>
- <msgnum>
- <msgph>
- <numcharref>
- <option>
- <overline>
- <parameterentity>
- <parmname>
- <ph>
- <sub>
- <sup>
- <synph>
- <systemoutput>
- <textentity>
- <tt>
- <u>
- <uicontrol>
- <userinput>
- <varname>
- <wintitle>
- <xmlatt>
- <xmlelement>
- <xmlnsname>
- <xmlpi>

Backward compatibility

There is no impact to backward compatibility, since the <ph> element is an addition to existing content models.

Migration plan

No existing content needs to be migrated, since the <ph> element is an addition to existing content models.

Existing constraints or specializations might need to be adjusted or updated to remove unwanted elements from the glossary content models. One benefit, is that any elements or domains constrained in the shell DTDs (particularly from the equation, highlight, programming, software, user interface, and XML mention domains) will not be allowed in the glossary models.

Costs

Grammar files need to be updated as outlined in this proposal. Downstream schema generation should be unaffected.

Since the content model appendix will be auto-generated, no additional changes to the DITA specification is needed.

Tool vendors may need to update functionality if they have unique support for display or processing of inline domain elements (particularly from the equation, highlight, programming, software, user interface, and XML mention domains) within the glossary elements. Otherwise, no changes should be necessary.

Examples

See Use cases (94), above.

2.2.2 Stage two: #106: Nest <steps>

Allow the <steps> element to nest, in order to address one of our more common authoring pain points, simplify reuse, and reduce overall element count.

Date and version information

Date that this feature proposal was completed
14 March 2018

Champion of the proposal
Robert D. Anderson

Links to any previous versions of the proposal

- Original thread raising the issue 20 July 2017: https://lists.oasis-open.org/archives/dita/201707/msg00037.html
- Discussed 1 August 2017: https://lists.oasis-open.org/archives/dita/201708/msg00004.html

Links to minutes where this proposal was discussed at stage 1 and moved to stage 2
https://lists.oasis-open.org/archives/dita/201803/msg00052.html

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/106

Original requirement or use case

Discussed August 1 2017: The current task structure allows <substeps> to be nested within <steps>, but you cannot nest <steps> within itself or <substeps> within itself, making it impossible to nest any

sort of step beyond the second level. We have heard the following requests related to that structure, all suggesting that <steps> should be able to nest within itself:

- I need the ability to nest three levels of steps/sub-steps. The current limitation is arbitrary, with the TC deciding for me that I am supposed to create additional nested tasks. The current limitation is also meaningless because I just insert <ol> inside of the <info> portion of the step.
- I am frustrated at having to insert different elements for the same type of content. There is no semantic difference between <steps> and <substeps> – the ability to nest one task within another means that steps in one context are sub-steps in another.
- I need to reuse a step in one context as a sub-step in another. I cannot, because the difference in elements makes it impossible to establish content reference relationships with @conref or @conkeyref between a step and a sub-step (even tough as established above, a step in one context is often a sub-step in another context).
- My sub-steps don't have a specified order, but the current markup only allows me to use ordered <substeps> even though I can have unordered steps at the task level.

Use cases

As indicated by the use cases / requirements above:

- Those who already create 3 levels of lists in a task would like to use proper markup (<steps> within <steps> within <steps>, rather than <ol> within <info> within <substeps> within <steps>).
- Those who need to reuse a <substep> from one task as a <step> in another (or vice versa) would like to do so as a single content reference (rather than having to use cut/paste or having to reference each individual sub-element).
- Those who want to use sub-steps without specifying an order would like to do so without resorting to <ul> inside of <info>.

New terminology

N/A

Proposed solution

1. Allow <steps> and <steps-unordered> to nest within <step> (after the command, as part of the 0-to-many group of elements including <choices>, <info>, and other components).
2. Remove the <substeps> and <substep> elements, which are no longer necessary.

Benefits

Who will benefit from this feature?

All authors using the task model.

What is the expected benefit?

- Reduced frustration
- Easier reuse
- Ability to use appropriate markup for all steps in a task

How many people probably will make use of this feature?

Many users of the task document type.

How much of a positive impact is expected for the users who will make use of the feature?

Probably a medium impact; this is a long-standing issue that regularly causes small annoyances, so fixing it will remove a small but constant source of irritation.

Technical requirements

Adding new elements or attributes

Adding an element

This is not exactly adding a new element, but adding an existing element to a new context. The definitions of the `<steps>` element and `<steps-unordered>` elements do not change.

Removing elements or attributes

Removing an element

The `<substeps>` and `<substep>` elements will be removed from the task specialization.

Processing impact

There should be little processing impact; tools with special processing for steps/sub-steps will need to handle the possibility of 3 or more levels of nesting, while tools that already rely on fallback processing to ordered / unordered lists will see no impact.

Overall usability

No additional setup is necessary; based on feedback, authors will consider the revised model more usable than the existing one, so this will be a slight improvement.

Backwards compatibility

Was this change previously announced in an earlier version of DITA?

No

Removing or renaming an element that was shipped in DITA 1.3?

Yes, removing `<substeps>` and `<substep>` from the task specialization.

Changing a content model by removing something that was previously allowed, or by requiring something that was not?

Yes; the change removes `<substeps>` from the content model of `<step>`

Migration plan

Might any existing documents need to be migrated?

Yes, any task that uses `<substeps>`. If there is a script to migrate files, this will make an easy addition to that script. Otherwise, this should be easily accomplished with a search/replace operation across files:

1. Replace `<substep` with `<step`
2. Replace `</substep` with `</step`

Though unlikely, any existing specializations of `<substeps>` outside of OASIS may require additional migration, but this might also be handled with relevant updates in the specialization module (leaving documents untouched). For example, if there is a specialization of `<substeps>` called `<littleSteps>`, the documents will not need migration. Instead, the declaration of the element's `@class` attribute will need to change so that the ancestry `task/substeps` becomes `task/steps`. The same change would be necessary for specializations of `<substep>`.

Might any existing processors or implementations need to change their expectations?

If a processor does something special with `<substeps>` that behavior will need to be updated to work with `<steps>` inside of `<steps>`, and will also need to ensure it can handle additional levels of nesting.

Might any existing specialization or constraint modules need to be migrated?

Potentially:

- If a specialization module specializes `<substeps>`, the element ancestry for that new element will need to be updated so that it includes `task/steps` rather than `task/substeps` (and an equivalent update for specializations of `<substep>`).
- If a specialization or constraint module explicitly allows or references either `<substeps>` or `<substep>`, that module will need to remove that reference.

Costs

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

Maintainers of the grammar files

Small cost of removing 2 element definitions and updating the content model of `<step>`

Editors of the DITA specification

- No new topics
- Need to remove 2 element reference topics

Vendors of tools

Low cost (if any) as described above.

DITA community-at-large

This will simplify the language (good impact!) but will result in an additional migration item for many task topics. Under the assumption that all topics will require some minimal migration regardless, this will present very little additional cost.

Producing migration instructions or tools

Migration instructions are simple, replacing 2 elements with 2 already known and similarly named elements. If tools already exist to migrate documents this should be added (at very low cost), but I do not anticipate tools explicitly for this.

Examples

Figure 20: Before/after of basic structure

Current markup for 3-level steps:

```
<steps>
  <step><cmd>Major command</cmd>
    <substeps>
      <substep><cmd>Sub-command</cmd>
        <info>
          <ol>
            <li>Long way to go, and is this still a command</li>
          </ol>
        </info>
      </substep>
    </substeps>
  </step>
</steps>
```

New markup for 3-level steps:

```
<steps>
  <step><cmd>Major command</cmd>
    <steps>
      <step><cmd>Sub-command</cmd>
        <steps>
          <step><cmd>Long way to go, and is this still a command</cmd></step>
        </steps>
      </step>
    </steps>
  </step>
</steps>
```

Figure 21: Example with unordered sub-steps

Currently any unordered sub-steps must establish an order (using the ordered `<substeps>`), or must be created with alternate markup:

```
<steps>
  <step><cmd>This command has some components</cmd>
    <info>
      <ul>
        <li>As you can see,</li>
        <li>the order is not important</li>
        <li>But I'd like to mark them up as steps / commands</li>
      </ul>
    </info>
  </step>
</steps>
```

With unordered steps allowed within a `<step>`, the correct markup can be used:

```
<steps>
  <step><cmd>This command has some components</cmd>
    <steps-unordered>
      <step><cmd>As you can see,</cmd></step>
      <step><cmd>the order is not important</cmd></step>
      <step><cmd>But I'd like to mark them up as steps / commands</cmd></step>
    </steps-unordered>
  </step>
</steps>
```

Figure 22: Reuse model

Currently the only ways to reuse a step from one task as a sub-step in another context are to either cut and paste, or to use content references on every component:

```
in topicA.dita:
<step id="stepZ">
  <cmd id="stepZcmd">command...</cmd>
  <stepxmp id="stepZxmp">example...</stepxmp>
  <info id="stepZinfo">more info...</info>
  <stepresult id="stepZresult">what you end up with...</stepresult>
</step>

in topicB.dita:
<substep>
  <cmd conkeyref="topicA/stepZcmd"/>
  <stepxmp conkeyref="topicA/stepZxmp"/>
  <info conkeyref="topicA/stepZinfo"/>
  <stepresult conkeyref="topicA/stepZresult"/>
</subtep>
```

This is ugly, difficult to maintain, and content will be broken if the original stepZ adds or removes markup (especially if it adds something like `<choices>` that is not even available in a sub-step). Allowing steps to

nest results in the following change; the reuse element still requires the empty <cmd> as with any content reference on an element with required children:

```
in topicA.dita:
<step id="stepZ">
  <cmd id="stepZcmd">command...</cmd>
  <stepxmp id="stepZxmp">example...</stepxmp>
  <info id="stepZinfo">more info...</info>
  <stepresult id="stepZresult">what you end up with...</stepresult>
</step>

in topicB.dita:
<step conkeyref="topicA/stepZ"><cmd/></step>
```

3 DITA 2.0: Stage three proposals

These proposals have been approved by the DITA TC at the stage three level. For more information about stage three criteria, see DITA 2.0 proposal process.

Unless otherwise noted, the proposals have been implemented in the DITA-2.0 branch of the following official GitHub repositories for the OASIS DITA TC:

- DITA Technical Committee
- DITA for Technical Communication subcommittee

3.1 Base edition

3.1.1 Stage 3: #08 Add <include> element

A new base <include> element for configuring the inclusion of text or markup, to be used as the new base element for <coderef>, <svgref> and <mathmlref>.

Champion

Chris Nitchie

Tracking information

Event	Date	Links
Stage 1 proposal accepted	July 19, 2016	Minutes
Stage 2 proposal submitted	February 25, 2018	HTML; DITA in SVN
Stage 2 proposal discussed	February 27, 2018 and March 6, 2018	Febrary 27, March 6
Stage 2 proposal approved	March 13, 2018	Minutes
Stage 3 proposal submitted to reviewers	June 5, 2018	Eliot Kimber, Robert Anderson
Stage 3 proposal (this document) submitted to TC	December 18, 2018	

Approved technical requirements

The <coderef> element is a transclusion element – it loads the referenced file and places its contents as CDATA at the location of the referencing element. However, <coderef> is a specialization of <xref>, which is *not* a transclusion element. That is, the specialization-based fallback behavior for <coderef> is fundamentally incompatible with the expected processing. The same is true for <svgref> and <mathmlref>. The only way for people other than the TC to create similar transclusion elements via specialization is to specialize from one of those, or customize their processors to do the right thing with their element.

Dependencies or interrelated proposals

This proposal reuses the <fallback> element from the proposal for Issue 27, the media domain.

Modified grammar files

Figure 23: commonElementsMod.rng

This proposal requires the addition of a new <include> element in the base vocabulary, defined as follows.

```
<div>
  <a:documentation>LONG NAME: Inclusion</a:documentation>
  <define name="include.content">
    <zeroOrMore>
      <ref name="data.elements.incl"/>
    </zeroOrMore>
    <optional>
      <ref name="fallback"/>
    </optional>
    <zeroOrMore>
      <ref name="foreign.unknown.incl"/>
    </zeroOrMore>
  </define>
  <define name="include.attributes">
    <optional>
      <attribute name="href"/>
    </optional>
    <optional>
      <attribute name="keyref"/>
    </optional>
    <optional>
      <attribute name="format"/>
    </optional>
    <optional>
      <attribute name="scope">
        <choice>
          <value>external</value>
          <value>local</value>
          <value>peer</value>
          <value>-dita-use-conref-target</value>
        </choice>
      </attribute>
    </optional>
    <optional>
      <attribute name="parse"/>
    </optional>
    <optional>
      <attribute name="encoding"/>
    </optional>
    <ref name="univ-atts"/>
  </define>
  <define name="include.element">
    <element name="include" dita:longName="Inclusion">
      <a:documentation>Use the inclusion (&lt;include>) element to transclude content stored
in another resource into a DITA topic. The
```

```
        @href and @keyref attributes specify the resource to be transcluded. The @parse
attribute declares the mode by which to include
        the content.Category: Body elements</a:documentation>
      <ref name="include.attlist"/>
      <ref name="include.content"/>
    </element>
  </define>
  <define name="include.attlist" combine="interleave">
    <ref name="include.attributes"/>
  </define>
</div>
```

This will also require a new entry in the "ELEMENT TYPE NAME PATTERNS" section:

```
<define name="include">
  <ref name="include.element"/>
</define>
```

This element will be added to the following named content model groups:

- `basic.ph`
- `basic.ph.notm`
- `fig.cnt`

The class attribute definition for `<include>` will also need to be added to the section at the bottom:

```
<define name="include.attlist" combine="interleave">
  <ref name="global-atts"/>
  <optional>
    <attribute name="class" a:defaultValue="- topic/include "/>
  </optional>
</define>
```

The specialization hierarchy for `<coderef>`, `<svgref>`, and `<mathmlref>` all need to be updated and their `@parse` attributes defaulted.

Figure 24: programmingDomain.rng

```
      <define name="coderef.attributes">
        <optional>
          <attribute name="href"/>
        </optional>
        <optional>
          <attribute name="keyref"/>
        </optional>
        <optional>
          <attribute name="type"/>
        </optional>
        <optional>
          <attribute name="format"/>
        </optional>
        <optional>
          <attribute name="parse" a:defaultValue="text"/>
        </optional>
        <optional>
          <attribute name="scope">
            <choice>
              <value>external</value>
              <value>local</value>
              <value>peer</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <ref name="univ-atts"/>
        <optional>
          <attribute name="outputclass"/>
```

```
      </optional>
    </define>
```

Later, in class declarations:

```
  <define name="coderef.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="+ topic/include pr-d/coderef "/>
    </optional>
  </define>
```

Figure 25: mathmlDomain.rng

```
    <define name="mathmlref.attributes">
      <optional>
        <attribute name="href"/>
      </optional>
      <optional>
        <attribute name="keyref"/>
      </optional>
      <optional>
        <attribute name="type"/>
      </optional>
      <optional>
        <attribute name="format" a:defaultValue="mml"/>
      </optional>
      <optional>
        <attribute name="parse" a:defaultValue="xml"/>
      </optional>
      <optional>
        <attribute name="scope">
          <choice>
            <value>external</value>
            <value>local</value>
            <value>peer</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <ref name="univ-atts"/>
      <optional>
        <attribute name="outputclass"/>
      </optional>
    </define>
```

Later, in class declarations:

```
  <define name="mathmlref.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="+ topic/include mathml-d/mathmlref "/>
    </optional>
  </define>
```

Figure 26: svgDomain.rng

```
    <define name="svgref.attributes">
      <optional>
        <attribute name="href"/>
      </optional>
      <optional>
        <attribute name="keyref"/>
      </optional>
      <optional>
        <attribute name="format" a:defaultValue="svg"/>
      </optional>
```

```
      <optional>
        <attribute name="parse" a:defaultValue="xml"/>
      </optional>
      <ref name="univ-atts"/>
      <optional>
        <attribute name="outputclass"/>
      </optional>
    </define>
```

Later, in class declarations:

```
  <define name="svgref.attlist" combine="interleave">
    <ref name="global-atts"/>
    <optional>
      <attribute name="class" a:defaultValue="+ topic/include svg-d/svgref "/>
    </optional>
  </define>
```

Modified terminology

None.

Modified specification documentation

There will be a new language reference topic describing the `<include>` element and, in particular, the extensibility of `@parse`. See below.

The `<coderef>`, `<svgref>`, and `<mathmlref>` reference topics will all need to be updated with the new specialization hierarchy. Since that is the only modification required for those topics, they are not included in full in this proposal.

The element-by-element recommendations for translators will also be updated with a new row for `<include>`:

Element name	Specialized from	Block/Inline (presentation)	Block/Inline (translation)	Translatable content?	Translatable attributes?
`<include>`	N/A	inline	inline	yes	

Migration plans for backwards incompatibilities

Any existing specializations of `<coderef>`, `<svgref>`, or `<mathmlref>` will need to be updated. Additionally, any content containing those elements with explicit `@class` attributes will likewise need to be updated to reflect the new specialization hierarchy.

Element reference topic: <include>

The inclusion element is an indicator that non-DITA content from a resource outside the current document should be placed at that location. The resource can be indicated using either a URI or a key reference. Processing expectations for the referenced data can also be specified.

Usage information

The `<include>` element can be used in many contexts, but is primarily intended for the following use cases.

- The transclusion of non-DITA XML within `<foreign>` or specializations of `<foreign>` using `parse="xml"`.

- The transclusion of preformatted textual content within <pre> or specializations of <pre> using parse="text".
- The transclusion of plain-text prose within DITA elements using parse="text".

It is an error when <include> is used to reference other DITA content. Authors should use @conref or @conkeyref for reuse of DITA XML content.

Formatting expectations

The <include> element should not normally be visible in processed output, though indications of its boundaries may be presented in authoring or debugging usage contexts. It should be replaced either by the content it references or by the contents of its <fallback> element.

Processing expectations

The <include> element instructs processors to insert the contents of the referenced resource at the location of the <include> element. If the content is unavailable to the processor or cannot be processed using the specified @parse value, the contents of the <fallback> element, if any, are presented instead. If the processor cannot process the referenced content using the rules implied by the @parse attribute, either because the referenced scheme is not supported or because there was a processing error, processors should issue a warning or error.

The @parse attribute specifies the processing expectations for the referenced resource. Processors must support the following values:

text

The contents should be treated as plain text. Reserved XML characters should be displayed, and not interpreted as XML markup.

xml

The contents of the referenced resource should be treated as an XML document, and the referenced element should be inserted at the location of the <include> element. If a fragment identifier is included in the address of the content, processors must select the element with the specified ID. If no fragment identifier is included, the root element of the referenced XML document is selected. Any grammar processing should be performed during resolution, such that default attribute values are explicitly populated. Prolog content must be discarded.

It is an error to use parse="xml" anywhere other than within <foreign> or a specialization thereof.

Processors may support other values for the @parse attribute with proprietary processing semantics. Processors should issue warnings and use <fallback> when they encounter unsupported @parse values. Non-standard @parse instructions should be expressed as URIs.

Note Proprietary @parse values will likely limit the portability and interoperability of DITA content, so should be used with care.

The @encoding attribute specifies the character encoding to use when translating the character data from the referenced content. If not specified, processors may make attempts to automatically determine the correct encoding, for example using HTTP headers, through analysis of the binary structure of the referenced data, or the <?xml?> processing instruction when including XML as text. The resource should be treated as UTF-8 if no other encoding information can be determined.

When `parse="xml"`, standard XML parsing rules apply for the detection of character encoding. The necessity and uses of `@encoding` for non-standard values of `@parse` are implementation-dependent.

Specialization hierarchy

- topic/include

Attributes

The following attributes are available on this element in addition to the Universal attribute group, Link relationship attribute group, and `@keyref`.

@parse
Indicates processing expectations for the referenced content.

@encoding
The encoding to use when interpreting textual data. The value should be a valid encoding name.

Examples

For the most part, `<include>` should be used as a basis for specialization. The following examples use it directly for purposes of illustration.

```
<!-- Inclusion of XML markup other than SVG or MathML -->
<fig>
  <title>JSP Tag Library Elements and Attributes</title>
  <foreign outputclass="tld">
    <include href="../src/main/webapp/WEB-INF/jsp-tag-library.tld"
             parse="xml" format="tld"/>
  </foreign>
</fig>

<!-- Inclusion of README text into a DITA topic, with fallback. -->
<shortdesc>
  <include href="../src/README.txt" parse="text" encoding="UTF-8">
    <fallback>This topic describes XYZ.</fallback>
  </include>
</shortdesc>

<!-- Inclusion of preformatted text -->
<pre>
  <include href="../src/config.json" format="json" parse="text" encoding="UTF-8"/>
</pre>

<!-- Inclusion of README as Markdown converted to DITA using a proprietary
     @parse value, with fallback. -->
<section>
  <include href="about.md" encoding="UTF-8"
    parse="http://www.example.com/dita/includeParsers/markdown-to-dita">
      <fallback>This section not available.</fallback>
  </include>
</section>

<!-- Proprietary vendor handling for CSV tables -->
<fig>
  <title>Data Table</title>
  <include href="data.csv"  encoding="UTF-8"
    parse="http://www.example.com/dita/includeParsers/csv-to-simpletable"/>
</fig>
```

3.1.2 Stage 3: #17 Make @outputclass universal

Make `@outputclass` a universal attribute on all elements defined as part of the DITA specification.

Champion

Robert D Anderson

Tracking information

Event	Date	Links
Stage 1 proposal accepted	Jan 24 2017	https://lists.oasis-open.org/archives/dita/201701/msg00097.html
Stage 2 proposal submitted	June 12 2018	• DITA version: https://tools.oasis-open.org/version-control/svn/dita/trunk/DITA-2.0/stage-2/Issue17-universalOutputclass.dita • HTML version: https://www.oasis-open.org/apps/org/workgroup/dita/download.php/60973/Issue17-universalOutputclass.html
Stage 2 proposal discussed	July 11 2018	https://www.oasis-open.org/committees/download.php/61199/minutes20170711.txt
Stage 2 proposal approved	July 18 2018	https://www.oasis-open.org/committees/document.php?document_id=61257&wg_abbrev=dita
Stage 3 proposal submitted to reviewers	February 28 2018	Sent to Dawn Stevens, Scott Hudson
Stage 3 proposal (this document) submitted	February 28 2018	https://tools.oasis-open.org/version-control/browse/wsvn/dita/trunk/DITA-2.0/stage-3/Issue17-stage3-universalOutputclass.dita

Approved technical requirements

Apart from the `<dita>` container element, add `@outputclass` to every element in the specification that does not already allow that attribute.

Dependencies or interrelated proposals

Any proposal for a new element in DITA 2.0 needs to ensure that `@outputclass` is allowed.

The original LwDITA proposal anticipated that `@outputclass` would be available everywhere but had to be adjusted when that was determined to be incorrect. A new version of LwDITA may want to make use of this attribute on those elements if it has not already done so.

Modified Grammar files

The `@outputclass` attribute should be declared for any element that does not already have it, as follows:

```
<optional><attribute name="outputclass"></optional>
```

The most reliable way to do this is to add it to the universal attribute group – assuming the other groups are not modified, this will make:

```
<define name="univ-atts">
      <ref name="id-atts"/>
      <ref name="select-atts"/>
      <ref name="localization-atts"/>
      <optional>
        <attribute name="outputclass">
      </optional>
    </define>
```

All modules that independently declare `@outputclass` will need to be updated, **EXCEPT** for those that also need to override `univ-atts` (for example, `<topic>` and its specializations must override `univ-atts` in order to make `@id` required). The attribute is currently declared in a large number of files, and can likely be removed with a good search/replace expression. The `@outputclass` attribute is currently declared in 46 independent RNG files from DITA 1.3, so this proposal effectively updates every module that declares elements to remove one or more instances of the line above. Those in the base package that will need updates include:

- `commonElementsMod.rng`
- `delayResolutionDomain.rng` (likely to be dropped from DITA 2.0)
- `ditavalrefDomain.rng`
- `hazardstatementDomain.rng`
- `highlightDomain.rng`
- `mapGroupDomain.rng`
- `tblDeclMod.rng`
- `topicMod.rng`
- `utilitiesDomain.rng`
- `classifyDomain.rng`
- `subjectSchemeMod.rng`

DTD updates are comparable. The following attribute declaration will be needed in the `univ-atts` declaration:

```
outputclass     CDATA      #IMPLIED
```

The same declaration will need to be removed from every element that currently uses it; as with RNG, this will effectively mean removing that same declaration from every module that declares elements.

Modified terminology

No new or updated terminology.

Modified specification documentation

- The `@outputclass` definition will move from the "Other common attributes" topic to the "Universal attribute group" topic.

- Every element reference topic that independently links to the `@outputclass` definition will need to remove that link. As in the technical requirements section, there is an exception for any element that overrides the universal attribute group, which will keep the link to `@outputclass` in its new location. 73 topics in `specification/langRef/base/` link to `@outputclass` and need to be updated. The attribute reuse topic also contains a few reusable definitions that link to `@outputclass`.

  Note For most DITA 2.0 proposals I'd expect this section to have a list of all modified topics. For this reason I don't think it is necessary, because:

  1. The change involves removing a single trivial item in many places, rather than adding any new language that must be reviewed.
  2. The change exactly the same in all 73 topics, so describing or explaining it 73 times will not aid in evaluating the proposal.
  3. Listing all 73 would also not be helpful for the spec editors; any editor implementing this change is going to do so with a quick search across the topics. That is, I would not expect any editor to make updates based on a list of 73 topics here (which I came up with using a search), because that process would be more time consuming and less reliable than doing that same search on the source.

Migration plans for backwards incompatibilities

Most structural or domain specialization modules that declare elements will need to be updated to remove declarations of `@outputclass`.

There is not currently a plan for an automated way to accomplish this for DTDs, due to the wide variety of coding practices for DTD modules. XSD and RNG could likely be automated with a search/replace algorithm that handles complex regular expressions, but this may result in removing the attribute from elements that do not use the universal attribute group.

Current suggestion for migration instructions is as follows (to be clarified with addition details based on our own experience updating the OASIS shells):

1. Use search/replace to remove all declarations of `@outputclass` from any working DITA 1.3 specialization modules. This can be done by hand with a search / manual removal or (depending on grammar format and coding practice) with a search/replace query that automatically removes the declaration.
2. Once the updated modules are added to DITA 2.0 level shells, use trang or a similar tool to convert the grammar files (DTD, XSD, or RNG) into a single RNG file.
3. Run a (to-be-written) script over that RNG to check for any elements that **do not** declare `@outputclass`. For any elements that do not declare the attribute, restore the declaration in your specialization module (in the DTD, RNG, or XSD – not in the single generated RNG).
4. The single generated RNG can be discarded once this process is completed.

Note As Dawn pointed out during her review of my first stage 3 draft, the definition of XML itself technically allows you to declare an attribute twice, but it's considered a warning. This could reduce the urgency of migration by some small amount because this change alone will not break anything. That said, the warnings you're likely to receive from your editors and build systems will likely push you to migrate quite quickly, and long term a migration is definitely worth the relatively small effort of removing declarations.

3.1.3 Stage 3: #18 Make @audience, @platform, @product, @otherprops into specializations

DITA 2.0 should refactor the existing profiling attributes – `@audience`, `@platform`, `@product`, and `@otherprops` – so that they're defined in domains and specialized from `@props`, as we did for `@deliveryTarget` in 1.3.

Champion

Championed by Robert D. Anderson, originally proposed by Chris Nitchie

Tracking information

Event	Date	Links
Stage 1 proposal accepted	2 May 2017	https://lists.oasis-open.org/archives/dita/201705/msg00015.html
Stage 2 proposal submitted	28 Feb 2018	HTML, DITA
Stage 2 proposal discussed	20 March 2018	https://lists.oasis-open.org/archives/dita/201803/msg00061.html
Stage 2 proposal approved	27 March 2018	https://www.oasis-open.org/apps/org/workgroup/dita/download.php/62798/minutes20180327.txt?referring_url=%2Fkws
Stage 3 proposal submitted to reviewers	25 Sept 2018	Deb Bissantz, Dawn Stevens
Stage 3 proposal (this document) submitted to TC	1 Oct 2018	DITA source in SVN

Approved technical requirements

1. Create four new base domain modules, one for each of `@audience`, `@platform`, `@product`, and `@otherprops`.
2. Remove the existing definitions of these attributes. In OASIS grammars, these are defined as base attributes in the `%filter-atts;` group along with `@props` and a placeholder for extensions of `@props`. This is the only place the attributes are named in OASIS grammars (DTD: `commonElements.mod`; RNG: `commonElementsMod.rng`).
3. Add the domain attributes into the shells of all OASIS-provided grammar files, and add them to the props-extension group in each configured shell. This will restore the attributes to any element that previously defined them.

Dependencies or interrelated proposals

N/A

Modified grammar files

commonElements.mod (before)

```
<!ENTITY % filter-atts
              "props
                          CDATA
                                       #IMPLIED
               platform
                          CDATA
                                       #IMPLIED
               product
                          CDATA
                                       #IMPLIED
               audience
                          CDATA
                                       #IMPLIED
               otherprops
                          CDATA
                                       #IMPLIED
               %props-attribute-extensions;"
>
```

commonElements.mod (after)

```
<!ENTITY % filter-atts
              "props
                          CDATA
                                       #IMPLIED
               platform
                          CDATA
                                       #IMPLIED
               product
                          CDATA
                                       #IMPLIED
               audience
                          CDATA
                                       #IMPLIED
               otherprops
                          CDATA
                                       #IMPLIED
               %props-attribute-extensions;"
>
```

All provided DTD shells (before)

```
<!ENTITY % deliveryTargetAtt-d-dec
  PUBLIC "-//OASIS//ENTITIES DITA 2.0
Delivery Target Attribute Domain//EN"
         "deliveryTargetAttDomain.ent"
>%deliveryTargetAtt-d-dec;
```

All provided DTD shells (after)

```
<!ENTITY % audienceAtt-d-dec
  PUBLIC "-//OASIS//ENTITIES DITA 2.0
Audience Attribute Domain//EN"
         "audienceAttDomain.ent"
>%audienceAtt-d-dec;
<!ENTITY % deliveryTargetAtt-d-dec
  PUBLIC "-//OASIS//ENTITIES DITA 2.0
Delivery Target Attribute Domain//EN"
         "deliveryTargetAttDomain.ent"
>%deliveryTargetAtt-d-dec;
<!ENTITY % platformAtt-d-dec
  PUBLIC "-//OASIS//ENTITIES DITA 2.0
Platform Attribute Domain//EN"
         "platformAttDomain.ent"
>%platformAtt-d-dec;
<!ENTITY % productAtt-d-dec
  PUBLIC "-//OASIS//ENTITIES DITA 2.0
Product Attribute Domain//EN"
         "productAttDomain.ent"
>%productAtt-d-dec;
<!ENTITY % otherpropsAtt-d-dec
  PUBLIC "-//OASIS//ENTITIES DITA 2.0
Otherprops Attribute Domain//EN"
         "otherpropsAttDomain.ent"
>%productAtt-d-dec;
```

```
<!ENTITY % props-attribute-extensions
  "%deliveryTargetAtt-d-attribute;"
>
```

```
<!ENTITY % props-attribute-extensions
  "%audienceAtt-d-attribute;
  %deliveryTargetAtt-d-attribute;
  %platformAtt-d-attribute;
  %productAtt-d-attribute;
  %otherpropsAtt-d-attribute;"
>
```

commonElements.rng (before)

```
<define name="filter-atts">
      <optional>
        <attribute name="props"/>
```

commonElements.rng (after)

```
<define name="filter-atts">
      <optional>
        <attribute name="props"/>
```

commonElements.rng (before)

```
      </optional>
      <optional>
        <attribute name="platform"/>
      </optional>
      <optional>
        <attribute name="product"/>
      </optional>
      <optional>
        <attribute name="audience"/>
      </optional>
      <optional>
        <attribute name="otherprops"/>
      </optional>
      <ref name="props-attribute-
extensions"/>
    </define>
```

commonElements.rng (after)

```
      </optional>
      <optional>
        <attribute name="platform"/>
      </optional>
      <optional>
        <attribute name="product"/>
      </optional>
      <optional>
        <attribute name="audience"/>
      </optional>
      <optional>
        <attribute name="otherprops"/>
      </optional>
      <ref name="props-attribute-
extensions"/>
    </define>
```

All provided RNG shells (before)

In the default domain declaration section:

```
<define name="domains-att">
         <optional>
            <attribute name="domains"
                       a:defaultValue="
                         ...content
domains...
                         a(props
deliveryTarget)"/>
         </optional>
      </define>
```

All provided RNG shells (after)

In the default domain declaration section:

```
<define name="domains-att">
         <optional>
            <attribute name="domains"
                       a:defaultValue="
                         ...content
domains...
                         a(props audience)
                         a(props
deliveryTarget)
                         a(props platform)
                         a(props product)
                         a(props
otherprops)"/>
         </optional>
      </define>
```

In the module inclusion section:

```
<include href="deliveryTargetAttDomain.rng"
dita:since="1.3"/>
```

In the module inclusion section:

```
<include href="audienceAttDomain.rng"
dita:since="2.0"/>
<include href="deliveryTargetAttDomain.rng"
dita:since="1.3"/>
<include href="platformAttDomain.rng"
dita:since="2.0"/>
<include href="productAttDomain.rng"
dita:since="2.0"/>
<include href="otherpropsAttDomain.rng"
dita:since="2.0"/>
```

Rather than make this proposal appear much longer than it should due to long headers + a few lines of meaning, I've uploaded the 4 new DTD domains and 4 new RNG domains to github; they can be reviewed at:

- https://github.com/robander/dita/blob/issue18/propertyatts/doctypes/dtd/base/dtd/audienceAttDomain.ent
- https://github.com/robander/dita/blob/issue18/propertyatts/doctypes/dtd/base/dtd/platformAttDomain.ent
- https://github.com/robander/dita/blob/issue18/propertyatts/doctypes/dtd/base/dtd/productAttDomain.ent

- https://github.com/robander/dita/blob/issue18/propertyatts/doctypes/dtd/base/dtd/otherpropsAttDomain.ent
- https://github.com/robander/dita/blob/issue18/propertyatts/doctypes/rng/base/rng/audienceAttDomain.rng
- https://github.com/robander/dita/blob/issue18/propertyatts/doctypes/rng/base/rng/platformAttDomain.rng
- https://github.com/robander/dita/blob/issue18/propertyatts/doctypes/rng/base/rng/productAttDomain.rng
- https://github.com/robander/dita/blob/issue18/propertyatts/doctypes/rng/base/rng/otherpropsAttDomain.rng

Modified terminology

N/A

Modified specification documentation

The following topic needs to be updated: http://docs.oasis-open.org/dita/dita/v1.3/errata02/os/complete/part1-base/langRef/attributes/metadataAttributes.html#select-atts

Before	After
@platform Indicates operating system and hardware. If no value is specified, but the attribute is specified on an ancestor within a map or within the related-links section, the value will cascade from the closest ancestor. **@product** Contains the name of the product to which the element applies. If no value is specified, but the attribute is specified on an ancestor within a map or within the related-links section, the value will cascade from the closest ancestor. **@audience** Indicates the intended audience for the element. If no value is specified, but the attribute is specified on an ancestor within a map or within the related-links section, the value will cascade from the closest ancestor. **@otherprops** This attribute can be used for any other properties that might be needed to describe an audience, or to provide selection criteria for the element. Alternatively, the `@props` attribute can be specialized to provide a new metadata attribute instead of using the general `@otherprops` attribute. If no value is specified, but the attribute is specified on an ancestor within a map or within the related-links section, the value will cascade from the closest ancestor. The `@audience`, `@platform`, `@product`, and `@otherprops` attributes are property attributes which	**Specializations of `@props`** The attributes `@audience`, `@deliveryTarget`, `@platform`, `@product`, and `@otherprops` are specialized from the `@props` attribute. They are defined in independent attribute extension domains, and integrated by default into all OASIS-provided document-type shells. If any of these domains is not integrated into a given document-type shell, the relevant attribute(s) will not be available. **@audience** Indicates the intended audience for the element. If no value is specified, but the attribute is specified on an ancestor within a map or within the related-links section, the value will cascade from the closest ancestor. **@deliveryTarget** Indicates the intended delivery target for the element; for example, "html", "pdf", or "epub". If no value is specified, but the attribute is specified on an ancestor within a map or within the related-links section, the value will cascade from the closest ancestor. **@platform** Indicates operating system and hardware. If no value is specified, but the attribute is specified on an ancestor within a map or within the related-links section, the value will cascade from the closest ancestor. **@product** Indicates the name of the product to which the element applies. If no value is specified, but the

Before	After
support conditional processing for filtering or flagging. Each takes a space-delimited set of values, with optional groups of space-delimited properties. Although these attributes are not specialized and not specializeable, the group syntax matches that for generalized attributes in *Attribute generalization*. **@deliveryTarget** The intended delivery target of the content, for example "html", "pdf", or "epub".	attribute is specified on an ancestor within a map or within the related-links section, the value will cascade from the closest ancestor. **@otherprops** This attribute can be used for any other properties that might be needed to describe an audience, or to provide selection criteria for the element. Alternatively, the `@props` attribute can be specialized to provide a new metadata attribute instead of using the general `@otherprops` attribute. If no value is specified, but the attribute is specified on an ancestor within a map or within the related-links section, the value will cascade from the closest ancestor.

In the topic on cascading map attributes: http://docs.oasis-open.org/dita/dita/v1.3/errata02/os/complete/part1-base/archSpec/base/cascading-of-attributes-from-map-to-map.html#cascading-of-attributes-from-map-to-map

Before	After
The following attributes cascade from map to map: • `@audience`, `@platform`, `@product`, `@otherprops`, `@rev` • `@props` and any attribute specialized from `@props` • ...	The following attributes cascade from map to map: • **`@rev`** • **`@props` and any attribute specialized from `@props` (including those integrated by default in OASIS shells: `@audience`, `@deliveryTarget`, `@platform`, `@product`, `@otherprops`)** • ...

From the topic on conditional processing, the `@product` attribute is missing from the current list: http://docs.oasis-open.org/dita/dita/v1.3/errata02/os/complete/part1-base/archSpec/base/condproc.html#condproc

Before	After
DITA defines attributes that can be used to enable filtering and flagging individual elements. The `@audience`, `@deliveryTarget`, `@otherprops`, `@platform`, and `@props` attributes (along with specializations of `@props`) allow conditions to be assigned to elements so that the elements can be included, excluded, or flagged during processing.	DITA defines attributes that can be used to enable filtering and flagging individual elements. The `@props` attribute and any attribute specialized from `@props` (including those integrated by default in OASIS shells: `@audience`, `@deliveryTarget`, `@platform`, `@product`, `@otherprops`) allow conditions to be assigned to elements so that the elements can be included, excluded, or flagged during processing.

Update to the list of common map attributes in http://docs.oasis-open.org/dita/dita/v1.3/errata02/os/complete/part1-base/archSpec/base/ditamap-attributes.html#ditamap-attributes

Before	After
The following metadata and reuse attributes are used by both DITA maps and DITA topics: • `@product`, `@platform`, `@audience`, `@otherprops`, `@rev`, `@status`, `@importance` • *...other atts...* • `@props` and any attribute specialized from `@props` (such as `@deliveryTarget`)	The following metadata and reuse attributes are used by both DITA maps and DITA topics: • `@rev`, `@status`, `@importance` • *...other atts...* • `@props` and any attribute specialized from `@props` (including those integrated by default in OASIS shells: `@audience`, `@deliveryTarget`, `@platform`, `@product`, `@otherprops`)

Remove a note in attribute generalization http://docs.oasis-open.org/dita/dita/v1.3/errata02/os/complete/part1-base/archSpec/base/generalization-attributes.html#attributegeneralize

Before	After
Note The `@audience`, `@platform`, `@product`, and `@otherprops` attributes allow grouped values that reuse the generalized syntax described here; however, these attributes are not specialized or specializeable. For these attributes, it can be typical to author or edit the grouped values directly.	*...removed...*

Updates needed in the topic on property attribute groups: http://docs.oasis-open.org/dita/dita/v1.3/errata02/os/complete/part1-base/archSpec/base/usage-of-conditional-processing-attributes.html#usage-of-conditional-processing-attributes

Before	After
Groups organize classification metadata into subcategories. This is intended to support situations where a predefined metadata attribute applies to multiple specialized subcategories. For example, the `@product` attribute can be used to classify an element based on both related databases and related application servers. Using groups for these subcategories allows each category to be processed independently; when filter conditions exclude all applicable databases within a group, the element can be safely excluded, regardless of any other `@product` conditions. Groups can be used within `@audience`, `@product`, `@platform`, or `@otherprops`. The following rules apply:	Groups can be used to organize classification metadata into subcategories. This is intended to support situations where a predefined metadata attribute applies to multiple specialized subcategories. The grouping syntax exactly matches the syntax used for generalized attributes, making it valid inside `@props` and any attribute specialized from `@props` (including those integrated by default in OASIS shells: `@audience`, `@deliveryTarget`, `@platform`, `@product`, `@otherprops`). For example, the `@product` attribute can be used to classify an element based on both related databases and related application servers. Using groups for these subcategories allows each category to be processed independently; when filter conditions exclude all applicable databases within a group, the element can be safely excluded, regardless of any other `@product` conditions. Groups can be used within `@props` and any attribute specialized from `@props` (including those integrated by default in OASIS shells: `@audience`,

Before	After
	`@deliveryTarget`, `@platform`, `@product`, `@otherprops`). The following rules apply:
Note While the grouped values reuse the generalized attribute syntax found in *Attribute generalization*, the `@audience`, `@product`, `@platform`, and `@otherprops` attributes cannot be specialized or generalized.	*...removed...*

In the definition of the DITAVAL `<prop>` element http://docs.oasis-open.org/dita/dita/v1.3/errata02/os/complete/part1-base/langRef/ditaval/ditaval-prop.html#ditaval-prop:

Before	After
The `<prop>` element identifies an attribute, and usually values in the attribute, to take an action on. The attribute either must be a conditional-processing attribute (`@platform`, `@product`, `@audience`, `@deliveryTarget`, `@props`, and `@otherprops`) or a specialization of the `@props` attribute.	The `<prop>` element identifies an attribute, and usually values in the attribute, to take an action on. The identified attribute is a conditional-processing attribute (either `@props` or a specialization of `@props`, such as `@audience`, `@deliveryTarget`, `@platform`, `@product`, or `@otherprops`).
@att The attribute to be acted upon. If using a literal attribute name, it must be one of `@props`, `@audience`, `@platform`, `@product`, `@otherprops`, `@deliveryTarget`, or a specialization of `@props`. Otherwise, the value should be the name of a group used within the `@audience`, `@platform`, `@product`, or `@otherprops` attributes. If the `@att` attribute is absent, then the `<prop>` element declares a default behavior for any conditional processing attribute.	**@att** The attribute to be acted upon. If using a literal attribute name, it is `@props` or a specialization of `@props` (such as `@audience`, `@deliveryTarget`, `@platform`, `@product`, or `@otherprops`). Otherwise, the value is the name of a group within one of those attributes, with the group name specified using the generalized attribute syntax. If the `@att` attribute is absent, then the `<prop>` element declares a default behavior for any conditional processing attribute.

Migration plans for backwards incompatibilities

- No documents will need to be migrated.
- Processors may optionally remove exceptions in any filter/flagging code that explicitly look for these attributes. As long as the processors correctly support `@props` and specializations, they will continue to work as designed even without migration.
- Document types that wish to retain all four attributes will need to add the four new domains into shells. When moving shells into a DITA 2.0 environment, documents may be updated manually using the tables above; the same changes made for OASIS provided DTD / RNG shells will work for local shells that are already using DITA 1.3 and use the `@deliveryTarget` module. (For a large number of document-type shells, a search/replace operation or script may be faster.) Shells that do not include `@deliveryTarget` will likely need to be updated manually (search and replace could also work well for those, but no other strings can be assumed to exist at the right spot in all shells).
- Constraints that *remove* any of these attributes today will need to be updated; it may be possible to discard the constraint module and simply not include the relevant domain(s). This will need to

be handled manually, because of the many ways it could potentially be done today, but searching over any modules for the four attributes should make them easy to find.

- Constraints that explicitly declare or restrict values on these attributes will need to be updated. This will need to be handled manually, because of the many ways it could potentially be done today, but searching over any modules for the four attributes should make them easy to find.

3.1.4 Stage 3: #27 Multimedia domain

A new domain for referencing multimedia content, specifically video and audio, modeled loosely on the HTML5 markup for the same purpose.

Champion

Chris Nitchie

Tracking information

Event	Date	Links
Stage 1 proposal accepted	May 16, 2017	Minutes
Stage 2 proposal submitted	August 17, 2017	HTML, DITA
Stage 2 proposal discussed	August 22, 2017	Minutes
Stage 2 proposal approved	August 29, 2017	Minutes
Stage 3 proposal submitted to reviewers	April 7, 2018	Eliot Kimber, Bill Burns
Stage 3 proposal (this document) submitted to TC	May 21, 2018	

Approved technical requirements

DITA should have an HTML5-compatible multimedia domain to simplify authoring references to audio and video information. Also, since Lightweight DITA will contain multimedia elements, it's important that they be accounted for in full DITA.

Dependencies or interrelated proposals

None.

Modified grammar files

This proposal requires the addition of a new `<fallback>` element in the base vocabulary, used within `<object>` to specify content to be rendered if the referenced object cannot be rendered or resolved. The definition for that element is as follows:

```
<div>
  <a:documentation xml:space="preserve">LONG NAME: Fallback

    Fallback is used in the context of object to provide content
    when an object reference cannot be rendered.
        </a:documentation>
  <define name="fallback.content">
    <zeroOrMore>
      <choice>
        <text/>
```

```
        <ref name="basic.block.notbfgobj"/>
        <ref name="basic.ph"/>
        <ref name="data.elements.incl"/>
        <ref name="draft-comment"/>
        <ref name="foreign.unknown.incl"/>
        <ref name="required-cleanup"/>
      </choice>
    </zeroOrMore>
  </define>
  <define name="fallback.attributes">
    <ref name="univ-atts"/>
  </define>
  <define name="fallback.element">
    <element name="fallback" dita:longName="Fallback">
      <a:documentation>The &lt;fallback> element contains content to be rendered when a
multimedia
        object reference cannot be rendered.</a:documentation>
      <ref name="fallback.attlist"/>
      <ref name="fallback.content"/>
    </element>
  </define>
  <define name="fallback.attlist" combine="interleave">
    <ref name="fallback.attributes"/>
  </define>
</div>
```

This will require modifying the `object.content` definition as follows:

```
<define name="object.content">
  <optional>
    <ref name="desc"/>
  </optional>
  <optional>
    <ref name="longdescref"/>
  </optional>
  <optional>
    <ref name="fallback"/>
  </optional>
  <zeroOrMore>
    <ref name="param"/>
  </zeroOrMore>
  <zeroOrMore>
    <ref name="foreign.unknown.incl"/>
  </zeroOrMore>
</define>
```

The lion's share of the changes for this proposal will be contained in a new domain file, `mediaDomain.rng`, defined as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="urn:oasis:names:tc:dita:rng:vocabularyModuleDesc.rng"
                          schematypens="http://relaxng.org/ns/structure/1.0"?>
<grammar xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
  xmlns:dita="http://dita.oasis-open.org/architecture/2005/"
  xmlns="http://relaxng.org/ns/structure/1.0">
  <moduleDesc xmlns="http://dita.oasis-open.org/architecture/2005/">
    <moduleTitle>DITA Media Domain</moduleTitle>
    <headerComment xml:space="preserve"><![CDATA[
 ============================================================
                    HEADER
 ============================================================
  MODULE:    DITA Media Domain
  VERSION:   2.0
  DATE:      April 2018

 ============================================================

 ============================================================
 ]]></headerComment>
    <moduleMetadata>
      <moduleType>elementdomain</moduleType>
```

```
      <moduleShortName>media-d</moduleShortName>
      <modulePublicIds>
        <dtdMod>-//OASIS//ELEMENTS DITA<var presep=" " name="ditaver"/> Media Domain//EN</
dtdMod>
        <dtdEnt>-//OASIS//ENTITIES DITA<var presep=" " name="ditaver"/> Media Domain//EN</
dtdEnt>
        <xsdMod>urn:oasis:names:tc:dita:xsd:mediaDomain.xsd<var presep=":" name="ditaver"/></
xsdMod>
        <rncMod>urn:oasis:names:tc:dita:rnc:mediaDomain.rnc<var presep=":" name="ditaver"/></
rncMod>
        <rngMod>urn:oasis:names:tc:dita:rng:mediaDomain.rng<var presep=":" name="ditaver"/></
rngMod>
      </modulePublicIds>
      <domainsContribution>(topic media-d)</domainsContribution>
    </moduleMetadata>
  </moduleDesc>
  <div>
    <a:documentation>DOMAIN EXTENSION PATTERNS</a:documentation>
    <define name="media-d-object">
      <choice>
        <ref name="video.element"/>
        <ref name="audio.element"/>
      </choice>
    </define>
    <define name="object" combine="choice">
      <ref name="media-d-object"/>
    </define>
  </div>

  <div>
    <a:documentation xml:space="preserve"> LONG NAME: Video object reference </
a:documentation>
    <define name="video.content">
      <optional>
        <ref name="desc"/>
      </optional>
      <optional>
        <ref name="longdescref"/>
      </optional>
      <optional>
        <ref name="fallback"/>
      </optional>
      <optional>
        <ref name="video-poster"/>
      </optional>
      <optional>
        <ref name="media-controls"/>
      </optional>
      <optional>
        <ref name="media-autoplay"/>
      </optional>
      <optional>
        <ref name="media-loop"/>
      </optional>
      <optional>
        <ref name="media-muted"/>
      </optional>
      <zeroOrMore>
        <ref name="media-source"/>
      </zeroOrMore>
      <zeroOrMore>
        <ref name="media-track"/>
      </zeroOrMore>
      <zeroOrMore>
        <ref name="param"/>
      </zeroOrMore>
      <zeroOrMore>
        <ref name="foreign.unknown.incl"/>
      </zeroOrMore>
    </define>
    <define name="video.attributes">
      <optional>
        <attribute name="data"/>
      </optional>
```

```
      <optional>
        <attribute name="datakeyref"/>
      </optional>
      <optional>
        <attribute name="type"/>
      </optional>
      <optional>
        <attribute name="height">
          <data type="NMTOKEN"/>
        </attribute>
      </optional>
      <optional>
        <attribute name="width">
          <data type="NMTOKEN"/>
        </attribute>
      </optional>
      <optional>
        <attribute name="tabindex">
          <data type="NMTOKEN"/>
        </attribute>
      </optional>
      <ref name="univ-atts"/>
      <optional>
        <attribute name="outputclass"/>
      </optional>
    </define>
    <define name="video.element">
      <element name="video" dita:longName="Video object reference">
        <a:documentation>DITA's &lt;video> element corresponds to the HTML &lt;video> element.
          Category: Body elements</a:documentation>
        <ref name="video.attlist"/>
        <ref name="video.content"/>
      </element>
    </define>
    <define name="video.attlist" combine="interleave">
      <ref name="video.attributes"/>
    </define>
  </div>

  <div>
    <a:documentation xml:space="preserve"> LONG NAME: Audo object reference </a:documentation>
    <define name="audio.content">
      <optional>
        <ref name="desc"/>
      </optional>
      <optional>
        <ref name="longdescref"/>
      </optional>
      <optional>
        <ref name="fallback"/>
      </optional>
      <optional>
        <ref name="media-controls"/>
      </optional>
      <optional>
        <ref name="media-autoplay"/>
      </optional>
      <optional>
        <ref name="media-loop"/>
      </optional>
      <optional>
        <ref name="media-muted"/>
      </optional>
      <zeroOrMore>
        <ref name="media-source"/>
      </zeroOrMore>
      <zeroOrMore>
        <ref name="media-track"/>
      </zeroOrMore>
      <zeroOrMore>
        <ref name="param"/>
      </zeroOrMore>
      <zeroOrMore>
        <ref name="foreign.unknown.incl"/>
```

```
      </zeroOrMore>
    </define>
    <define name="audio.attributes">
      <optional>
        <attribute name="data"/>
      </optional>
      <optional>
        <attribute name="datakeyref"/>
      </optional>
      <optional>
        <attribute name="type"/>
      </optional>
      <optional>
        <attribute name="tabindex">
          <data type="NMTOKEN"/>
        </attribute>
      </optional>
      <ref name="univ-atts"/>
      <optional>
        <attribute name="outputclass"/>
      </optional>
    </define>
    <define name="audio.element">
      <element name="audio" dita:longName="Audio object reference">
        <a:documentation>DITA's &lt;audio> element corresponds to the HTML &lt;audio> element.
          Category: Body elements</a:documentation>
        <ref name="audio.attlist"/>
        <ref name="audio.content"/>
      </element>
    </define>
    <define name="audio.attlist" combine="interleave">
      <ref name="audio.attributes"/>
    </define>
  </div>

  <div>
    <a:documentation xml:space="preserve"> LONG NAME: Media control configuration </
a:documentation>
    <define name="media-controls.content">
      <empty/>
    </define>
    <define name="media-controls.attributes">
      <optional>
        <attribute name="name" a:defaultValue="controls">
          <value>controls</value>
        </attribute>
      </optional>
      <optional>
        <attribute name="value" a:defaultValue="true">
          <choice>
            <value>true</value>
            <value>false</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <ref name="univ-atts"/>
    </define>
    <define name="media-controls.element">
      <element name="media-controls" dita:longName="Audio object reference">
        <a:documentation>DITA's &lt;media-controls> element corresponds to the HTML @media
attribute on &lt;video> and &lt;audio> elements.</a:documentation>
        <ref name="media-controls.attlist"/>
        <ref name="media-controls.content"/>
      </element>
    </define>
    <define name="media-controls.attlist" combine="interleave">
      <ref name="media-controls.attributes"/>
    </define>
  </div>

  <div>
    <a:documentation xml:space="preserve"> LONG NAME: Media autoplay configuration </
a:documentation>
```

```
    <define name="media-autoplay.content">
      <empty/>
    </define>
    <define name="media-autoplay.attributes">
      <optional>
        <attribute name="name" a:defaultValue="autoplay">
          <value>autoplay</value>
        </attribute>
      </optional>
      <optional>
        <attribute name="value" a:defaultValue="true">
          <choice>
            <value>true</value>
            <value>false</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <ref name="univ-atts"/>
    </define>
    <define name="media-autoplay.element">
      <element name="media-autoplay" dita:longName="Audio object reference">
        <a:documentation>DITA's &lt;media-autoplay> element corresponds to the HTML @autoplay
attribute on &lt;video> and &lt;audio> elements.</a:documentation>
        <ref name="media-autoplay.attlist"/>
        <ref name="media-autoplay.content"/>
      </element>
    </define>
    <define name="media-autoplay.attlist" combine="interleave">
      <ref name="media-autoplay.attributes"/>
    </define>
  </div>

  <div>
    <a:documentation xml:space="preserve"> LONG NAME: Media loop configuration </
a:documentation>
    <define name="media-loop.content">
      <empty/>
    </define>
    <define name="media-loop.attributes">
      <optional>
        <attribute name="name" a:defaultValue="loop">
          <value>loop</value>
        </attribute>
      </optional>
      <optional>
        <attribute name="value" a:defaultValue="true">
          <choice>
            <value>true</value>
            <value>false</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <ref name="univ-atts"/>
    </define>
    <define name="media-loop.element">
      <element name="media-loop" dita:longName="Audio object reference">
        <a:documentation>DITA's &lt;media-loop> element corresponds to the HTML @loop
attribute on &lt;video> and &lt;audio> elements.</a:documentation>
        <ref name="media-loop.attlist"/>
        <ref name="media-loop.content"/>
      </element>
    </define>
    <define name="media-loop.attlist" combine="interleave">
      <ref name="media-loop.attributes"/>
    </define>
  </div>

  <div>
    <a:documentation xml:space="preserve"> LONG NAME: Media mute configuration </
a:documentation>
    <define name="media-muted.content">
      <empty/>
```

```
    </define>
    <define name="media-muted.attributes">
      <optional>
        <attribute name="name" a:defaultValue="muted">
          <value>muted</value>
        </attribute>
      </optional>
      <optional>
        <attribute name="value" a:defaultValue="true">
          <choice>
            <value>true</value>
            <value>false</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <ref name="univ-atts"/>
    </define>
    <define name="media-muted.element">
      <element name="media-muted" dita:longName="Audio object reference">
        <a:documentation>DITA's &lt;media-muted> element corresponds to the HTML @muted
attribute on &lt;video> and &lt;audio> elements.</a:documentation>
        <ref name="media-muted.attlist"/>
        <ref name="media-muted.content"/>
      </element>
    </define>
    <define name="media-muted.attlist" combine="interleave">
      <ref name="media-muted.attributes"/>
    </define>
  </div>

  <div>
    <a:documentation xml:space="preserve"> LONG NAME: Media source </a:documentation>
    <define name="media-source.content">
      <empty/>
    </define>
    <define name="media-source.attributes">
      <optional>
        <attribute name="name" a:defaultValue="source">
          <value>source</value>
        </attribute>
      </optional>
      <optional>
        <attribute name="value"/>
      </optional>
      <optional>
        <attribute name="valuetype" a:defaultValue="ref">
          <value>ref</value>
        </attribute>
      </optional>
      <optional>
        <attribute name="type"/>
      </optional>
      <optional>
        <attribute name="keyref"/>
      </optional>
      <ref name="univ-atts"/>
    </define>
    <define name="media-source.element">
      <element name="media-source" dita:longName="Audio object reference">
        <a:documentation>DITA's &lt;media-source> element corresponds to the HTML &lt;source>
element within &lt;video> and &lt;audio> elements.</a:documentation>
        <ref name="media-source.attlist"/>
        <ref name="media-source.content"/>
      </element>
    </define>
    <define name="media-source.attlist" combine="interleave">
      <ref name="media-source.attributes"/>
    </define>
  </div>

  <div>
    <a:documentation xml:space="preserve"> LONG NAME: Media track </a:documentation>
    <define name="media-track.content">
```

```
      <empty/>
    </define>
    <define name="media-track.attributes">
      <optional>
        <attribute name="name" a:defaultValue="track">
          <value>track</value>
        </attribute>
      </optional>
      <optional>
        <attribute name="value"/>
      </optional>
      <optional>
        <attribute name="valuetype" a:defaultValue="ref">
          <value>ref</value>
        </attribute>
      </optional>
      <optional>
        <attribute name="type">
          <choice>
            <value>subtitles</value>
            <value>captions</value>
            <value>descriptions</value>
            <value>chapters</value>
            <value>metadata</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="keyref"/>
      </optional>
      <ref name="univ-atts"/>
    </define>
    <define name="media-track.element">
      <element name="media-track" dita:longName="Audio object reference">
        <a:documentation>DITA's &lt;media-track> element corresponds to the HTML &lt;track>
element within &lt;video> and &lt;audio> elements.</a:documentation>
        <ref name="media-track.attlist"/>
        <ref name="media-track.content"/>
      </element>
    </define>
    <define name="media-track.attlist" combine="interleave">
      <ref name="media-track.attributes"/>
    </define>
  </div>

  <div>
    <a:documentation xml:space="preserve"> LONG NAME: Video poster </a:documentation>
    <define name="video-poster.content">
      <empty/>
    </define>
    <define name="video-poster.attributes">
      <optional>
        <attribute name="name" a:defaultValue="poster">
          <value>poster</value>
        </attribute>
      </optional>
      <optional>
        <attribute name="value"/>
      </optional>
      <optional>
        <attribute name="valuetype" a:defaultValue="ref">
          <value>ref</value>
        </attribute>
      </optional>
      <optional>
        <attribute name="type"/>
      </optional>
      <optional>
        <attribute name="keyref"/>
      </optional>
      <ref name="univ-atts"/>
    </define>
    <define name="video-poster.element">
```

```
      <element name="video-poster" dita:longName="Audio object reference">
        <a:documentation>DITA's &lt;video-poster> element corresponds to the HTML &lt;track>
element within &lt;video> and &lt;audio> elements.</a:documentation>
        <ref name="video-poster.attlist"/>
        <ref name="video-poster.content"/>
      </element>
    </define>
    <define name="video-poster.attlist" combine="interleave">
      <ref name="video-poster.attributes"/>
    </define>
  </div>

  <div>
    <a:documentation> Specialization attributes. Global attributes and class defaults </
a:documentation>
    <define name="video.attlist" combine="interleave">
      <ref name="global-atts"/>
      <optional>
        <attribute name="class"
          a:defaultValue="+ topic/object media-d/video "/>
      </optional>
    </define>
    <define name="audio.attlist" combine="interleave">
      <ref name="global-atts"/>
      <optional>
        <attribute name="class"
          a:defaultValue="+ topic/object media-d/audio "/>
      </optional>
    </define>
    <define name="media-controls.attlist" combine="interleave">
      <ref name="global-atts"/>
      <optional>
        <attribute name="class"
          a:defaultValue="+ topic/param media-d/media-controls "/>
      </optional>
    </define>
    <define name="media-autoplay.attlist" combine="interleave">
      <ref name="global-atts"/>
      <optional>
        <attribute name="class"
          a:defaultValue="+ topic/param media-d/media-autoplay "/>
      </optional>
    </define>
    <define name="media-loop.attlist" combine="interleave">
      <ref name="global-atts"/>
      <optional>
        <attribute name="class"
          a:defaultValue="+ topic/param media-d/media-loop "/>
      </optional>
    </define>
    <define name="media-muted.attlist" combine="interleave">
      <ref name="global-atts"/>
      <optional>
        <attribute name="class"
          a:defaultValue="+ topic/param media-d/media-muted "/>
      </optional>
    </define>
    <define name="video-poster.attlist" combine="interleave">
      <ref name="global-atts"/>
      <optional>
        <attribute name="class"
          a:defaultValue="+ topic/param media-d/video-poster "/>
      </optional>
    </define>
    <define name="media-source.attlist" combine="interleave">
      <ref name="global-atts"/>
      <optional>
        <attribute name="class"
          a:defaultValue="+ topic/param media-d/media-source "/>
      </optional>
    </define>
    <define name="media-track.attlist" combine="interleave">
      <ref name="global-atts"/>
      <optional>
```

```
        <attribute name="class"
          a:defaultValue="+ topic/param media-d/media-track "/>
      </optional>
    </define>
  </div>
</grammar>
```

Finally, references to this new module will need to be added to all of the base document type shells.

- `base/rng/basemap.rng`
- `base/rng/basetopic.rng`

For instance:

```
<include href="mediaDomain.rng"/>
```

Modified terminology

None.

Modified specification documentation

Changes to the architectural specification will be limited to the element-by-element recommendations to translators topic, `non-normative/elementsMerged.dita`. A subset of this topic is reproduced below with the new content.

The changes to the language reference are as follows:

- Additions to `langRef/quick-reference/base-elements-a-to-z.dita` for each new element; not presented here.
- Updates to the generated content model topics, not presented here.
- The `<object>` topic (`langRef/base/object.dita`) will be modified such that examples will include the use of `<fallback>`. See the table below.
- A new topic will appear in the Body elements section (3.2.2) describing the `<fallback>` element; see below.
- A new topic describing the media domain will be added under the Domain Elements section (3.5); see below.
- Beneath the new domain topic will be a new topic for each element in the domain; see below.

Table 1: Updates to Existing Topics

File	DITA 1.3 Language	DITA 2.0 Language
langRef/base/object.dita	Cutting the keys from the system unit: <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,0,0" data="cutkey370.swf" type="application/x-shockwave-flash" height="280" width="370" id="cutkey370"> <desc>A description of the task</desc> <param name="movie" value="cutkey370.swf"/> <param name="quality" value="high"/> <param name="bgcolor" value="#FFFFFF"/> </object>	Cutting the keys from the system unit: <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,0,0" data="cutkey370.swf" type="application/x-shockwave-flash" height="280" width="370" id="cutkey370"> <desc>A description of the task</desc> <fallback>Media not available.</fallback> <param name="movie" value="cutkey370.swf"/> <param name="quality" value="high"/> <param name="bgcolor" value="#FFFFFF"/> </object>
	What's EIM? <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,0,0" data="eim.swf" height="400" width="500" id="eim"> <desc>Some great, glorious info</desc> <param name="movie" value="eim.swf"/> <param name="quality" value="high"/> <param name="bgcolor" value="#FFFFFF"/> <param name="pluginspace" value="http://www.macromedia.com/go/getflashplayer"/> </object>	What's EIM? <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,0,0" data="eim.swf" height="400" width="500" id="eim"> <desc>Some great, glorious info</desc> <fallback><image href="media-not-available.png"/></fallback> <param name="movie" value="eim.swf"/> <param name="quality" value="high"/> <param name="bgcolor" value="#FFFFFF"/> <param name="pluginspace" value="http://www.macromedia.com/go/getflashplayer"/> </object>

File	DITA 1.3 Language	DITA 2.0 Language
langRef/base/object.dita	<object id="E5123_026.mp4" width="300" height="300"> <param name="poster" **keyref="E5123_026_poster"** /> <param name="source" **keyref="E5123_026_video"** /> </object>	<object id="E5123_026.mp4" width="300" height="300"> <fallback>Media not available.</fallback> <param name="poster" **keyref="E5123_026_poster"** /> <param name="source" **keyref="E5123_026_video"** /> </object>
langRef/base/object.dita	<object **classidkeyref="video_classid"** **codebasekeyref="video_codebase"** **datakeyref="cutkey370"** height="280" width="370" id="cutkey370"> <desc>A description of the task</desc> <param name="movie" **keyref="cutkey370"**/> <param name="quality" value="high"/> <param name="bgcolor" value="#FFFFFF"/> </object>	<object **classidkeyref="video_classid"** **codebasekeyref="video_codebase"** **datakeyref="cutkey370"** height="280" width="370" id="cutkey370"> <desc>A description of the task</desc> <fallback>Media not available.</fallback> <param name="movie" **keyref="cutkey370"**/> <param name="quality" value="high"/> <param name="bgcolor" value="#FFFFFF"/> </object>

Migration plans for backwards incompatibilities

N/A

Element-by-element recommendations for translators: Base edition

topic elements

Note The vast majority of this topic is omitted; only the new row in the **topic elements** table and the new section for **media-d elements** are presented.

Element name	Specialized from	Block/Inline (presentation)	Block/Inline (translation)	Translatable content?	Translatable attributes?
<fallback>	N/A	block	block	yes	

media-d elements (media domain) *(new in DITA 2.0)*

Element name	Specialized from	Inherits everything from ancestor?	Block/Inline (presentation)	Block/Inline (translation)	Translatable content?	Translatable attributes?
<video>	<object>	no	block	block	yes	
<audio>	<object>	no	block	block	yes	
<media-controls>	<param>	no	block	block	n/a	
<media-autoplay>	<param>	no	block	block	n/a	
<media-loop>	<param>	no	block	block	n/a	
<media-muted>	<param>	no	block	block	n/a	
<video-poster>	<param>	no	block	block	n/a	
<media-source>	<param>	no	block	block	n/a	
<media-track>	<param>	no	block	block	n/a	

Media elements

The media elements are used to reference audio or video content. The elements in this domain are modeled loosely on the HTML5 <audio> and <video> elements, allowing for extensive configuration of media in multiple formats with supplemental external resources.

Figure 27: Relationship between HTML5 and DITA markup

The elements in the media domain can be mapped to HTML5 multimedia elements as follows.

DITA Markup	Analogous HTML5 Markup
<video>	<video>
<audio>	<audio>
<desc>	@title
@data or @datakeyref	@src
<media-autoplay>	@autoplay
<media-controls>	@controls
<media-loop>	@loop
<media-muted>	@muted
<video-poster>	@poster
<media-source>	<source>
<media-track>	<track>

DITA Markup	Analogous HTML5 Markup
`@value` or `@keyref` on `<media-source>` and `<media-track>`	`@src` on `<source>` and `<track>`
`@type` on `<media-track>`	`@kind` on `<track>`
`@xml:lang` on `@media-track`	`@srclang` on `<track>`
`<fallback>`	Other markup directly within `<audo>` or `<video>`

Element reference topic: <fallback>

The `<fallback>` element provides content to be presented when multimedia objects cannot be rendered.

Usage information

Comment by chris.nitchie
I think this is covered by the shortdesc? I don't really have anything to add here.

Formatting expectations

The contents of this element should be displayed only when the media referenced by the containing `<object>` element cannot be displayed.

Processing expectations

None.

Specialization hierarchy

- topic/fallback

Attributes

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

Example

See `<object>` for examples of this element.

Element reference topic: <video>

The `<video>` element is used to reference video content in a DITA topic. It is modeled roughly on the HTML5 element of the same name.

Usage information

Media can be referenced either using the `@data` or `@datakeyref` attributes or using nested `<media-source>` elements. Extensive configuration of the presentation of the media can be configured using optional nested elements.

Formatting expectations

When possible, the video should be displayed at the location of the `<video>` reference. If `@width` and/or `@height` are specified, those dimensions should be honored. When the media cannot be presented in a meaningful way, the contents of the `<fallback>` element, if any, should be presented.

Processing expectations

When converting to HTML5 or HTML5-derived output formats, `<video>` should be mapped to its HTML5-equivalent markup. Processing for other output formats is highly dependent on the capabilities of those formats. Implementers should implement support for presentation of multimedia on a best-effort basis.

Specialization hierarchy

+ topic/object media-d/video

Attributes

The following attributes are available on this element: Universal attributes group and the attributes defined below.

@data
: Contains the absolute or relative URI of the video media. If this attribute is specified, `@type` should also be specified. If the video is available in multiple formats, nested `<media-source>` elements should be used instead.

@datakeyref
: Key reference to the location of the video data, as for `@data`. When specified and the key is resolvable, the key-provided URI is used. A key that has no associated resource, only link text, is considered to be unresolved for the purpose of the `@datakeyref` attribute. If `@data` is specified, it is used as a fallback when the key cannot be resolved to a resource.

@type
: Indicates the content type (MIME type) for the data specified by the `@data` or `@datakeyref` attribute. This attribute should be set when the `@data` attribute is set to avoid loading unsupported content types. Note that this differs from the `@type` attribute on many other DITA elements in that it specifies a MIME type rather than a content type. If `@type` is not specified, the effective type value for the key named by the `@datakeyref` attribute is used as the this attribute's value.

@height
: Indicates the vertical dimension for the resulting video display. If necessary, the object is scaled to the specified size. The value of this attribute is a real number (expressed in decimal notation) optionally followed by a unit of measure from the set of pc, pt, px, in, cm, mm, em (picas, points, pixels, inches, centimeters, millimeters, and ems respectively). The default unit is px (pixels). Possible values include: "5", "5in", and "10.5cm". If a height value is specified and no width value is specified, the width will be scaled by the same factor as the height. If both a height value and width value are specified, implementations **MAY** ignore one of the two values when they are unable to scale to each direction using different factors.

@width
: Indicates the horizontal dimension for the resulting video display. If necessary, the object is scaled to the specified size. If both a height value and width value are specified, implementations **MAY** ignore one of the two values when they are unable to scale to each direction using different factors. If a width value is specified and no height value is specified, the height will be scaled by the same factor as the width. If both a height value and width value are specified, implementations **MAY** ignore one of the two values when they are unable to scale to each direction using different factors.

@tabindex
Position the video in tabbing order.

A very simple video reference

```
<video data="video.mp4" type="video/mp4"/>
```

A similar example using a key reference:

```
<video datakeyref="video" type="video/mp4"/>
```

Referencing a video in multiple formats

```
<video>
  <media-source keyref="video-mp4" value="video.mp4" type="video/mp4"/>
  <media-source keyref="video-ogg" value="video.ogg" type="video/ogg"/>
  <media-source keyref="video-webm" value="video.webm" type="video/webm"/>
</video>
```

Complete example configuring video presentation, with multiple formats and multi-lingual subtitles

The following video reference explicitly defines multiple presentational details for a video available in multiple formats, referenced using key references, with URIs when the keys are not resolvable.

```
<video width="400px" height="300px">
  <desc>A video illustrating this procedure.</desc>
  <fallback>
    <image keyref="video-not-available" href="video-not-available.png">
      <alt>This video cannot be displayed.</alt>
    </image>
  </fallback>

  <!-- Reference the poster via both key and URI -->
  <video-poster keyref="video-poster" value="poster.png"/>

  <!--
    The default value is 'true', so @value is optional on these controls
    when enabling their features, but mandatory when disabling them.
  -->
  <media-controls value="true"/>
  <media-autoplay/>
  <media-loop value="false"/>
  <media-muted value="false"/>

  <!-- Multiple formats, referenced via both key and URI -->
  <media-source keyref="video-mp4" value="video.mp4" type="video/mp4"/>
  <media-source keyref="video-ogg" value="video.ogg" type="video/ogg"/>
  <media-source keyref="video-webm" value="video.webm" type="video/webm"/>

  <!-- Subtitle tracks in multiple languages -->
  <media-track xml:lang="en" keyref="video-subtitles-en"
    value="video-subtitles-en.vtt" type="subitles"/>
  <media-track xml:lang="fr" keyref="video-subtitles-fr"
    value="video-subtitles-fr.vtt" type="subitles"/>
  <media-track xml:lang="de" keyref="video-subtitles-de"
    value="video-subtitles-de.vtt" type="subitles"/>
</video>
```

Element reference topic: <audio>

The <audio> element is used to reference sound content in a DITA topic. It is modeled roughly on the HTML5 element of the same name.

Usage information

Media can be referenced either using the @data or @datakeyref attributes or using nested <media-source> elements. Extensive configuration of the presentation of the media can be configured using optional nested elements.

Formatting expectations

Presentation of the <audio> element will vary depending on the presentation format. When possible, HTML-based outputs should present the audio object using the analogous HTML <audio> element. When the media cannot be presented in a meaningful way, the contents of the <fallback> element, if any, should be presented.

Processing expectations

When converting to HTML5 or HTML5-derived output formats, <audio> should be mapped to its HTML5-equivalent markup. Processing for other output formats is highly dependent on the capabilities of those formats. Implementers should implement support for the presentation of multimedia on a best-effort basis.

Specialization hierarchy

+ topic/object media-d/audio

Attributes

The following attributes are available on this element: Universal attributes group and the attributes defined below.

@data
: Contains the absolute or relative URI of the audio media. If this attribute is specified, @type should also be specified. If the audio is available in multiple formats, nested <media-source> elements should be used instead.

@datakeyref
: Key reference to the location of the audio data, as for @data. When specified and the key is resolvable, the key-provided URI is used. A key that has no associated resource, only link text, is considered to be unresolved for the purpose of the @datakeyref attribute. If @data is specified, it is used as a fallback when the key cannot be resolved to a resource.

@type
: Indicates the content type (MIME type) for the data specified by the @data or @datakeyref attribute. This attribute should be set when the @data attribute is set to avoid loading unsupported content types. Note that this differs from the @type attribute on many other DITA elements in that it specifies a MIME type rather than a content type. If @type is not specified, the effective type value for the key named by the @datakeyref attribute is used as the this attribute's value.

@tabindex
: Position the audio in tabbing order.

A very simple audio object reference

```
<audio data="message.mp3" type="audio/mp3"/>
```

A similar example using a key reference:

```
<audio datakeyref="message" type="audio/mp3"/>
```

Referencing an audio object in multiple formats

```
<audio>
  <media-source keyref="message-mp3" value="message.mp3" type="audio/mp3"/>
  <media-source keyref="message-wav" value="message.wav" type="audio/wav"/>
</audio>
```

Complete example configuring audio presentation, with multiple formats

The following video reference explicitly defines multiple presentational details for a video available in multiple formats, referenced using key references, with URIs when the keys are not resolvable.

```
<audio>
  <desc>A sound file narrating the execution of this procedure.</desc>
  <fallback>The audio track walking through this procedure is not available.</fallback>

  <!--
    The default value is 'true', so @value is optional on these controls
    when enabling their features, but mandatory when disabling them.
  -->
  <media-controls value="true"/>
  <media-autoplay/>
  <media-loop value="false"/>
  <media-muted value="false"/>

  <!-- Multiple formats, referenced via both key and URI -->
  <media-source keyref="walkthrough-mp3" value="walkthrough.mp3" type="audio/mp3"/>
  <media-source keyref="walkthrough-wav" value="walkthrough.wav" type="audio/wav"/>
</audio>
```

Element reference topic: <media-controls>

The `<media-controls>` element controls whether or not User Interface controls should be presented for controlling the playback of the media referenced by the enclosing `<video>` or `<audio>` element. It is analogous to the `@controls` attribute on HTML5 media elements.

Usage information

The boolean `@value` attribute specifies whether the controls should be presented. The default value for this attribute is `true`, so the mere presence of this element within `<video>` or `<audio>` is sufficient to turn the setting on. The `@value` attribute must be explicitly set to `false` to specify that controls should not be presented.

If this element is not present, the default behavior is determined by the user agent being used to present the media.

Formatting expectations

None. This element configures the behavior when the media is displayed.

Processing expectations

None.

Specialization hierarchy

+ topic/param media-d/media-controls

Attributes

The following attributes are available on this element: Universal attributes group and the attributes defined below.

@value
Specifies whether the media presentation should include user interface controls. The following values are recognized:

true
Default. Controls should be presented.

false
Controls should not be presented.

-dita-use-conref-target
See Using the -dita-use-conref-target value for more information.

@name
This attribute is required due to this element's derivation from the `<param>` element. The value is fixed to "controls" by the grammar definition.

Example

See `<video>` and `<audio>` for examples of this element.

Element reference topic: <media-autoplay>

The `<media-autoplay>` element controls whether or not the media referenced by the surrounding `<video>` or `<audio>` element should begin playing immediately when the topic is displayed. It is analogous to the `@autoplay` attribute on HTML5 media elements.

Usage information

The boolean `@value` attribute specifies whether the media should automatically begin playing. The default value for this attribute is `true`, so the mere presence of this element within `<video>` or `<audio>` is sufficient to turn the setting on. The `@value` attribute must be explicitly set to `false` to disable automatic starting.

If this element is not present, the default behavior is determined by the user agent being used to present the media.

Formatting expectations

None. This element configures the behavior when the media is displayed.

Processing expectations

None.

Specialization hierarchy

+ topic/param media-d/media-autoplay

Attributes

The following attributes are available on this element: Universal attributes group and the attributes defined below.

@value
Specifies whether the media should automatically begin playing when the topic is displayed. The following values are recognized:

true
Default. Autoplaying is enabled.

false
Autoplay is disabled.

-dita-use-conref-target
See Using the -dita-use-conref-target value for more information.

@name
This attribute is required due to this element's derivation from the `<param>` element. The value is fixed to "autoplay" by the grammar definition.

Example

See `<video>` and `<audio>` for examples of this element.

Element reference topic: <media-loop>

The `<media-loop>` element controls whether or not the media referenced by the surrounding `<video>` or `<audio>` element should restart automatically when it completes playing. It is analogous to the `@loop` attribute on HTML5 media elements.

Usage information

The boolean `@value` attribute specifies whether the media should loop. The default value for this attribute is `true`, so the mere presence of this element within `<video>` or `<audio>` is sufficient to turn the setting on. The `@value` attribute must be explicitly set to `false` to disable looped playback.

If this element is not present, the default behavior is determined by the user agent being used to present the media.

Formatting expectations

None. This element configures the behavior when the media is displayed.

Processing expectations

None.

Specialization hierarchy

+ topic/param media-d/media-loop

Attributes

The following attributes are available on this element: Universal attributes group and the attributes defined below.

@value
Specifies whether the media should loop when played. The following values are recognized:

true
Default. Looped playback is enabled.

false
Playback will not be looped.

-dita-use-conref-target
See Using the -dita-use-conref-target value for more information.

@name
This attribute is required due to this element's derivation from the `<param>` element. The value is fixed to "loop" by the grammar definition.

Example

See `<video>` and `<audio>` for examples of this element.

Element reference topic: <media-muted>

The `<media-muted>` element controls whether or not the media referenced by the surrounding `<video>` or `<audio>` element should play without sound. It is analogous to the `@muted` attribute on HTML5 media elements.

Usage information

The boolean `@value` attribute specifies whether the media should be muted. The default value for this attribute is `true`, so the mere presence of this element within `<video>` or `<audio>` is sufficient to mute playback. The `@value` attribute must be explicitly set to `false` to unmute playback by default.

If this element is not present, the default behavior is determined by the user agent being used to present the media.

Formatting expectations

None. This element configures the behavior when the media is displayed.

Processing expectations

None.

Specialization hierarchy

+ topic/param media-d/media-muted

Attributes

The following attributes are available on this element: Universal attributes group and the attributes defined below.

@value
Specifies whether the media should be muted. The following values are recognized:

true
Default. Playback will be muted by default.

false
Playback will not be muted.

-dita-use-conref-target
See Using the -dita-use-conref-target value for more information.

@name
This attribute is required due to this element's derivation from the `<param>` element. The value is fixed to "muted" by the grammar definition.

Example

See `<video>` and `<audio>` for examples of this element.

Element reference topic: <video-poster>

The `<video-poster>` element specifies an image to display in place of the video referenced by the enclosing `<video>` element before playback begins. It is analogous to the `@poster` attribute on the HTML5 `<video>` element.

Usage information

The image to use is specified using the `@value` or `@keyref` attribute.

If this element is not present, the default behavior is determined by the user agent being used to present the video.

Formatting expectations

None. This element configures the behavior when the media is displayed.

Processing expectations

None.

Specialization hierarchy

+ topic/param media-d/video-poster

Attributes

The following attributes are available on this element: Universal attributes group and the attributes defined below.

@value
Specifies the URL of the image.

@keyref
A key reference to the image. If both `@keyref` and `@value` are specified, the image referenced by `@keyref` takes precedence unless the key cannot be resolved, in which case the resource specified by `@value` is used as a fallback.

@type
Specifies the content type (MIME type) of the image specified by `@value`.

@name
This attribute is required due to this element's derivation from the `<param>` element. The value is fixed to "poster" by the grammar definition.

@valuetype
This attribute is present due to this element's derivation from the `<param>` element. The value is fixed to "ref" by the grammar definition.

Example

See `<video>` for examples of this element.

Element reference topic: <media-source>

The `<media-source>` element specifies the location of a resource containing a representation of the media being referenced by the enclosing `<video>` or `<audio>` element in a particular format. It is analogous to the `<source>` element used in HTML5 multimedia elements.

Usage information

The resource location is specified using `@value` or `@keyref` attribute. The content type (MIME type) of the content is specified using the `@type` attribute.

Formatting expectations

Not applicable.

Processing expectations

None.

Specialization hierarchy

+ topic/param media-d/media-source

Attributes

The following attributes are available on this element: Universal attributes group and the attributes defined below.

@value
Specifies the URL of the media resource.

@keyref
A key reference to the media resource. If both `@keyref` and `@value` are specified, the resource referenced by `@keyref` takes precedence unless the key cannot be resolved, in which case the resource specified by `@value` is used as a fallback.

@type
Specifies the content type (MIME type) of the resource specified by `@value`.

@name
This attribute is required due to this element's derivation from the `<param>` element. The value is fixed to "source" by the grammar definition.

@valuetype
This attribute is present due to this element's derivation from the `<param>` element. The value is fixed to "ref" by the grammar definition.

Example

See `<video>` and `<audio>` for examples of this element.

Element reference topic: <media-track>

The `<media-track>` element specifies the location of a resource containing a supplemental timed text track or other time-based data for the media referenced by the enclosing `<video>` or `<audio>` element. It is analogous to the `<track>` element used in HTML5 multimedia elements.

Usage information

Track resources should be in Web Video Text Track Format (WebVTT). The resource location is specified using `@value` or `@keyref` attribute. The kind of track is specified by the `@type` attribute. The language of the track is specified using the `@xml:lang` attribute.

Formatting expectations

The display of the track information is determined by the user agent being used to present the media.

Processing expectations

None.

Specialization hierarchy

+ topic/param media-d/media-source

Attributes

The following attributes are available on this element: Universal attributes group and the attributes defined below.

@value
Specifies the URL of the track resource.

@keyref
A key reference to the track resource. If both `@keyref` and `@value` are specified, the resource referenced by `@keyref` takes precedence unless the key cannot be resolved, in which case the resource specified by `@value` is used as a fallback.

@type
Specifies the usage for the referenced track data. This attribute is analogous to the `@kind` attribute on the HTML5 `<track>` element as described by the *W3C HTML5 specification*. The values recognized for this attribute are derived from the HTML5 standard, from which the below descriptions are copied:

subtitles
Transcription or translation of the dialogue, suitable for when the sound is available but not understood (e.g. because the user does not understand the language of the media resource's soundtrack). Displayed over the video.

captions
Transcription or translation of the dialogue, sound effects, relevant musical cues, and other relevant audio information, suitable for when the soundtrack is unavailable (e.g. because it is muted or because the user is deaf). Displayed over the video; labeled as appropriate for the hard-of-hearing.

descriptions
Textual descriptions of the video component of the media resource, intended for audio synthesis when the visual component is unavailable (e.g. because the user is interacting with the application without a screen while driving, or because the user is blind). Synthesized as separate audio track.

chapters
Chapter titles, intended to be used for navigating the media resource. Displayed as an interactive list in the user agent's interface.

metadata
Tracks intended for use from script. Not displayed by the user agent.

-dita-use-conref-target
See Using the -dita-use-conref-target value for more information.

@name
This attribute is required due to this element's derivation from the `<param>` element. The value is fixed to "track" by the grammar definition.

@valuetype
This attribute is present due to this element's derivation from the `<param>` element. The value is fixed to "ref" by the grammar definition.

Example

See `<video>` and `<audio>` for examples of this element.

3.1.5 Stage 3: #36 Remove deprecated elements and attributes

Remove elements and attributes that have been designated as deprecated, "reserved for future use," or defined by mistake and retained only to maintain backwards compatibility

Champion

Kristen James Eberlein, Eberlein Consulting LLC

Tracking information

Event	Date	Links
Stage 1 proposal accepted	6 June 2017	Minutes of TC meeting
Stage 2 proposal submitted	1 February 2018; updated on 19 February 2018 and 26 February 2018	• DITA • HTML: – 01 – 02 – 03
Stage 2 proposal discussed		Minutes of TC meetings:

Event	Date	Links
		• 06 February 2018 • 13 February 2018 • 20 February 2018
Stage 2 proposal approved	27 February 2018	Minutes of TC meeting: 27 February 2018
Stage 3 proposal submitted to reviewers	27 February 2018; revised and resubmitted to reviewers on 1 March 2018 (based on feedback from Robert Anderson); revised and resubmitted to reviewers on 2 March 2018 (based on feedback from Stan Doherty).	• Robert Anderson • Stan Doherty • Nancy Harrison • Keith Schengili-Roberts
Stage 3 proposal (this document) submitted to TC	2 March 2018	

Approved technical requirements

Remove elements and attributes that have been designated as deprecated, "reserved for future use," or defined by mistake and retained only to maintain backwards compatibility

Dependencies or interrelated proposals

None

Modified grammar files

The following files must be modified:

DTDs

- `commonElements.mod`
- `map.mod`
- `mapGroup.mod`
- `topic.mod`

RNG

- `commonElementsMod.rng`
- `mapMod.rng`
- `mapGroupMod.rng`
- `topicMod.rng`

In the content below, the following conventions are used:

- Bold is used to indicate code to be added.
- Line-through is used to indicate code to be removed.

- Ellipses indicate where code is snipped for brevity.

Figure 28: Removing <boolean>

commonElements.mod

```
<!ENTITY % basic.ph
               "%boolean; |
                %cite; |
                %keyword; |
                %ph; |
                %q; |
                %term; |
                %text; |
                %tm; |
                %xref; |
                %state;"
>
...
<!ENTITY % basic.ph.noxref.nocite
               "%boolean; |
                %keyword; |
                %ph; |
                %q; |
                %term; |
                %text; |
                %tm; |
                %state;"
>
...
<!ENTITY % basic.ph.notm
               "%boolean; |
                %cite; |
                %keyword; |
                %ph; |
                %q; |
                %term; |
                %text; |
                %xref; |
                %state;"
>

<!--                        LONG NAME: Boolean  (deprecated)                 -->
<!ENTITY % boolean.content
                          "EMPTY"
>
<!ENTITY % boolean.attributes
               "state
                                (no |
                                yes |
                                -dita-use-conref-target)
                                          #REQUIRED
               %univ-atts;
               outputclass
                                CDATA
                                          #IMPLIED"
>
<!ELEMENT  boolean %boolean.content;>
<!ATTLIST  boolean %boolean.attributes;>

...
<!ATTLIST  boolean       %global-atts;  class CDATA "- topic/boolean "     >
```

Figure 29: Removing <boolean>

commonElementsMod.rng

```
 <div>
    <a:documentation>ELEMENT TYPE NAME PATTERNS</a:documentation>
    ...
    <define name="boolean">
```

```
      <ref name="boolean.element"/>
    </define>
    ...
<define name="basic.ph">
        <choice>
          <ref name="boolean"/>
          <ref name="cite"/>
          <ref name="keyword"/>
          <ref name="ph"/>
          <ref name="q"/>
          <ref name="term"/>
          <ref name="text" dita:since="1.3"/>
          <ref name="tm"/>
          <ref name="xref"/>
          <ref name="state"/>
        </choice>
      </define>
...
<define name="basic.ph.noxref.nocite" dita:since="1.3">
        <choice>
          <ref name="boolean"/>
          <ref name="keyword"/>
          <ref name="ph"/>
          <ref name="q"/>
          <ref name="term"/>
          <ref name="text" dita:since="1.3"/>
          <ref name="tm"/>
          <ref name="state"/>
        </choice>
      </define>
...
<define name="basic.ph.notm">
        <choice>
          <ref name="boolean"/>
          <ref name="cite"/>
          <ref name="keyword"/>
          <ref name="ph"/>
          <ref name="q"/>
          <ref name="term"/>
          <ref name="text" dita:since="1.3"/>
          <ref name="xref"/>
          <ref name="state"/>
        </choice>
      </define>
...
<div>
      <a:documentation>LONG NAME: Boolean (deprecated)</a:documentation>
      <define name="boolean.content">
        <empty/>
      </define>
      <define name="boolean.attributes">
        <attribute name="state">
          <choice>
            <value>no</value>
            <value>yes</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
        <ref name="univ-atts"/>
        <optional>
          <attribute name="outputclass"/>
        </optional>
      </define>
      <define name="boolean.element">
        <element name="boolean" dita:longName="Boolean  (deprecated)">
          <a:documentation>The &lt;boolean> element is used to express one of two opposite values, such as yes or no, on or off, true or false, high or low, and so forth. The element itself is
            empty; the value of the element is stored in its state attribute, and the semantic associated with the value is typically in a specialized name derived from this element. Category:
            Specialization elements</a:documentation>
          <ref name="boolean.attlist"/>
          <ref name="boolean.content"/>
```

```
          </element>
        </define>
        <define name="boolean.attlist" combine="interleave">
          <ref name="boolean.attributes"/>
        </define>

      </div>
...
    <define name="boolean.attlist" combine="interleave">
      <ref name="global-atts"/>
      <optional>
        <attribute name="class" a:defaultValue="- topic/boolean "/>
      </optional>
    </define>
```

Figure 30: Removing <indextermref>

commonElements.mod

```
<!ENTITY % txt.incl
              "%draft-comment; |
               %fn; |
               %indextermref; |
               %indexterm; |
               %required-cleanup;"
>
...
<!--                      LONG NAME: Index term reference                 -->
<!ENTITY % indextermref.content
                         "EMPTY"
>
<!ENTITY % indextermref.attributes
              "keyref
                           CDATA
                                        #REQUIRED
              %univ-atts;"
>
<!ELEMENT  indextermref %indextermref.content;>
<!ATTLIST  indextermref %indextermref.attributes;>
...
<!ATTLIST  indextermref %global-atts;  class CDATA "- topic/indextermref ">
```

Figure 31: Removing <indextermref>

commonElementsMod.rng

```
 <div>
    <a:documentation>ELEMENT TYPE NAME PATTERNS</a:documentation>
    ...
    <define name="indextermref">
      <ref name="indextermref.element"/>
    </define>
    ...
<define name="txt.incl">
        <a:documentation>Inclusions: defined sets that can be added into appropriate models</
a:documentation>
        <choice>
          <ref name="draft-comment"/>
          <ref name="fn"/>
          <ref name="indextermref"/>
          <ref name="indexterm"/>
          <ref name="required-cleanup"/>
        </choice>
      </define>
...
<div>
        <a:documentation>LONG NAME: Index term reference</a:documentation>
        <define name="indextermref.content">
          <empty/>
        </define>
        <define name="indextermref.attributes">
```

```
          <attribute name="keyref"/>
          <ref name="univ-atts"/>
        </define>
        <define name="indextermref.element">
          <element name="indextermref" dita:longName="Index term reference">
            <a:documentation>This element is not completely defined, and is reserved for
future use. Category: Miscellaneous elements</a:documentation>
            <ref name="indextermref.attlist"/>
            <ref name="indextermref.content"/>
          </element>
        </define>
        <define name="indextermref.attlist" combine="interleave">
          <ref name="indextermref.attributes"/>
        </define>
</div>
...
    <define name="indextermref.attlist" combine="interleave">
      <ref name="global-atts"/>
      <optional>
        <attribute name="class" a:defaultValue="- topic/indextermref "/>
      </optional>
    </define>
```

Figure 32: Removing @alt and @longdescref on <image>

commonElements.mod

```
<!ENTITY % image.attributes
             "href
                        CDATA
                                  #IMPLIED
              scope
                        (external |
                         local |
                         peer |
                         -dita-use-conref-target)
                                  #IMPLIED
              keyref
                        CDATA
                                  #IMPLIED
              alt
                        CDATA
                                  #IMPLIED
              longdescref
                        CDATA
                        #IMPLIED
              height
                        NMTOKEN
                                  #IMPLIED
              width
                        NMTOKEN
                                  #IMPLIED
              align
                        CDATA
                                  #IMPLIED
              scale
                        NMTOKEN
                                  #IMPLIED
              scalefit
                        (yes |
                         no |
                         -dita-use-conref-target)
                                  #IMPLIED
              placement
                        (break |
                         inline |
                         -dita-use-conref-target)
                                  'inline'
              %univ-atts;
              outputclass
                        CDATA
```

```
                                    #IMPLIED"
>
```

Figure 33: Removing @alt and @longdescref on <image>

commonElementsMod.rng

```
 <define name="image.attributes">
          <optional>
            <attribute name="href"/>
          </optional>
          <optional>
            <attribute name="scope">
              <choice>
                <value>external</value>
                <value>local</value>
                <value>peer</value>
                <value>-dita-use-conref-target</value>
              </choice>
            </attribute>
          </optional>
          <optional>
            <attribute name="keyref"/>
          </optional>
          <optional>
            <attribute name="alt"/>
          </optional>
          <optional>
            <attribute name="longdescref"/>
          </optional>
          <optional>
            <attribute name="height">
              <data type="NMTOKEN"/>
            </attribute>
          </optional>
          <optional>
            <attribute name="width">
              <data type="NMTOKEN"/>
            </attribute>
          </optional>
          <optional>
            <attribute name="align"/>
          </optional>
          <optional>
            <attribute name="scale">
              <data type="NMTOKEN"/>
            </attribute>
          </optional>
          <optional>
            <attribute name="scalefit">
              <choice>
                <value>yes</value>
                <value>no</value>
                <value>-dita-use-conref-target</value>
              </choice>
            </attribute>
          </optional>
          <optional>
            <attribute name="placement" a:defaultValue="inline">
              <choice>
                <value>break</value>
                <value>inline</value>
                <value>-dita-use-conref-target</value>
              </choice>
            </attribute>
          </optional>
          <optional>
            <attribute name="format" dita:since="1.3 errata 02"/>
          </optional>
          <ref name="univ-atts"/>
          <optional>
            <attribute name="outputclass"/>
```

```
          </optional>
        </define>
```

Figure 34: Removing @chunk="to-navigation"

No changes are needed to the grammar files.

Figure 35: Removing @collection-type="tree" on <linkpool> and <linklist>

The only changes required are to the DTDs.

`topic.mod`

```
<!ENTITY % linklist.attributes
              "collection-type
                          (choice |
                           family |
                           sequence |
                           unordered |
                           -dita-use-conref-target-|
-----------------------------tree)
                                    #IMPLIED
               duplicates
                          (no |
                           yes |
                           -dita-use-conref-target)
                                    #IMPLIED
               mapkeyref
                          CDATA
                                    #IMPLIED
               %relational-atts;
               %univ-atts;
               spectitle
                          CDATA
                                    #IMPLIED
               outputclass
                          CDATA
                                    #IMPLIED"
>
```

```
<!ENTITY % linkpool.attributes
              "collection-type
                          (choice |
                           family |
                           sequence |
                           unordered |
                           -dita-use-conref-targe-t-|
-----------------------------tree)
                                    #IMPLIED
               duplicates
                          (no |
                           yes |
                           -dita-use-conref-target)
                                    #IMPLIED
               mapkeyref
                          CDATA
                                    #IMPLIED
               %relational-atts;
               %univ-atts;
               outputclass
                          CDATA
                                    #IMPLIED"
>
```

Figure 36: Removing @collection-type on <reltable> and <relcolspec>

`map.mod`

Adding a new attribute group:

```
<!ENTITY % topicref-atts-for-reltable
              "type
                           CDATA
                                    #IMPLIED
               cascade
                           CDATA
                                    #IMPLIED
               processing-role
                           (normal |
                            resource-only |
                            -dita-use-conref-target)
                                    #IMPLIED
               scope
                           (external |
                            local |
                            peer |
                            -dita-use-conref-target)
                                    #IMPLIED
               locktitle
                           (no |
                            yes |
                            -dita-use-conref-target)
                                    #IMPLIED
               format
                           CDATA
                                    #IMPLIED
               linking
                           (none |
                            normal |
                            sourceonly |
                            targetonly |
                            -dita-use-conref-target)
                                    #IMPLIED
               toc
                           (no |
                            yes |
                            -dita-use-conref-target)
                                    'no'
               print
                           (no |
                            printonly |
                            yes |
                            -dita-use-conref-target)
                                    #IMPLIED
               search
                           (no |
                            yes |
                            -dita-use-conref-target)
                                    #IMPLIED
               chunk
                           CDATA
                                    #IMPLIED"
>
```

Modifying declarations for <reltable> and <relcolspec>:

```
<!ENTITY % reltable.attributes
              "title
                           CDATA
                                    #IMPLIED
               outputclass
                           CDATA
                                    #IMPLIED
               %topicref-atts-no-toc-no-keyscope;
               %topicref-atts-for-reltable;
               %univ-atts;"
>
...
<!ENTITY % relcolspec.attributes
              "outputclass
```

```
                            CDATA
                                        #IMPLIED
                  %topicref-atts-no-toc-no-keyscope;
                  %topicref-atts-for-reltable;
                  %univ-atts;"
>
```

Figure 37: Removing @collection-type on <reltable> and <relcolspec>

`mapMod.rng`

Adding a new attribute group for use on `<reltable>` and `<relcolspec>`:

```
<define name="topicref-atts-for-reltable" dita:since="2.0">
      <optional>
        <attribute name="type"/>
      </optional>
      <optional>
        <attribute name="cascade" dita:since="1.3"/>
      </optional>
      <optional>
        <attribute name="processing-role">
          <choice>
            <value>normal</value>
            <value>resource-only</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="scope">
          <choice>
            <value>external</value>
            <value>local</value>
            <value>peer</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="locktitle">
          <choice>
            <value>no</value>
            <value>yes</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="format"/>
      </optional>
      <optional>
        <attribute name="linking">
          <choice>
            <value>none</value>
            <value>normal</value>
            <value>sourceonly</value>
            <value>targetonly</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="toc" a:defaultValue="no">
          <choice>
            <value>no</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="print">
```

```
          <choice>
            <value>no</value>
            <value>printonly</value>
            <value>yes</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="search">
          <choice>
            <value>no</value>
            <value>yes</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="chunk"/>
      </optional>
    </define>
```

`mapMod.rng`

Modifying declarations for `<reltable>` and `<relcolspec>`; changes indicated in bold.

```
<define name="reltable.attributes">
        <optional>
          <attribute name="title"/>
        </optional>
        <optional>
          <attribute name="outputclass"/>
        </optional>
        <ref name="topicref-atts-no-toc-no-keyscope"/>
        <ref name="topicref-atts-for-reltable"/>
        <ref name="univ-atts"/>
      </define>
...
<define name="relcolspec.attributes">
        <optional>
          <attribute name="outputclass"/>
        </optional>
        <ref name="topicref-atts-no-toc-no-keyscope"/>
        <ref name="topicref-atts-for-reltable"/>
        <ref name="univ-atts"/>
      </define>
```

Figure 38: Removing @keyref on <navref>

`map.mod`

```
<!ENTITY % navref.attributes
              "%univ-atts;
                keyref
                                CDATA
                                                #IMPLIED
               mapref
                                CDATA
                                                #IMPLIED
               outputclass
                                CDATA
```

```
                                    #IMPLIED"
>
```

Figure 39: Removing @keyref on <navref>

`mapMod.rng`

```
<define name="navref.attributes">
        <ref name="univ-atts"/>
        <optional>
          <attribute name="keyref"/>
        </optional>
        <optional>
          <attribute name="mapref"/>
        </optional>
        <optional>
          <attribute name="outputclass"/>
        </optional>
      </define>
```

Figure 40: Removing @locktitle on <topichead> and <topicgroup>

`map.mod`

Add a new attribute entity

```
<!ENTITY % topicref-atts-no-locktitle
              "collection-type
                          (choice |
                           family |
                           sequence |
                           unordered |
                           -dita-use-conref-target)
                                    #IMPLIED
               type
                          CDATA
                                    #IMPLIED
               cascade
                          CDATA
                                    #IMPLIED
               processing-role
                          (normal |
                           resource-only |
                           -dita-use-conref-target)
                                    #IMPLIED
               scope
                          (external |
                           local |
                           peer |
                           -dita-use-conref-target)
                                    #IMPLIED
               format
                          CDATA
                                    #IMPLIED
               linking
                          (none |
                           normal |
                           sourceonly |
                           targetonly |
                           -dita-use-conref-target)
                                    #IMPLIED
               toc
                          (no |
                           yes |
                           -dita-use-conref-target)
                                    #IMPLIED
               print
                          (no |
                           printonly |
                           yes |
                           -dita-use-conref-target)
```

```
                                    #IMPLIED
               search
                          (no |
                           yes |
                           -dita-use-conref-target)
                                    #IMPLIED
               chunk
                          CDATA
                                    #IMPLIED
               keyscope
                          CDATA
                                    #IMPLIED"
>
```

mapGroup.mod

```
<!ENTITY % topichead.attributes
              "navtitle
                          CDATA
                                    #IMPLIED
               outputclass
                          CDATA
                                    #IMPLIED
               keys
                          CDATA
                                    #IMPLIED
               copy-to
                          CDATA
                                    #IMPLIED
               %topicref-atts;
               %topicref-atts-no-locktitle;
               %univ-atts;"
>
...
<!ENTITY % topicgroup.attributes
              "outputclass
                          CDATA
                                    #IMPLIED
               %topicref-atts;
               %topicref-atts-no-locktitle;
               %univ-atts;"
>
```

Figure 41: Removing @locktitle on <topichead> and <topicgroup>

mapMod.rng

Add a new attribute entity

```
<define name="topicref-atts-no-locktitle">
      <optional>
        <attribute name="collection-type">
          <choice>
            <value>choice</value>
            <value>family</value>
            <value>sequence</value>
            <value>unordered</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="type"/>
      </optional>
      <optional>
        <attribute name="cascade" dita:since="1.3"/>
      </optional>
      <optional>
        <attribute name="processing-role">
          <choice>
            <value>normal</value>
```

```
            <value>resource-only</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="scope">
          <choice>
            <value>external</value>
            <value>local</value>
            <value>peer</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="format"/>
      </optional>
      <optional>
        <attribute name="linking">
          <choice>
            <value>none</value>
            <value>normal</value>
            <value>sourceonly</value>
            <value>targetonly</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="toc">
          <choice>
            <value>no</value>
            <value>yes</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="print">
          <choice>
            <value>no</value>
            <value>printonly</value>
            <value>yes</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="search">
          <choice>
            <value>no</value>
            <value>yes</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="chunk"/>
      </optional>
      <optional>
        <attribute name="keyscope" dita:since="1.3"/>
      </optional>
    </define>
```

mapGroupDomain.rng

```
<define name="topichead.attributes">
        <optional>
          <attribute name="navtitle"/>
        </optional>
        <optional>
          <attribute name="outputclass"/>
```

```
        </optional>
        <optional>
          <attribute name="keys"/>
        </optional>
        <optional>
          <attribute name="copy-to"/>
        </optional>
        <ref name="topicref-atts"/>
        <ref name=""topicref-atts-no-locktitle">
        <ref name="univ-atts"/>
      </define>
...
<define name="topicgroup.attributes">
        <optional>
          <attribute name="outputclass"/>
        </optional>
        <ref name="topicref-atts"/>
        <ref name=""topicref-atts-no-locktitle">
        <ref name="univ-atts"/>
      </define>
```

Figure 42: Removing @navtitle

map.mod

```
<!ENTITY % topicref.attributes
              "navtitle
                              CDATA
                                              #IMPLIED
               href
                              CDATA
                                              #IMPLIED
               keyref
                              CDATA
                                              #IMPLIED
               keys
                              CDATA
                                              #IMPLIED
               query
                              CDATA
                                              #IMPLIED
               copy-to
                              CDATA
                                              #IMPLIED
               outputclass
                              CDATA
                                              #IMPLIED
               %topicref-atts;
               %univ-atts;"
>
```

Figure 43: Removing @navtitle

mapMod.rng

```
<define name="topicref.attributes">
        <optional>
          <attribute name="navtitle"/>
        </optional>
        <optional>
          <attribute name="href"/>
        </optional>
        <optional>
          <attribute name="keyref"/>
        </optional>
        <optional>
          <attribute name="keys"/>
        </optional>
        <optional>
          <attribute name="query"/>
        </optional>
```

```
        <optional>
          <attribute name="copy-to"/>
        </optional>
        <optional>
          <attribute name="outputclass"/>
        </optional>
        <ref name="topicref-atts"/>
        <ref name="univ-atts"/>
      </define>
```

Figure 44: Removing @print

`map.mod`

Remove the following code:

```
print
                          (no |
                           printonly |
                           yes |
                           -dita-use-conref-target)
                                    #IMPLIED
```

From the following entities:

- `%topicref-atts;`
- `%topicref-atts-no-toc;`
- `%topicref-atts-no-toc-no-keyscope;`
- `%topicref-atts-without-format;`

Figure 45: Removing @print

`mapMod.rng`

Remove the following code:

```
optional>
        <attribute name="print">
          <choice>
            <value>no</value>
            <value>printonly</value>
            <value>yes</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
```

From the following entities:

- `%topicref-atts;`
- `%topicref-atts-no-toc;`
- `%topicref-atts-no-toc-no-keyscope;`
- `%topicref-atts-without-format;`

Figure 46: Removing @query

`map.mod`

```
<!ENTITY % topicref.attributes
              "navtitle
                          CDATA
                                    #IMPLIED
               href
                          CDATA
```

```
                                        #IMPLIED
                keyref
                          CDATA
                                        #IMPLIED
                keys
                          CDATA
                                        #IMPLIED
                query
                          CDATA
                                        #IMPLIED
                copy-to
                          CDATA
                                        #IMPLIED
                outputclass
                          CDATA
                                        #IMPLIED
                %topicref-atts;
                %univ-atts;"
>
```

mapGroup.mod

Remove the following code:

```
query
                          CDATA
                                        #IMPLIED
```

From the following entities:

- %anchorref.attributes;
- %mapref.attributes;
- %topicset.attributes;
- %topicsetref.attributes;
- %keydef.attributes;

topic.mod

```
<!ENTITY % link.attributes
               "href
                          CDATA
                                        #IMPLIED
                keyref
                          CDATA
                                        #IMPLIED
                query
                          CDATA
                                        #IMPLIED
                %relational-atts;
                %univ-atts;
                outputclass
                          CDATA
                                        #IMPLIED"
>
```

Figure 47: Removing @query

mapMod.rng

Remove the following code:

```
<optional>
          <attribute name="query"/>
        </optional>
```

From the following entities:

- %anchorref.attributes;
- %mapref.attributes;
- %topicset.attributes;
- %topicsetref.attributes;
- %keydef.attributes;

```
<define name="topicref.attributes">
        <optional>
          <attribute name="navtitle"/>
        </optional>
        <optional>
          <attribute name="href"/>
        </optional>
        <optional>
          <attribute name="keyref"/>
        </optional>
        <optional>
          <attribute name="keys"/>
        </optional>
        <optional>
          <attribute name="query"/>
        </optional>
        <optional>
          <attribute name="copy-to"/>
        </optional>
        <optional>
          <attribute name="outputclass"/>
        </optional>
        <ref name="topicref-atts"/>
        <ref name="univ-atts"/>
      </define>
```

mapGroupDomain.rng

```
<define name="anchorref.attributes">
        <optional>
          <attribute name="navtitle"/>
        </optional>
        <optional>
          <attribute name="href"/>
        </optional>
        <optional>
          <attribute name="keyref"/>
        </optional>
        <optional>
          <attribute name="keys"/>
        </optional>
        <optional>
          <attribute name="keyscope" dita:since="1.3"/>
        </optional>
        <optional>
          <attribute name="query"/>
        </optional>
        <optional>
          <attribute name="copy-to"/>
        </optional>
        <optional>
          <attribute name="outputclass"/>
        </optional>
        <optional>
          <attribute name="collection-type">
            <choice>
              <value>choice</value>
              <value>family</value>
              <value>sequence</value>
              <value>unordered</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
```

```
        </optional>
        <optional>
          <attribute name="processing-role">
            <choice>
              <value>normal</value>
              <value>resource-only</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="type" a:defaultValue="anchor"/>
        </optional>
        <optional>
          <attribute name="cascade" dita:since="1.3"/>
        </optional>
        <optional>
          <attribute name="scope">
            <choice>
              <value>external</value>
              <value>local</value>
              <value>peer</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="locktitle">
            <choice>
              <value>no</value>
              <value>yes</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="format" a:defaultValue="ditamap"/>
        </optional>
        <optional>
          <attribute name="linking">
            <choice>
              <value>none</value>
              <value>normal</value>
              <value>sourceonly</value>
              <value>targetonly</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="toc">
            <choice>
              <value>no</value>
              <value>yes</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="print">
            <choice>
              <value>no</value>
              <value>printonly</value>
              <value>yes</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="search">
            <choice>
              <value>no</value>
              <value>yes</value>
              <value>-dita-use-conref-target</value>
```

```
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="chunk"/>
        </optional>
        <ref name="univ-atts"/>
      </define>
...
<define name="mapref.attributes">
        <optional>
          <attribute name="navtitle"/>
        </optional>
        <optional>
          <attribute name="href"/>
        </optional>
        <optional>
          <attribute name="keyref"/>
        </optional>
        <optional>
          <attribute name="keys"/>
        </optional>
        <optional>
          <attribute name="query"/>
        </optional>
        <optional>
          <attribute name="copy-to"/>
        </optional>
        <optional>
          <attribute name="outputclass"/>
        </optional>
        <optional>
          <attribute name="format" a:defaultValue="ditamap"/>
        </optional>
        <ref name="topicref-atts-without-format"/>
        <ref name="univ-atts"/>
      </define>
...
<define name="topicset.attributes">
        <optional>
          <attribute name="navtitle"/>
        </optional>
        <optional>
          <attribute name="href"/>
        </optional>
        <optional>
          <attribute name="keyref"/>
        </optional>
        <optional>
          <attribute name="keys"/>
        </optional>
        <optional>
          <attribute name="keyscope" dita:since="1.3"/>
        </optional>
        <optional>
          <attribute name="query"/>
        </optional>
        <optional>
          <attribute name="copy-to"/>
        </optional>
        <optional>
          <attribute name="outputclass"/>
        </optional>
        <optional>
          <attribute name="collection-type">
            <choice>
              <value>choice</value>
              <value>family</value>
              <value>sequence</value>
              <value>unordered</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
```

```
        <optional>
          <attribute name="processing-role">
            <choice>
              <value>normal</value>
              <value>resource-only</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="type"/>
        </optional>
        <optional>
          <attribute name="cascade" dita:since="1.3"/>
        </optional>
        <optional>
          <attribute name="scope">
            <choice>
              <value>external</value>
              <value>local</value>
              <value>peer</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="locktitle">
            <choice>
              <value>no</value>
              <value>yes</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="format"/>
        </optional>
        <optional>
          <attribute name="linking">
            <choice>
              <value>none</value>
              <value>normal</value>
              <value>sourceonly</value>
              <value>targetonly</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="toc">
            <choice>
              <value>no</value>
              <value>yes</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="print">
            <choice>
              <value>no</value>
              <value>printonly</value>
              <value>yes</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="search">
            <choice>
              <value>no</value>
              <value>yes</value>
              <value>-dita-use-conref-target</value>
            </choice>
```

```
          </attribute>
        </optional>
        <optional>
          <attribute name="chunk"/>
        </optional>
        <attribute name="id">
          <data type="NMTOKEN"/>
        </attribute>
        <ref name="conref-atts"/>
        <ref name="select-atts"/>
        <ref name="localization-atts"/>
      </define>
...
<define name="topicsetref.attributes">
        <optional>
          <attribute name="navtitle"/>
        </optional>
        <optional>
          <attribute name="href"/>
        </optional>
        <optional>
          <attribute name="keyref"/>
        </optional>
        <optional>
          <attribute name="keys"/>
        </optional>
        <optional>
          <attribute name="keyscope" dita:since="1.3"/>
        </optional>
        <optional>
          <attribute name="query"/>
        </optional>
        <optional>
          <attribute name="copy-to"/>
        </optional>
        <optional>
          <attribute name="outputclass"/>
        </optional>
        <optional>
          <attribute name="collection-type">
            <choice>
              <value>choice</value>
              <value>family</value>
              <value>sequence</value>
              <value>unordered</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="processing-role">
            <choice>
              <value>normal</value>
              <value>resource-only</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="type" a:defaultValue="topicset"/>
        </optional>
        <optional>
          <attribute name="cascade" dita:since="1.3"/>
        </optional>
        <optional>
          <attribute name="scope">
            <choice>
              <value>external</value>
              <value>local</value>
              <value>peer</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
```

```
        <optional>
          <attribute name="locktitle">
            <choice>
              <value>no</value>
              <value>yes</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="format" a:defaultValue="ditamap"/>
        </optional>
        <optional>
          <attribute name="linking">
            <choice>
              <value>none</value>
              <value>normal</value>
              <value>sourceonly</value>
              <value>targetonly</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="toc">
            <choice>
              <value>no</value>
              <value>yes</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="print">
            <choice>
              <value>no</value>
              <value>printonly</value>
              <value>yes</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="search">
            <choice>
              <value>no</value>
              <value>yes</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="chunk"/>
        </optional>
        <ref name="univ-atts"/>
      </define>
...
<define name="keydef.attributes">
        <optional>
          <attribute name="navtitle"/>
        </optional>
        <optional>
          <attribute name="href"/>
        </optional>
        <optional>
          <attribute name="keyref"/>
        </optional>
        <attribute name="keys"/>
        <optional>
          <attribute name="keyscope" dita:since="1.3"/>
        </optional>
        <optional>
          <attribute name="query"/>
        </optional>
```

```
        <optional>
          <attribute name="copy-to"/>
        </optional>
        <optional>
          <attribute name="outputclass"/>
        </optional>
        <optional>
          <attribute name="collection-type">
            <choice>
              <value>choice</value>
              <value>family</value>
              <value>sequence</value>
              <value>unordered</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="processing-role" a:defaultValue="resource-only">
            <choice>
              <value>normal</value>
              <value>resource-only</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="type"/>
        </optional>
        <optional>
          <attribute name="cascade" dita:since="1.3"/>
        </optional>
        <optional>
          <attribute name="scope">
            <choice>
              <value>external</value>
              <value>local</value>
              <value>peer</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="locktitle">
            <choice>
              <value>no</value>
              <value>yes</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="format"/>
        </optional>
        <optional>
          <attribute name="linking">
            <choice>
              <value>none</value>
              <value>normal</value>
              <value>sourceonly</value>
              <value>targetonly</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="toc">
            <choice>
              <value>no</value>
              <value>yes</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
```

```
        <optional>
          <attribute name="print">
            <choice>
              <value>no</value>
              <value>printonly</value>
              <value>yes</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="search">
            <choice>
              <value>no</value>
              <value>yes</value>
              <value>-dita-use-conref-target</value>
            </choice>
          </attribute>
        </optional>
        <optional>
          <attribute name="chunk"/>
        </optional>
        <ref name="univ-atts"/>
      </define>
```

topicMod.rng

```
<define name="link.attributes">
          <optional>
            <attribute name="href"/>
          </optional>
          <optional>
            <attribute name="keyref"/>
          </optional>
          <optional>
            <attribute name="query"/>
          </optional>
          <ref name="relational-atts"/>
          <ref name="univ-atts"/>
          <optional>
            <attribute name="outputclass"/>
          </optional>
        </define>
```

Figure 48: Removing @refcols

commonElements.mod

```
<!ENTITY % simpletable.attributes
              "relcolwidth
                          CDATA
                                    #IMPLIED
               keycol
                          NMTOKEN
                                    #IMPLIED
               refcols
                          NMTOKENS
                                    #IMPLIED
               %display-atts;
               spectitle
                          CDATA
                                    #IMPLIED
               %univ-atts;
               outputclass
                          CDATA
```

```
                                    #IMPLIED"
>
```

Figure 49: Removing @refcols

commonElementsMod.rng

```
 <define name="simpletable.attributes">
          <optional>
            <attribute name="relcolwidth"/>
          </optional>
          <optional>
            <attribute name="keycol">
              <data type="NMTOKEN"/>
            </attribute>
          </optional>
          <optional>
            <attribute name="refcols">
              <data type="NMTOKENS"/>
            </attribute>
          </optional>
          <ref name="display-atts"/>
          <optional>
            <attribute name="spectitle"/>
          </optional>
          <ref name="univ-atts"/>
          <optional>
            <attribute name="outputclass"/>
          </optional>
        </define>
```

Figure 50: Removing @role="sample" and @role="external"

topic.mod

```
<!ENTITY % relational-atts
               "type
                           CDATA
                                    #IMPLIED
                cascade
                           CDATA
                                    #IMPLIED
                format
                           CDATA
                                    #IMPLIED
                scope
                           (external |
                            local |
                            peer |
                            -dita-use-conref-target)
                                    #IMPLIED
                role
                           (ancestor |
                            child |
                            cousin |
                            descendant |
                            external |
                            friend |
                            next |
                            other |
                            parent |
                            previous |
                            sample |
                            sibling |
                            -dita-use-conref-target)
                                    #IMPLIED
                otherrole
                           CDATA
                                    #IMPLIED"
>
<!ENTITY % rel-atts
```

```
              "type
                          CDATA
                                    #IMPLIED
               role
                          (ancestor |
                           child |
                           cousin |
                           descendant |
                           external |
                           friend |
                           next |
                           other |
                           parent |
                           previous |
                           sample |
                           sibling |
                           -dita-use-conref-target)
                                    #IMPLIED
               otherrole
                          CDATA
                                    #IMPLIED"
>
```

Figure 51: Removing @role="sample" and @role="external"

topicMod.rng

```
<define name="relational-atts">
      <optional>
        <attribute name="type"/>
      </optional>
      <optional>
        <attribute name="cascade" dita:since="1.3"/>
      </optional>
      <optional>
        <attribute name="format"/>
      </optional>
      <optional>
        <attribute name="scope">
          <choice>
            <value>external</value>
            <value>local</value>
            <value>peer</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="role">
          <choice>
            <value>ancestor</value>
            <value>child</value>
            <value>cousin</value>
            <value>descendant</value>
            <value>external</value>
            <value>friend</value>
            <value>next</value>
            <value>other</value>
            <value>parent</value>
            <value>previous</value>
            <value>sample</value>
            <value>sibling</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="otherrole"/>
      </optional>
    </define>
<define name="rel-atts">
      <a:documentation>rel-atts is deprecated as of DITA 1.2, retained for backward
```

```
compatibility.</a:documentation>
      <optional>
        <attribute name="type"/>
      </optional>
      <optional>
        <attribute name="role">
          <choice>
            <value>ancestor</value>
            <value>child</value>
            <value>cousin</value>
            <value>descendant</value>
            <value>external</value>
            <value>friend</value>
            <value>next</value>
            <value>other</value>
            <value>parent</value>
            <value>previous</value>
            <value>sample</value>
            <value>sibling</value>
            <value>-dita-use-conref-target</value>
          </choice>
        </attribute>
      </optional>
      <optional>
        <attribute name="otherrole"/>
      </optional>
    </define>
```

Figure 52: Removing @title on <map>

map.mod

```
<!ENTITY % map.attributes
              "title
                              CDATA
                                          #IMPLIED
               id
                           ID
                                          #IMPLIED
               %conref-atts;
               anchorref
                           CDATA
                                          #IMPLIED
               outputclass
                           CDATA
                                          #IMPLIED
               %localization-atts;
               %topicref-atts;
               %select-atts;"
>
```

Figure 53: Removing @title on <map>

mapMod.rng

```
<define name="map.attributes">
        <optional>
          <attribute name="title"/>
        </optional>
        <optional>
          <attribute name="id">
            <data type="ID"/>
          </attribute>
        </optional>
        <ref name="conref-atts"/>
        <optional>
          <attribute name="anchorref"/>
        </optional>
        <optional>
          <attribute name="outputclass"/>
        </optional>
```

```
        <ref name="localization-atts"/>
        <ref name="topicref-atts"/>
        <ref name="select-atts"/>
      </define>
```

Figure 54: Removing @type="internal" and @type="external" on <lq>

No changes are required to `commonElements.mod`.

Figure 55: Removing @type="internal" and @type="external" on <lq>

No changes are required to `commonElementsMod.rng`.

Modified terminology

None

Modified specification documentation

The following architectural spec topics need to be modified as indicated:

- 2.2.2.4 DITA map attributes
  - In `@collection-type`, remove "Where the `@collection-type` attribute is available on elements that cannot directly contain elements (such as `<reltable>` or `<topicref>`), the behavior of the attribute is reserved for future use."
  - Remove `@navtitle` entry
  - In `@locktitle`, remove note about `@navtitle`
- 2.2.2.5.4 Example: How the @cascade attribute functions: Remove `@navtitle` from example
- 2.2.3.6 Scaling a list of controlled values to define a taxonomy: Remove `@navtitle` from example
- 2.2.3.8.1 Example: How hierarchies defined in a subject scheme map affect filtering: Remove `@navtitle` from example
- 2.2.3.8.2 Example: Extending a subject scheme: Remove `@navtitle` from example
- 2.2.3.8.3 Example: Extending a subject scheme upwards: Remove `@navtitle` from example
- 2.2.4.2.1 Conditional processing attributes: In `@deliveryTarget`, remove "This attribute is a replacement for the now deprecated `@print` attribute."
- 2.2.4.4 Cascading of metadata attributes in a DITA map: Remove mention of `@print`
- 2.2.4.5 Reconciling topic and map metadata elements: Remove `@navtitle` from example
- 2.2.4.6.1 Cascading of attributes from map to map: Remove mention of `@print` from text and example
- 2.3.4.9 Processing key references to generate text or link text: Remove "(such as `@alt` on the `<image>`element)"
- 2.4.1.1 Table of contents: Remove list item about `@print`
- 2.4.3.4 Conditional processing to generate multiple deliverable types
  - Remove "The map or topic references can still use the deprecated `@print` attribute to indicate whether they are applicable to print deliverables."
  - In `@deliveryTarget`,This attribute is a replacement for the now deprecated print attribute. "
  - Remove entry for `@print`
- 2.4.5.2 Chunking examples: Remove value of "to-navigation"

- 2.6.3.3 DTD: Coding requirements for element type declarations: Modify declaration of topichead.attributes to include new attribute group entity defined to exclude collection-type
- 2.6.4.3 RELAX NG: Coding requirements for element type declarations: Modify declaration of topichead.attributes to include new attribute group entity defined to exclude collection-type

The following element-reference topics need to be removed:

- 3.7.1 `<boolean>`
- 3.4.2.2 `<indextermref>`

The following element-reference topics need to be modified as indicated:

- 3.1 Base DITA elements, A to Z: Remove links to `<boolean>` and `<indextermref>`
- 3.2.1.5 `<navtitle>`: Remove mentions of `@navtitle`
- 3.2.2.1 `<alt>`: Remove mention of `@alt` from the short description
- 3.2.2.17 `<image>`: Remove `@alt` and `@longdescref`
- 3.2.2.23 `<lq>`: Remove "internal" and "external" as specified values for the `@type` attribute
- 3.2.2.25 `<object>`: Remove `@longdescref`
- 3.2.4.1 `<link>`: Remove `@query`
- 3.2.4.2 `<linklist>`: Remove paragraph about "tree" as a value for `@collection-type`
- 3.2.4.3 `<linkpool>`: Remove paragraph about "tree" as a value for `@collection-type`
- 3.3.1.5 `<navref>`: Remove `@keyref`
- 3.3.1.1 `<map>`: Remove `@title`
- 3.3.1.6 `<reltable>`: Modify attributes section to indicate that `<reltable>` does not have `@collection-type`
- 3.3.1.10 `<relcolspec>`: Modify attributes section to indicate that `<relcolspec>` does not have `@collection-type`
- 3.3.2.4 `<topicgroup>`: Modify attributes section to indicate that `<topicgroup>` does not have `@locktitle`
- 3.3.2.5 `<topichead>`: Modify attributes seciton to indicate that `<topichead>` does not have `@locktitle`
- 3.4.1.22`<resourceid>`: Remove paragraph about deprecated usage of `@id`
- 3.5.1.3 `<hazardsymbol>`: Remove `@longdescref`
- 3.6.1.2 `<schemeref>`: Remove mention of `@navtitle`
- 3.6.1.15 `<relatedSubjects>`: Remove mention of `@navtitle` from text AND the example
- 3.6.2.2 `<topicapply>`: Remove mention of `@navtitle`
- 3.10.1.2 Metadata attribute group: In description of `@deliveryTarget`, remove mention of `@print`
- 3.10.3 Attributes common to many map elements:
  - Remove the following paragraph: "Usage of the `@collection-type` attribute on `<relatable>` and `<relcolspec>` is currently undefined and reserved for future use."
  - Remove `@print`
- 3.10.10 Simpletable attribute group: Remove `@refcols`
- 3.10.12 Topicref element attributes group: Remove `@navtitle` and `@query`
- 3.10.13.5.1 Using the -dita-use-conref-target value: Remove note about `@navtitle` and usage of `@navtitle` in the examples
- 3.10.13.12 The `@role` and `@otherrole` attributes: Remove values of "sample" and "external"
- B.6 Element-by-element recommendations for translators

- In "Topic elements," remove rows for `<boolean>` and `<indextermref>`
- In "Topic elements," remove `@alt` on `<image>`
- In "Map elements," remove paragraph about `@navtitle`
- In "Map elements," remove `@title` on `<map>` and footnote 8
- In "Subject scheme elements," remove mentions of `@navtitle`
- Remove footnote 5 about `@alt` on `<image>`

The following element-reference topics contain examples that must be modified to remove instances of `@navtitle`. The `@navtitle` attribute can be removed (and the file name made more descriptive) or replaced by either an XML comment or a `<navtitle>` element.

- 3.3.1.4 `<anchor>`
- 3.3.2.3`<mapref>`
- 3.4.1.14 `<metadata>`
- 3.6.1.1 `<subjectScheme>`
- 3.6.1.3 `<hasInstance>`
- 3.6.1.4 `<hasKind>`
- 3.6.1.5 `<hasNarrower>`
- 3.6.1.6 `<hasPart>`
- 3.6.1.7 `<hasRelated>`
- 3.6.1.8 `<enumerationdef>`
- 3.6.1.10 `<attributedef>`
- 3.6.1.14 `<subjectdef>`
- 3.6.2.1 `<subjectref>`
- 3.6.2.3 `<topicsubject>`
- 3.6.2.4 `<topicSubjectTable>`

The following table contains precise suggestions for the changes to be made.

Topic	DITA 1.3 content	DITA 2.0 content

3.3.1.4 `<anchor>`	`<map>`
	`  <title>MyComponent tasks</title>`
	`  <topicref navtitle="Start here" href="start.dita" toc="yes">`
	`  <navref mapref="othermap2.ditamap"/>`
	`  <navref mapref="othermap3.ditamap"/>`
	`  <anchor id="a1"/>`
	`  </topicref>`
	`</map>`
		`<map>`
		`  <title>MyComponent tasks</title>`
		`  <topicref href="start.dita" toc="yes">`
		`  <navref mapref="othermap2.ditamap"/>`
		`  <navref mapref="othermap3.ditamap"/>`
		`  <anchor id="a1"/>`
		`  </topicref>`
		`</map>`

3.3.2.3`<mapref>`	**Figure 56: Map that reuses lib.ditamap**
	`<map id="standardlib">`
	`  <topichead navtitle="Developing with standard libraries">`
	`  <mapref href="lib.ditamap"/>`
		Figure 57: Map that reuses lib.ditamap
		`<map id="standardlib">`
		`  <topichead>`
		`  <topicmeta>`
		`  <navtitle>Developing with standard libraries</navtitle>`

Topic	DITA 1.3 content	DITA 2.0 content
	<pre> </topichead> <!-- ... --> </map></pre>	<pre> </topicmeta> <mapref href="lib.ditamap"/> </topichead> <!-- ... --> </map></pre>
3.4.1.14 <metadata>	Metadata within a map:<pre> <topicref href="metadata.dita" navtitle="metadata element"> <topicmeta> **<metadata>** <keywords> <indexterm>metadata element</indexterm> </keywords> **</metadata>** </topicmeta> </topicref></pre>	Metadata within a map:<pre> <topicref href="metadata.dita"> <topicmeta> **<metadata>** <keywords> <indexterm>metadata element</indexterm> </keywords> **</metadata>** </topicmeta> </topicref></pre>
3.6.1.1 <subjectScheme>	Same as <enumerationdef>	Same as <enumerationdef>
3.6.1.3 <hasInstance>	<pre> <subjectScheme> <hasInstance> <subjectdef keys="city" navtitle="City"> <subjectdef keys="nyc" navtitle="New York City"/> <subjectdef keys="reykjavik" navtitle="Reykjavik"/> <subjectdef keys="moscow" navtitle="Moscow"/> </subjectdef> </hasInstance> </subjectScheme></pre>	<pre> <subjectScheme> <hasInstance> <subjectdef keys="city"> <subjectdef keys="nyc"/> <subjectdef keys="reykjavik"/> <subjectdef keys="moscow"/> </subjectdef> </hasInstance> </subjectScheme></pre>
3.6.1.4 <hasKind>	This example specifies that cities, towns, and villages are each a kind of settlement. Additionally, bigcity, mediumcity, and smallcity are each a kind of city.<pre> <subjectScheme> <hasKind> <subjectdef keys="settlement" navtitle="Human settlement"> <subjectdef keys="city" navtitle="City"> <subjectdef keys="bigcity" navtitle="Big city"/> <subjectdef keys="mediumcity" navtitle="Medium city"/> <subjectdef keys="smallcity" navtitle="Small city"/> </subjectdef></pre>	This example specifies that cities, towns, and villages are each a kind of settlement. Additionally, big-city, medium-city, and small-city are each a kind of city.<pre> <subjectScheme> <hasKind> <subjectdef keys="settlement"> <subjectdef keys="city"> <subjectdef keys="big-city"/> <subjectdef keys="medium-city"/> <subjectdef keys="small-city"/> </subjectdef> <subjectdef keys="town"/> <subjectdef keys="village"</pre>

Topic	DITA 1.3 content	DITA 2.0 content
	<subjectdef keys="town" navtitle="Town"/> <subjectdef keys="village" navtitle="Village"/> </subjectdef> </hasKind> </subjectScheme>	navtitle="Village"/> </subjectdef> </hasKind> </subjectScheme>
3.6.1.5 <hasNarrower>	<subjectScheme> <hasNarrower> <subjectdef keys="horticulture" navtitle="Horticulture"> <subjectdef keys="plantrose" navtitle="Planting Roses"/> </subjectdef> </hasNarrower> </subjectScheme>	<subjectScheme> <hasNarrower> <subjectdef keys="horticulture"> <subjectdef keys="planting-roses"/> </subjectdef> </hasNarrower> </subjectScheme>
3.6.1.6 <hasPart>	<subjectScheme> <hasPart> <subjectdef keys="car" navtitle="Car"> <subjectdef keys="tire" navtitle="Tire"/> <subjectdef keys="horn" navtitle="Horn"/> </subjectdef> </hasPart> </subjectScheme>	<subjectScheme> <hasPart> <subjectdef keys="car"> <subjectdef keys="tire"/> <subjectdef keys="horn"/> </subjectdef> </hasPart> </subjectScheme>
3.6.1.7 <hasRelated>	<subjectScheme> <subjectdef keys="myProgram" navtitle="My Program"> <hasRelated keys="runsOn" navtitle="runs on"> <subjectdef keys="linux" navtitle="Linux"/> <subjectdef keys="mswin" navtitle="Microsoft Windows"/> </hasRelated> </subjectdef> </subjectScheme>	<subjectScheme> <subjectdef keys="myProgram"> <hasRelated keys="platforms"> <subjectdef keys="linux"> <subjectdef keys="windows"/> </hasRelated> </subjectdef> </subjectScheme>
3.6.1.8 <enumerationdef>	<subjectScheme> <!-- Pull in a scheme that defines unix OS values --> <schemeref href="unixOS.ditamap"/> <!-- Define new OS values that are merged with those in the unixOS scheme --> <subjectdef keys="os">	<subjectScheme> <!-- Pull in a scheme that defines unix OS values --> <schemeref href="unixOS.ditamap"/> <!-- Define new OS values that are merged with those in the unixOS scheme --> <subjectdef

Topic	DITA 1.3 content	DITA 2.0 content
	<subjectdef keys="linux"/> <subjectdef keys="mswin"/> <subjectdef keys="zos"/> </subjectdef> <!-- Define application values --> <subjectdef keys="app" navtitle="Applications"> <subjectdef keys="apacheserv" href="subject/apache.dita"/> <subjectdef keys="mysql" href="subject/sql.dita"/> </subjectdef> <!-- Define an enumeration of the platform attribute, equal to each value in the OS subject. This makes the following values valid for the platform attribute: linux, mswin, zos --> <enumerationdef> <attributedef name="platform"/> <subjectdef keyref="os"/> </enumerationdef> <!-- Define an enumeration of the otherprops attribute, equal to each value in the application subjects. This makes the following values valid for the otherprops attribute: apacheserv, mysql --> <enumerationdef> <attributedef name="otherprops"/> <subjectdef keyref="app"/> </enumerationdef> </subjectScheme>	keys="operating-systems"> <subjectdef keys="linux"/> <subjectdef keys="windows"/> <subjectdef keys="zOS"/> </subjectdef> <!-- Define application values --> <subjectdef keys="applications"> <subjectdef keys="apache-server" href="subject/apache.dita"/> <subjectdef keys="my-sql" href="subject/sql.dita"/> </subjectdef> <!-- Define an enumeration of the platform attribute, equal to each value in the OS subject. This makes the following values valid for the platform attribute: linux, windows, zOS --> <enumerationdef> <attributedef name="platform"/> <subjectdef keyref="os"/> </enumerationdef> <!-- Define an enumeration of the otherprops attribute, equal to each value in the application subjects. This makes the following values valid for the otherprops attribute: apache-server, my-sql --> <enumerationdef> <attributedef name="otherprops"/> <subjectdef keyref="applications"/> </enumerationdef> </subjectScheme>
3.6.1.10 <attributedef>	Same as <enumerationdef>	Same as <enumerationdef>
3.6.1.14 <subjectdef>	Same as <enumerationdef>	Same as <enumerationdef>
3.6.2.1 <subjectref>	In the following example, the map is classified as covering the Linux subject, and the "Developing web applications" topic is classified as covering the web and development subjects. These subjects (and their keys) are defined externally in a subject scheme map; in order to reference the subject directly without	In the following example, the map is classified as covering the Linux subject, and developing-web-applications.dita is classified as covering the web and development subjects. These subjects (and their keys) are defined externally in a subject scheme map; in order to reference the subject

Topic	DITA 1.3 content	DITA 2.0 content

	the subject scheme map, the `@href` attribute would be used in place of `@keyref`.
	
	`<map>`
	`  <title>Working with Linux</title>`
	`  <topicsubject keyref="linux"/>`
	`  <!-- ... -->`
	`  <topicref href="webapp.dita" navtitle="Developing web applications">`
	`  <topicsubject>`
	`  **<subjectref keyref="web"/>**`
	`  **<subjectref keyref="development"/>**`
	`  </topicsubject>`
	`  <!-- ... -->`
	`  </topicref>`
	`  <!-- ... -->`
	`</map>`
		directly without the subject scheme map, the `@href` attribute would be used in place of `@keyref`.
		
		`<map>`
		`  <title>Working with Linux</title>`
		`  <topicsubject keyref="linux"/>`
		`  <!-- ... -->`
		`  <topicref href="developing-web-applications.dita">`
		`  <topicsubject>`
		`  **<subjectref keyref="web"/>**`
		`  **<subjectref keyref="development"/>**`
		`  </topicsubject>`
		`  <!-- ... -->`
		`  </topicref>`
		`  <!-- ... -->`
		`</map>`

3.6.2.3 `<topicsubject>`	Same as `<subjectref>`; should be reused	Same as `<subjectref>`; should be reused
3.6.2.4 `<topicSubjectTable>`	The following `<topicSubjectTable>` classifies several topics according to subjects defined in the previous map. As with any `<topicSubjectTable>`, the first column is used to specify topics. In this specific example, the second column is used to specify a goal, based on the "goal" subject in the header. The third column is used to specify an operating system. Based on those definitions, the following classifications are made by this table: • The topics "Configuring cron for efficient startup" and "Allocating raw storage" are each classified by the goal of "performance"; they also are classified by the operating systems "linux" and "unix". • The topics "Analyzing web logs for service issues" and "Detecting denial-of-service attacks" are each classified by the goal of "reliability"; they also are classified by the operating systems	The following `<topicSubjectTable>` classifies several topics according to subjects defined in the previous map. As with any `<topicSubjectTable>`, the first column is used to specify topics. In this specific example, the second column is used to specify a goal, based on the "goal" subject in the header. The third column is used to specify an operating system. Based on those definitions, the following classifications are made by this table: • The topics `configure-cron-for-efficiency.dita` and `allocating-raw-storage.dita` are each classified by the goal of "performance"; they also are classified by the operating systems "linux" and "unix". • The topics `analyze-web-logs.dita` and `detect-denial-of-service-attacks.dita` are each classified by the goal of "reliability"; they also are

Topic	DITA 1.3 content	DITA 2.0 content

	"linux", "unix", and "windows".
	• No relationship is defined between subjects in the table, meaning that this table does not define any relationship between the goal of "performance" and the operating systems "linux" or "unix".
	
	`<map>`
	`<!-- ... -->`
	`<topicSubjectTable>`
	`<topicSubjectHeader>`
	`<topicCell type="task"/>`
	`<subjectCell>`
	`<topicsubject keyref="goal"/>`
	`</subjectCell>`
	`<subjectCell>`
	`<topicapply keyref="os"/>`
	`</subjectCell>`
	`</topicSubjectHeader>`
	`<topicSubjectRow>`
	`<topicCell>`
	`<topicref href="webServerStart.dita" navtitle="Configuring cron for efficient startup"/>`
	`<topicref href="dbDisk.dita" navtitle="Allocating raw storage"/>`
	`</topicCell>`
	`<subjectCell>`
	`<topicsubject keyref="performance"/>`
	`</subjectCell>`
	`<subjectCell>`
	`<topicapply keyref="linux"/>`
	`<topicapply keyref="unix"/>`
	`</subjectCell>`
	`</topicSubjectRow>`
	`<topicSubjectRow>`
	`<topicCell>`
	`<topicref href="webLogAnalyze.dita" navtitle="Analyzing web logs for service issues"/>`
	`<topicref href="webDenialService.dita" navtitle="Detecting denial-of-service attacks"/>`
	`</topicCell>`
	`<subjectCell>`
	`<topicsubject keyref="reliability"/>`
	`</subjectCell>`
	`<subjectCell>`
	`<topicapply keyref="linux"/>`
	`<topicapply keyref="unix"/>`
		classified by the operating systems "linux", "unix", and "windows".
		• No relationship is defined between subjects in the table, meaning that this table does not define any relationship between the goal of "performance" and the operating systems "linux" or "unix".
		
		`<map>`
		`<!-- ... -->`
		`<topicSubjectTable>`
		`<topicSubjectHeader>`
		`<topicCell type="task"/>`
		`<subjectCell>`
		`<topicsubject keyref="goal"/>`
		`</subjectCell>`
		`<subjectCell>`
		`<topicapply keyref="os"/>`
		`</subjectCell>`
		`</topicSubjectHeader>`
		`<topicSubjectRow>`
		`<topicCell>`
		`<topicref href="configure-cron-for-efficiency.dita"/>`
		`<topicref href="allocating-raw-storage.dita"/>`
		`</topicCell>`
		`<subjectCell>`
		`<topicsubject keyref="performance"/>`
		`</subjectCell>`
		`<subjectCell>`
		`<topicapply keyref="linux"/>`
		`<topicapply keyref="unix"/>`
		`</subjectCell>`
		`</topicSubjectRow>`
		`<topicSubjectRow>`
		`<topicCell>`
		`<topicref href="analyze-web-logs.dita"/>`
		`<topicref href="detect-denial-of-service-attacks.dita"/>`
		`</topicCell>`
		`<subjectCell>`
		`<topicsubject keyref="reliability"/>`
		`</subjectCell>`
		`<subjectCell>`
		`<topicapply keyref="linux"/>`
		`<topicapply keyref="unix"/>`
		`<topicapply keyref="windows"/>`
		`</subjectCell>`

Topic	DITA 1.3 content	DITA 2.0 content
	<topicapply keyref="windows"/> </subjectCell> </topicSubjectRow> <!-- ... --> </topicSubjectTable> </map>	</topicSubjectRow> <!-- ... --> </topicSubjectTable> </map>

Migration plans for backwards incompatibilities

Information architects and content strategists will need to search their content for elements, attributes, and attribute values that were removed from DITA 2.0:

- **<boolean>**
- <indextermref>
- **@alt**
- @chunk="to-navigation
- @collection-type="tree" on <linkpool> and <linklist>
- @collection-type on <reltable> and <relcolspec>
- @keyref on <navref>
- @locktitle on <topichead> and <topicgroup>
- <image>@longdescref
- **@navtitle**
- **@print**
- @query
- @refcols
- @role="sample" and @role="external"
- **@title**<map>
- @type="internal" and @type="external" on <lq>

The deprecated elements, attributes, and attribute values will need to be removed.

The DITA TC should offer migration advice only for deprecated elements and attributes (highlighted in the list above); we cannot guess at how users have used attribute and attribute values that were never formally defined. In addition, we cannot provide migration advice to vendors who have directly modified the OASIS-provided grammar files.

The following table outlines the basic information for **migrating existing content.**

Deprecated item	Strategy	DITA 1.3 markup	DITA 2.0 markup
@alt	1. Remove the @alt attribute. 2. Create an <alt> element as child of <image>. 3. Insert the value of the @alt into the <alt> element.	<image href="bike.gif" alt="Two-wheeled bicycle"/>	<image href="bike.gif" <alt>Two-wheeled bicycle</alt> </image>

Deprecated item	Strategy	DITA 1.3 markup	DITA 2.0 markup
`<boolean>`	Replace the `<boolean>` with a `<state>` or `<data>` element.	`She said "<boolean state="yes"/>" when I asked her to marry me!` **Note** This is the example from the specification. It is very stupid and embarrassing. The element only had value as a specialization base.	`"She said <state name="answer" value="yes"/> when I asked her to marry me!"` **Note** Again, this is also a very stupid example.
`@print`	1. Identify maps that use the `@print` attribute. 2. Develop a subjectScheme map with appropriate values for the `@deliveryTarget` attribute. 3. Replace the `@print`attribute with a `@deliveryTarget` attribute and a appropriate value.	`<topicref href="foo.dita" print="no">`	`<topicref href="foo.dita" deliveryTarget="Web-only">`
`@navtitle`	1. Identify maps that use the `@navtitle` attribute. 2. Remove the `@navtitle` attribute and replace it with an XML comment or a `@navtitle` element, whichever is appropriate **Note** The DIT suggest simply removin `@navti` if the `@locktitle`	`<subjectScheme> <subjectdef keys="os" navtitle="Operating system"> <subjectdef keys="linux" navtitle="Linux"> <subjectdef keys="redhat" navtitle="RedHat Linux"/> <subjectdef keys="suse" navtitle="SuSE Linux"/> </subjectdef> <subjectdef keys="windows" navtitle="Windows"/> <subjectdef keys="zos" navtitle="z/OS"/> </subjectdef> <enumerationdef>`	`<subjectScheme> <subjectdef keys="os"> <topicmeta> <navtitle>Operating systems</navtitle> </topicmeta> <subjectdef keys="linux"> <topicmeta> <navtitle>Linux</navtitle> </topicmeta> <subjectdef keys="redhat"> <topicmeta> <navtitle>RedHat Linux</navtitle>`

Deprecated item	Strategy	DITA 1.3 markup	DITA 2.0 markup
	attribute not set. `@lockt` attribute set to "y add a `<navtitle>` element.	`<attributedef name="platform"/> <subjectdef keyref="os"/> </enumerationdef> </subjectScheme>`	`</ topicmeta> </ subjectdef> <subjectdef keys="suse"> <topicmeta> <navtitle>SUSE Linux</navtitle> </ topicmeta> </ subjectdef> </ subjectdef> <subjectdef keys="windows"> <topicmeta> <navtitle>Windows</ navtitle> </ topicmeta> </ subjectdef> <subjectdef keys="zos"> <topicmeta> <navtitle>z/OS</ navtitle> </ topicmeta> </ subjectdef> </subjectdef> <enumerationdef> <attributedef name="platform"/> <subjectdef keyref="os"/> </ enumerationdef> </subjectScheme>`
`@longdescref` on `< image>`	1. Remove the `@longdescref` attribute. 2. Insert a `<longdescref>` element.	`<image href="puffin.jpg" longdescref="http:// www.example.org/ birds/puffin.html"> <alt>Puffin picture</alt> </image>`	`<image href="puffin.jpg"> <alt>Puffin pigure</alt> <longdescref href="http:// www.example.org/ birds/puffin.html" scope="external" format="html"/> </image>`
`@title` on `<map>`	1. Remove the `@title` attribute.	`<map id="mybats" title="Bats>`	`<map id="mybats"> <title>Bats</`

Deprecated item	Strategy	DITA 1.3 markup	DITA 2.0 markup
	2. Insert a `<title>` element.	`<topicref href="bats.dita" type="topic"> <topicref href="batcaring.dita" type="task"/> <topicref href="batfeeding.dita" type="task"/> <topicref href="batsonar.dita" type="concept"/> <topicref href="batguano.dita" type="reference"/> <topicref href="bathistory.dita" type="reference"/> </topicref> </map>`	`title> <topicref href="bats.dita" type="topic"> <topicref href="batcaring.dita" type="task"/> <topicref href="batfeeding.dita" type="task"/> <topicref href="batsonar.dita" type="concept"/> <topicref href="batguano.dita" type="reference"/> <topicref href="bathistory.dita" type="reference"/> </topicref> </map>`

DITA practitioners who have developed specialization and constraint modules that enumerate any of the deprecated elements and attributes will need to remove them from the modules.

3.1.6 Stage 3: #46 Remove @xtrf and @xtrc

The `@xtrf` and `@xtrc` attributes were added in DITA 1.0 purely for processing purposes, to add a standard way for processors to store mid-conversion debug information while (in theory) retaining validity against an original DTD. This is a legacy concern outside the scope of the standard / outside of the usual interoperability concern; they should be removed as part of the effort to clean up DITA 2.0 and remove obsolete or unnecessary markup.

Champion

Robert Anderson

Tracking information

Event	Date	Links
Stage 1 proposal accepted	11 July 2017	Minutes
Stage 2 proposal submitted	2 Feb 2018	HTML, DITA
Stage 2 proposal discussed	6 Feb 2018	Minutes
Stage 2 proposal approved	13 Feb 2018	https://lists.oasis-open.org/archives/dita/201802/msg00050.html
Stage 3 proposal submitted to reviewers	1 March 2018	Carsten Brennecke, Kris Eberlein
Stage 3 proposal (this document) submitted to TC	6 March 2018	

Approved technical requirements

1. Remove declarations of `@xtrf` and `@xtrc` from the grammar files.

2. Remove use of the `%global-atts;` attribute group, which is currently used solely to define those two attributes.

Dependencies or interrelated proposals

N/A

Modified grammar files

For RNG, the following group and declarations will be removed from `commonElementsMod.rng`:

```
<define name="global-atts">
      <optional>
        <attribute name="xtrc"/>
      </optional>
      <optional>
        <attribute name="xtrf"/>
      </optional>
    </define>
```

In addition, all uses of the following should be removed; this is currently used on **every** RNG element declaration, so there is no benefit to listing every instance in the grammar:

```
<ref name="global-atts"/>
```

For DTDs, the following entity and declarations will be deleted from `commonElements.mod`:

```
<!ENTITY % global-atts
               "xtrc
                           CDATA
                                     #IMPLIED
                xtrf
                           CDATA
                                     #IMPLIED"
>
```

In addition, all uses of the following should be removed; this is currently used on **every** DTD element declaration, so there is no benefit to listing every instance in the grammar:

```
%global-atts;
```

Modified terminology

N/A

Modified specification documentation

The Debug Attributes topic will be removed from the specification. None of the content will be preserved and no new content is required elsewhere. It is currently referenced from `ditaref-attributes.ditamap`

Links to this topic will need to be removed from any attribute definitions that currently include them (a small number of elements that modify or do not include most of "universal attributes" group):

- 3.2.1.11 `<dita>` (new text in italics): "The following attributes are available on this element: `@xmlns:ditaarch` and `@DITAArchVersion` from Architectural attribute group, *and the* Localization attribute group ~~and Debug attribute group~~.

- 3.3.1.11 `<ux-window>`: "The following attributes are available on this element: ID attribute group, Metadata attribute group, ~~Debug attribute group,~~ class (Not for use by authors), and the attributes defined below."
- 3.6.1.8 `<enumerationdef>`: The following attributes are available on this element: ID attribute group, `@status` and `@base` from Metadata attribute group, outputclass, ~~Debug attribute group,~~ class (Not for use by authors), and the attributes defined below.
- 3.6.1.9 `<elementdef>`: The following attributes are available on this element: ID attribute group, `@status` and `@base` from Metadata attribute group, outputclass, ~~Debug attribute group,~~ class (Not for use by authors), and the attributes defined below.
- 3.6.1.10 `<attributedef>`: "The following attributes are available on this element: ID attribute group, `@status` and `@base` from Metadata attribute group, outputclass, ~~Debug attribute group,~~ class (Not for use by authors), and the attributes defined below."
- 3.7.7 `<no-topic-nesting>`: "The following attributes are available on this element: ~~Debug attribute group and~~ class (Not for use by authors), and the attributes defined below."

Migration plans for backwards incompatibilities

- Any documents that specify `@xtrf` or `@xtrc` in the source will need to remove the attributes.

  Note There is no migration plan for moving information to other attributes; these attributes were never intended for use in source, so any existing use is non-standard and we cannot predict the proper migration path. One option for this scenario is to create specialization modules that define `@xtrf` and `@xtrc` as specializations of `@base`, which will restore them to all elements. Alternatively, a better solution might be to create new specialized attributes with names that correspond to the content, and to migrate `@xtrf` and `@xtrc` to those new attributes.
- Specialization or constraint modules that include references to the `global-atts` group or parameter entity will need to remove those references. This will most easily be accomplished with a search/replace operation that replaces the reference with an empty value.
- Any specialization or constraint modules that explicitly add these attributes to an attribute list will need to remove the attribute declarations, but this is less likely as the most common design pattern is to reference the group as we do in every OASIS element declaration.

3.1.7 Stage 3: #73 Remove delayed conref domain

Given the complexity of this feature, lack of implementation support, and lack of evidence that it is actually used by DITA adopters, the TC proposes to remove this feature from DITA 2.0. This proposal details the changes to the DITA 2.0 doctypes and specification.

Champion

Alan Houser

Tracking information

Event	Date	Links
Stage 1 proposal accepted	Sept 5 2017	https://www.oasis-open.org/committees/document.php?document_id=61652&wg_abbrev=dita

Event	Date	Links
Stage 2 proposal submitted	6 Feb 2018	https://www.oasis-open.org/apps/org/workgroup/dita/download.php/62481/latest/delayed_conref_revised.html
Stage 2 proposal discussed	13 Feb 2018	https://www.oasis-open.org/committees/document.php?document_id=62511&wg_abbrev=dita
Stage 2 proposal approved	20 Feb 2018	https://www.oasis-open.org/committees/document.php?document_id=62588&wg_abbrev=dita
Stage 3 proposal submitted to reviewers	10 June 2018	Robert Anderson, Kris Eberlein
Stage 3 proposal (this document) submitted to TC	18 June 2018	

Approved technical requirements

Not applicable; does not support any new technical requirements. Proposal meets DITA 2.0 goals of reducing complexity of the DITA 2.0 specification, and reducing complexity for implementors.

Dependencies or interrelated proposals

None.

Modified grammar files

Remove grammar files (RNG): `base/rng/delayResolutionDomain.rng`

Modified grammar files:

base/rng/basemap.rng

Remove `<include href="delayResolutionDomain.rng"/>`

base/rng/catalog.xml

Remove:

```
<system systemId="urn:oasis:names:tc:dita:rng:delayResolutionDomain.rng:1.3"
        uri="rng/delayResolutionDomain.rng"/>
<system systemId="urn:oasis:names:tc:dita:rng:delayResolutionDomain.rng"
        uri="rng/delayResolutionDomain.rng"/>
```

```
<uri name="urn:oasis:names:tc:dita:rng:delayResolutionDomain.rng:1.3"
     uri="rng/delayResolutionDomain.rng"/>
<uri name="urn:oasis:names:tc:dita:rng:delayResolutionDomain.rng"
     uri="rng/delayResolutionDomain.rng"/>
```

Modified terminology

No new terminology.

Modified specification documentation

Removed topics:

- containers/delayedconref-d.dita
- base/exportanchors.dita
- base/anchorid.dita
- base/anchorkey.dita

Modified topics:

langRef/base-elements.ditamap

Delete lines:

```
  <topicref href="containers/delayconref-d.dita">
   <topicref href="base/exportanchors.dita" keys="exportanchors"
navtitle="exportanchors"/>
   <topicref href="base/anchorid.dita" keys="anchorid" navtitle="anchorid"/>
   <topicref href="base/anchorkey.dita" keys="anchorkey" navtitle="anchorkey"/>
  </topicref>
 </topicref>
```

langRef/quick-reference/base-elements-a-to-z.dita

Delete lines:

```
<sli><xref keyref="anchorid"></xref></sli>
<sli><xref keyref="anchorkey"></xref></sli>
<sli><xref keyref="exportanchors"></xref></sli>
```

langRef/quick-reference/all-elements-a-to-z.dita

Delete lines:

```
<sli><xref keyref="anchorid"></xref></sli>
<sli><xref keyref="anchorkey"></xref></sli>
<sli><xref keyref="exportanchors"></xref></sli>
```

common/commonNavLibraryTable.dita

Delete lines:

```
<reference id="contentmodel-anchorid" product="base"><title/><refbody><section
id="contains" otherprops="contains containedby"><title>Content models</title><p>See <xref
keyref="cmoda/anchorid" type="reference">appendix<desc/></xref> for information about
this element in OASIS document type shells.</p></section></refbody></reference>
<reference id="contentmodel-anchorkey" product="base"><title/><refbody><section
id="contains" otherprops="contains containedby"><title>Content models</title><p>See <xref
keyref="cmoda/anchorkey" type="reference">appendix<desc/></xref> for information about
this element in OASIS document type shells.</p></section></refbody></reference>
<reference id="contentmodel-exportanchors" product="base"><title/><refbody><section
id="contains" otherprops="contains containedby"><title>Content models</title><p>See <xref
keyref="cmode/exportanchors" type="reference">appendix<desc/></xref> for information
about this element in OASIS document type shells.</p></section></refbody></reference>
```

non-normative/basedoctypes.dita

Remove two instances of: `<li>Conref delayed resolution (map/topic)</li>`

non-normative/basedomains.dita

Remove:

```
 <strow>
  <stentry>Conref delayed resolution (map/topic)</stentry>
  <stentry>
   <ul>
    <li product="base">Base map</li>
    <li product="technicalContent">Technical content maps: Map, Bookmap, Classification
map</li>
    <li product="learningTraining">Learning and training document types: all maps</li>
   </ul>
  </stentry>
  <stentry>
```

```
   <ul>
    <li product="base">Base topic</li>
    <li product="technicalContent">Technical content maps: Subject scheme</li>
    <li product="technicalContent">Technical content topics: all topics</li>
    <li product="learningTraining">Learning and training document types: all topics</li>
   </ul>
  </stentry>
 </strow>
```

non-normative/oasisdomains.dita

Remove:

```
     <row>
      <entry>Conref delayed resolution (map/topic)</entry>
      <entry>For delaying resolution of <xmlatt>conref</xmlatt> or <xmlatt>keyref</
xmlatt> on some
       elements.</entry>
      <entry>delay-d</entry>
     </row>
```

(Also remove commented reference to domain).

non-normative/elementsMerged.dita

Remove:

```
    <section id="section_delayresolution" product="base">
      <title>Delayed Conref Resolution domain elements <i>(new in DITA 1.2)</i></title>
      <simpletable id="simpletable_delayresolution">
        <sthead>
          <stentry>Element name</stentry>
          <stentry>Specialized from</stentry>
          <stentry>Inherits everything from ancestor?</stentry>
          <stentry>Block/Inline (presentation)</stentry>
          <stentry>Block/Inline (translation)</stentry>
          <stentry>Translatable content?</stentry>
          <stentry>Translatable attributes?</stentry>
        </sthead>
        <strow>
          <stentry><xmlelement>anchorid</xmlelement></stentry>
          <stentry><xmlelement>keyword</xmlelement></stentry>
          <stentry><b>no</b></stentry>
          <stentry>N/A (metadata)</stentry>
          <stentry><b>n/a</b></stentry>
          <stentry><b>no</b></stentry>
          <stentry/>
        </strow>
        <strow>
          <stentry><xmlelement>anchorkey</xmlelement></stentry>
          <stentry><xmlelement>keyword</xmlelement></stentry>
          <stentry><b>no</b></stentry>
          <stentry>N/A (metadata)</stentry>
          <stentry><b>n/a</b></stentry>
          <stentry><b>no</b></stentry>
          <stentry/>
        </strow>
        <strow>
          <stentry><xmlelement>exportanchors</xmlelement></stentry>
          <stentry><xmlelement>keywords</xmlelement></stentry>
          <stentry><b>no</b></stentry>
          <stentry>N/A (metadata)</stentry>
          <stentry><b>n/a</b></stentry>
          <stentry><b>no</b></stentry>
          <stentry/>
        </strow>
      </simpletable>
    </section>
```

</section>

Migration plans for backwards incompatibilities

The DITA TC will not explicitly support migration or provide a mechanism to retain backwards compatibility. DITA 2.0 adopters who use this feature can integrate the DITA 1.3 delayed conref domain; or support this feature in some other way.

3.1.8 Stage 3: #105 Redesign chunking

Simplify how the `@chunk` attribute is defined to 1) make it easier for authors to use, and 2) make implementation easier and more reliable.

Champion

Provide information about the champion. If the proposal is submitted by a subcommittee, include the name of the point person. He or she should have prepared this proposal and thoroughly understand all of the content. The point person must be present at the TC calls when this proposal is discussed.

Tracking information

Event	Date	Links
Stage 1 proposal accepted	13 March 2018	Minutes
Stage 2 proposal submitted	Submitted 26 March 2018, updated 9 April 2018	DITA source, HTML
Stage 2 proposal discussed	3 April and 10 April	Minutes for 3 April 2018, 10 April 2018
Stage 2 proposal approved	17 April	Minutes
Stage 3 proposal submitted to reviewers	December 2018	Stan Doherty, Eliot Kimber (extra review from Chris Nitchie)
Stage 3 proposal (initial version) submitted to TC	22 April 2019	Email to TC
Stage 3 proposal (with updated examples) submitted to TC	06 May 2019	SVN link

Approved technical requirements

- No change to grammar files. The attribute name is the same; chunk tokens are not defined in the grammar for 1.x and this will not change in 2.x.
- For some processors or output formats, chunking is irrelevant. Where it is irrelevant, nothing changes for processors.
- All previously defined tokens for `@chunk` will be removed from the specification.
- Two new tokens will be defined in the specification:

1. `chunk="combine"` This is roughly equivalent to `chunk="to-content"` in DITA 1.x.
   - When specified on a map, everything within the scope of that map is published as a single result document.
   - When specified on a branch of a map (or reference to a map), everything within the scope of that branch or reference is published as a single result document.
   - There is no way to undo the combine action for something within that scope.
2. `chunk="split"` This is roughly equivalent to `chunk="by-topic"` in DITA 1.x.

  - When specified on a `<topicref>`, it indicates that all topics *within the referenced document* should be split into multiple documents. ***For example***, in a context where each individual DITA document is published as a single HTML file, specifying `chunk="split"` on a reference to a document that contains five topics will result in 5 documents + 5 output files.
  - When a document is split, the effective map hierarchy remains the same. ***In the example from the previous item***, this means that evaluating `@chunk` for the one map reference would result in five references, which preserve any nesting hierarchy from that document.
  - When specified on a root map, it indicates that all references should treat `chunk="split"` as the default operation. Setting `chunk="combine"` anywhere will override that default for that section of the map.
  - When specified on a nested map, it indicates that the all references within the scope of that nested map should treat `chunk="split"` as the default operation. Setting `chunk="combine"` anywhere will override that default for that section of the map.
  - Specifying `chunk="split"` on any other container element that does not reference a document (such as on `<topicgroup>`) has no meaning.

3. While we view this use as unlikely, the attribute is left open for application-specific tokens.

Dependencies or interrelated proposals

N/A

Modified grammar files

N/A

Modified terminology

N/A

Modified specification documentation

The following section of the specification should be removed, and replaced with the updated topics attached below:

- 2.4.5 Chunking
  - 2.4.5.1 Using the `@chunk` attribute
  - 2.4.5.2 Chunking examples

The current definition of `@chunk` in the Attributes common to many map elements is general enough that it applies equally to DITA 1.3 and DITA 2.0.

Migration plans for backwards incompatibilities

The best way to address this change is a search/replace type operation that finds values in the `@chunk` attribute. When the value contains `to-content`, replace the entire value with `combine`. When the value contains `by-topic`, replace the entire value with `split`.

Chunking

Content often needs to be delivered in a different granularity than it is authored. The `@chunk` attribute enables map authors to specify that multiple source documents should be combined into a single document for delivery, or that a single source document should be split into multiple documents for delivery.

About the @chunk attribute

The `@chunk` attribute is intended to handle cases where the best organization for creating DITA topics is not equivalent to the best organization for publishing those topics.

The `@chunk` attribute is composed of a single token without any white space. DITA defines two tokens for the `@chunk` attribute, `combine` and `split`. Other tokens may be defined by applications but support for those tokens will vary.

chunk="combine"
: The `combine` token in `@chunk` is intended for cases where a publishing process normally results in a single output artifact for each source XML document. When some or all of those source XML documents are better presented as a single output artifact, setting `chunk="combine"` instructs a processor to combine the referenced source documents for rendering purposes.

chunk="split"
: The `split` token in `@chunk` is intended for cases where a publishing process normally results in a single output artifact for each single source XML document, regardless of how many DITA topics exist within each source document. When a source XML document containing many topics is better rendered as multiple output artifacts, setting `chunk="split"` instructs a processor to split each topic from the referenced source document into its own document for rendering purposes.

The following rules apply to all values of the `@chunk` attribute:

- The `@chunk` attribute describes how a processor can split or combine source DITA documents into alternate organizational schemes for rendering purposes. This means that the `@chunk` attribute is only relevant when the organization of source DITA documents has an effect on organization of published documents. When the source document organization has no effect on published output, such as when producing a single PDF or EPUB, implementations **MAY** ignore the `@chunk` attribute.
- When the `@chunk` attribute results in more or fewer documents based on the `combine` or `split` tokens, the hierarchy of topics within the resulting map and topic organization **SHOULD** match the hierarchy in the original topics and maps.
- When the `@chunk` attribute results in more or fewer documents, implementations **MAY** create their own naming schemes for those reorganized documents.
- The `@chunk` attribute does not cascade to nested `<topicref>` elements or to nested map references.
- `@chunk` attribute values apply to DITA topic documents referenced from a map. Processors **MAY** apply equivalent processing to non-DITA documents.

Using the @chunk attribute to combine documents

The `@chunk` attribute can be set to `chunk="combine"` to produce a single result document out of topic documents that are best managed independently.

The `combine` token in `@chunk` is intended for cases where a publishing process normally results in a single output artifact for each source XML document. When some or all of those source XML documents are better presented as a single output artifact, setting `chunk="combine"` instructs a processor to combine the referenced source documents for rendering purposes.

Processing chunk="combine"

- When specified on a map, all source DITA documents referenced by the map are treated as one DITA document.
- When specified on a branch of a map, all source DITA documents referenced within that branch are treated as one DITA document.

  Note This is true regardless of whether the element that specifies `@chunk` refers to a topic or specifies a heading. In cases such as `<topicgroup>` where a grouping element specifies `chunk="combine"`, the equivalent DITA document would be a single DITA document with a `<dita>` root element grouping peer topics.
- When specified on a reference to a map, all source DITA documents within the scope of the referenced map are treated as one DITA document.
- Once `chunk="combine"` is specified on a map, branch, or map reference, all source DITA documents grouped by that reference are treated as a single resource. Any additional `@chunk` attributes on elements within the hierarchy are ignored.

Using the @chunk attribute to split documents

The `@chunk` attribute can be set to `chunk="split"` to produce multiple result documents out of topics that are best managed within a single source document.

The `split` token in `@chunk` is intended for cases where a publishing process normally results in a single output artifact for each single source XML document, regardless of how many DITA topics exist within each source document. When a source XML document containing many topics is better rendered as multiple output artifacts, setting `chunk="split"` instructs a processor to split each topic from the referenced source document into its own document for rendering purposes.

Processing chunk="split"

- When specified on a `<topicref>` element that refers to a source DITA document, it indicates that all topics within the referenced document should be rendered as individual documents.
- When specified on an element such as `<topicgroup>` that does not refer to a topic or result in a published topic, the attribute has no meaning.
- When specified on a map, `chunk="split"` sets a default operation for all source DITA documents in the map (outside the context of relationship tables). The default `split` value is used except where a `combine` value is encountered, in which case `combine` takes over for that entire branch.

Using the @chunk attribute for other purposes

Additional tokens can be defined by applications for use in the `@chunk` attribute. These tokens are necessarily implementation dependent and might not be supported by other applications.

Examples of the @chunk attribute

These examples illustrate the processing expectations for various scenarios that involve the `@chunk` attribute. Processing examples use either before and after sample markup or expanded syntax that shows the equivalent markup without the `@chunk` attribute.

Note Examples use sample files with modified file names to help illustrate equivalent before and after resolution of `@chunk` attributes. However, there is no requirement for implementations processing `@chunk` to generate files, as long as the rendered result is split or combined as described. If generating files, actual file names are implementation dependent.

Example: Using @chunk to combine all documents into one

Where a publishing system would typically render each topic document as an independent result document, a single `@chunk` attribute can be used to render all content as a single document.

Consider the following source documents, with root map `input.ditamap`:

Figure 58: Input map without chunking

```
input.ditamap:
<map>
  <title>Lesson plan</title>
  <topicref href="background.dita">
    <!-- more background topics -->
  </topicref>
  <topicref href="goals.dita">
    <!-- more goal topics -->
  </topicref>
  <!-- more topics -->
</map>

background.dita:
<topic id="background">
  <title>Prerequisite concepts</title>
  <shortdesc>This information is necessary before starting</shortdesc>
  <body><!-- ...background content... --></body>
</topic>

goals.dita:
<topic id="goals">
  <title>Lesson gals</title>
  <shortdesc>After you complete the lesson, ...</shortdesc>
  <body><!-- ...goal content... --></body>
</topic>
```

For many systems or output formats, each document in the map will be rendered as an independent document. For example, rendering this map as HTML5 might result in `background.html` and `goals.html` (along with other HTML5 files). In such cases, if output requirements demand only a single result document, adding `chunk="combine"` to the root map element instructs a processor to render one document that combines all topics.

Figure 59: Root map with chunking specified

```
input.ditamap:
<map chunk="combine">
  <title>Lesson plan</title>
  <topicref href="background.dita">
    <!-- more background topics -->
  </topicref>
  <topicref href="goals.dita">
    <!-- more goal topics -->
  </topicref>
  <!-- more topics -->
</map>
```

The result of evaluating this `@chunk` attribute is equivalent to the following map and topic document; content from all topics within the map has now been combined into a single result, with the topic order and topic nesting structure matching the original map hierarchy.

Figure 60: Equivalent source content

```
input.ditamap:
<map>
  <title>Lesson plan</title>
  <topicref href="input.dita"/>
</map>
```

```
input.dita:
<dita>
  <!-- original content of background.dita -->
  <topic id="background">
    <title>Prerequisite concepts</title>
    <shortdesc>This information is necessary before starting</shortdesc>
    <body><!-- ...background content... --></body>
    <!-- more background topics -->
  </topic>
  <!-- original content of goals.dita -->
  <topic id="goals">
    <title>Lesson gals</title>
    <shortdesc>After you complete the lesson, ...</shortdesc>
    <body><!-- ...goal content... --></body>
    <!-- more goal topics -->
  </topic>
  <!-- more topics -->
</dita>
```

Example: Using @chunk *to render a single document from one branch*

Where a publishing system would typically render each topic document as an independent result document, a single `@chunk` attribute can be used to render individual branches of a map as single documents.

Consider the following source documents, with root map `input.ditamap`:

Figure 61: Input map without chunking

```
input.ditamap:
<map>
  <title>Lesson plan</title>
  <topicref href="goals.dita">
    <!-- more goal topics -->
  </topicref>
  <topicref href="firstLesson.dita">
    <!-- more tasks in the first lesson -->
  </topicref>
  <topicref href="nextLesson.dita">
    <!-- more tasks in the next lesson -->
  </topicref>
  <!-- More branches -->
</map>

firstLesson.dita:
<task id="firstLesson">
  <title>Starting to work with scissors</title>
  <shortdesc>This lesson will teach ...</shortdesc>
  <taskbody><!-- ... --></taskbody>
</task>

nextLesson.dita:
<task id="nextLesson">
  <title>Advanced cutting</title>
  <shortdesc>This lesson will introduce complicated shapes...</shortdesc>
  <taskbody><!-- ... --></taskbody>
</task>
```

For many systems or output formats, each document in the map will be rendered as an independent document. For example, rendering this map as HTML5 might result in `goals.html`, `firstLesson.html`, and `nextLesson.html`, while child documents within each branch would each result in their own HTML files.

When output requirements demand some portions of the map be combined into a single document, adding `chunk="combine"` to a branch of the map instructs a processor to render one document that combines all topics in that branch. This is particularly useful when the topics need to be rendered

independently for other contexts, or when the way topics are contributed makes creating a single source document impossible.

In the following sample, the original map is updated with `@chunk` attributes to indicate that each lesson branch should render as a single result document; topics in the first branch with `goals.dita` are not affected as a result of the `@chunk` attribute.

Figure 62: Map with chunking specified for one branch

```
input.ditamap:
<map>
  <title>Lesson plan</title>
  <topicref href="goals.dita">
    <!-- more goal topics -->
  </topicref>
  <topicref href="firstLesson.dita" chunk="combine">
    <!-- more tasks in the first lesson -->
  </topicref>
  <topicref href="nextLesson.dita" chunk="combine">
    <!-- more tasks in the next lesson -->
  </topicref>
  <!-- More branches -->
</map>
```

The result of evaluating this `@chunk` attribute is equivalent to the following map and topic documents. Content from each combined branch has now been combined into a single result document for each branch, with the order and topic nesting structure matching the original map hierarchy. Content from outside of those branches remains unchanged.

Figure 63: Equivalent source content

```
input.ditamap:
<map>
  <title>Lesson plan</title>
  <topicref href="goals.dita">
    <!-- more goal topics -->
  </topicref>
  <topicref href="firstLesson.dita"/>
  <topicref href="nextLesson.dita"/>
  <!-- More branches -->
</map>

firstLesson.dita:
<task id="firstLesson">
  <title>Starting to work with scissors</title>
  <shortdesc>This lesson will teach ...</shortdesc>
  <taskbody><!-- ... --></taskbody>
  <!-- more tasks in the first lesson -->
</task>

nextLesson.dita:
<task id="nextLesson">
  <title>Advanced cutting</title>
  <shortdesc>This lesson will introduce complicated shapes...</shortdesc>
  <taskbody><!-- ... --></taskbody>
  <!-- more tasks in the next lesson -->
</task>
```

Example: Using @chunk to combine a group of topics

The `@chunk` attribute can be used on grouping elements to combine multiple source documents into one result document.

Assume the following map `input.ditamap`, where `@chunk` is used on both `<topicgroup>` and `<topichead>`.

```
<map>
  <title>Groups are combined</title>
  <topicgroup chunk="combine">
    <topicref href="ingroup1.dita"/>
    <topicref href="ingroup2.dita"/>
  </topicgroup>
  <topichead chunk="combine">
    <topicmeta><navtitle>Heading for a branch</navtitle></topicmeta>
    <topicref href="inhead1.dita"/>
    <topicref href="inhead2.dita"/>
  </topichead>
</map>
```

- The result of evaluating the `@chunk` attribute on `<topicgroup>` is equivalent to a single DITA document with the content of both `ingroup1.dita` and `ingroup2.dita`.
- The `@chunk` attribute on `<topichead>` also results in a single DITA document. In many applications, a `<topichead>` is equivalent to a single title-only topic; in that case, the chunked result is equivalent to a root topic with the title "Heading for a branch", containing as child topics the content of both `inhead1.dita` and `inhead2.dita`. If `<topichead>` is ignorable in the current processing context, the chunked result would be equivalent to processing `<topicgroup>` (a single DITA document with the content of both `inhead1.dita` and `inhead2.dita`).

Figure 64: Equivalent source content

```
<map>
  <title>Groups are combined</title>
  <topicref href="chunkgroup-1.dita"/>
  <topicref href="chunkgroup-2.dita"/>
</map>

chunkgroup-1.dita
<dita>
  <!-- content of ingroup1.dita -->
  <!-- content of ingroup2.dita -->
</dita>

chunkgroup-2.dita
<dita>
  <topic id="head">
    <title>Heading for a branch</title>
    <!-- content of inhead1.dita -->
    <!-- content of inhead2.dita -->
  </topic>
</dita>
```

Example: Using @chunk to combine nested documents

Special attention is necessary when combining a nested map hierarchy that includes documents with their own nested topics.

Consider the following source map `input.ditamap`:

Figure 65: Input map without chunking

```
input.ditamap:
<map chunk="combine">
```

```
  <title>Generation example</title>
  <topicref href="ancestor.dita">
    <topicref href="middle.dita">
      <topicref href="child.dita"/>
    </topicref>
  </topicref>
</map>
```

In this case, the `@chunk` attribute instructs a processor to treat the three topics as a single combined DITA document, while preserving the original map hierarchy. Now consider the following three topic documents, each of which includes nested or peer topics:

Figure 66: Source documents with nested structures

```
ancestor.dita:
<dita>
  <topic id="ancestor-first">
    <title>First major topic in ancestor composite doc</title>
    <!-- ...topic content... -->
  </topic>
  <!-- more topics in ancestor composite doc -->
  <topic id="ancestor-last">
    <title>Last major topic in ancestor composite doc</title>
    <!-- ...topic content... -->
    <topic id="ancestor-last-child">
      <title>Child of last major topic in ancestor composite doc</title>
      <!-- ...topic content... -->
    </topic>
  </topic>
</dita>

middle.dita:
<topic id="middle-root">
  <title>Root topic in middle doc</title>
  <body><!-- ... --></body>
  <topic id="middle-child">
    <title>Child of root topic in middle doc</title>
    <!-- body content, maybe more children of middle topic's root -->
  </topic>
</topic>

child.dita:
<topic id="child">
  <title>Small child topic</title>
  <!-- small child topic content -->
</topic>
```

When `chunk="combine"` is evaluated, the three source documents are combined into one. Both the ancestor and middle documents have child topics that must be taken into account.

- `ancestor.dita` has a root `<dita>` element, so content from each nested topic reference is located after any nested topics within the final child of the `<dita>` element.
- `middle.dita` does not have `<dita>` but does have a nested topic, so content from any nested topic references is located after that nested topic.

Figure 67: Equivalent source content

```
input.ditamap:
<map>
  <title>Generation example</title>
  <topicref href="input.dita"/>
</map>

input.dita:
<dita>
  <topic id="ancestor-first">
    <title>First major topic in ancestor composite doc</title>
```

```
    <!-- ...topic content... -->
  </topic>
  <!-- more topics in ancestor composite doc -->
  <topic id="ancestor-last">
    <title>Last major topic in ancestor composite doc</title>
    <!-- ...topic content... -->
    <topic id="ancestor-last-child">
      <title>Child of last major topic in ancestor composite doc</title>
      <!-- ...topic content... -->
    </topic>
    <!-- content of middle.dita combined here -->
    <topic id="middle-root">
      <title>Root topic in middle doc</title>
      <body><!-- ... --></body>
      <topic id="middle-child">
        <title>Child of root topic in middle doc</title>
        <!-- body content, maybe more children of middle topic's root -->
      </topic>
      <!-- content of child.dita combined here -->
      <topic id="child">
        <title>Small child topic</title>
        <!-- small child topic content -->
      </topic>
    </topic>
  </topic>
</dita>
```

Example: Using @chunk to split documents

When topics are most easily created or generated in a single DITA document, `chunk="split"` will instruct processors to render them individually when possible.

Splitting a single document in the map

Consider the following example, where a map includes generated topics used to document message numbers from an application:

Figure 68: Source map and topics

```
<map>
  <title>Message guide for WidgetAnalyzer</title>
  <topicref href="about.dita">
    <topicref href="messages-install.dita"/>
    <topicref href="messages-run.dita"/>
    <topicref href="messages-other.dita"/>
  </topicref>
</map>

about.dita:
<topic id="about">
  <title>About this guide</title>
  <shortdesc>Warnings or errors will appear if...<shortdesc>
</topic>

messages-install.dita:
<dita>
  <topic id="INS001">
    <title>INS001: Installation failure</title>
    <!-- explanation and recovery... -->
  </topic>
  <!-- more install messages... -->
</dita>

messages-run.dita:
<dita>
  <topic id="RUN001">
    <title>RUN001: Failed to initialize</title>
    <!-- explanation and recovery... -->
  </topic>
  <!-- hundreds of messages... -->
  <topic id="RUN999">
```

```
    <title>RUN999: Out of memory</title>
    <!-- explanation and recovery... -->
  </topic>
</dita>

messages-other.dita:
<topic id="othermsg">
  <title>Other messages</title>
  <shortdesc>You could also encounter ...</shortdesc>
  <topic id="OTHER001">
    <title>OTHER001: Analyzer is tired</title>
    <!-- explanation and recovery... -->
  </topic>
  <topic id="OTHER002">
    <title>OTHER002: Analyzer needs to be updated</title>
    <!-- explanation and recovery... -->
  </topic>
</topic>
```

In a normal build to HTML5, this map might result in four result documents `about.html`, `messages-install.html`, `messages-run.html`, and `messages-other.html`. With hundreds of messages in `messages-run.dita`, it will be better in some situations to render one result document for each message topic in the document. This is done by setting `chunk="split"` on the topic reference.

Figure 69: Splitting all topics in one document

```
<map>
  <title>Message guide for WidgetAnalyzer</title>
  <topicref href="about.dita">
    <topicref href="messages-install.dita"/>
    <topicref href="messages-run.dita" chunk="split"/>
    <topicref href="messages-other.dita"/>
  </topicref>
</map>
```

The result of evaluating `@chunk` in this case is equivalent to the following map and topics. While `messages-run.dita` is now split into hundreds of topics, other topics in the map are unaffected.

Figure 70: Equivalent source content

```
<map>
  <title>Message guide for WidgetAnalyzer</title>
  <topicref href="about.dita">
    <topicref href="messages-install.dita"/>
    <topicref href="RUN001.dita"/>
    <!-- hundreds of messages... -->
    <topicref href="RUN999.dita"/>
    <topicref href="messages-other.dita"/>
  </topicref>
</map>

RUN001.dita:
<topic id="RUN001">
  <title>RUN001: Failed to initialize</title>
  <!-- explanation and recovery... -->
</topic>

RUN999.dita:
<topic id="RUN999">
  <title>RUN999: Out of memory</title>
  <!-- explanation and recovery... -->
</topic>
```

Note Because the `@chunk` attribute does not cascade, even if the reference to `messages-install.dita` had child topic references, they would be unaffected by the `chunk="split"` value in this example.

Splitting every document in the map

Similarly, because setting `chunk="split"` on the map element sets a default for the entire map, the following change to the original map would result in *every* referenced DITA document being split into one document per topic. The only source document not affected by this split is `about.dita`, because it only contained a single topic to begin with.

Figure 71: Splitting every topic in the map

```
<map chunk="split">
  <title>Message guide for WidgetAnalyzer</title>
  <topicref href="about.dita">
    <topicref href="messages-install.dita"/>
    <topicref href="messages-run.dita"/>
    <topicref href="messages-other.dita"/>
  </topicref>
</map>
```

Using `chunk="split"` on the map is equivalent to the following structure:

- `about.dita` is unchanged.
- `messages-install.dita` is split into one document per message (as in the previous example that split `messages-run.dita`).
- `messages-run.dita` is split exactly as in the previous example.
- `messages-other.dita` contains a root topic and two child topics, so it results in three documents. The hierarchy of those documents is preserved in the map.

Figure 72: Equivalent source content

```
<map>
  <title>Message guide for WidgetAnalyzer</title>
  <topicref href="about.dita">
    <topicref href="INS001.dita"/>
    <!-- more install messages... -->
    <topicref href="RUN001.dita"/>
    <!-- hundreds of messages... -->
    <topicref href="RUN999.dita"/>
    <topicref href="othermsg.dita">
      <topicref href="OTHER001.dita"/>
      <topicref href="OTHER002.dita"/>
    </topicref>
  </topicref>
</map>

INS001.dita:
<topic id="INS001">
  <title>INS001: Installation failure</title>
  <!-- explanation and recovery... -->
</topic>

RUN001.dita:
<topic id="RUN001">
  <title>RUN001: Failed to initialize</title>
  <!-- explanation and recovery... -->
</topic>

RUN999.dita:
<topic id="RUN999">
  <title>RUN999: Out of memory</title>
  <!-- explanation and recovery... -->
</topic>

othermsg.dita:
<topic id="othermsg">
  <title>Other messages</title>
  <shortdesc>You could also encounter ...</shortdesc>
</topic>
```

```
OTHER001.dita:
<topic id="OTHER001">
  <title>OTHER001: Analyzer is tired</title>
  <!-- explanation and recovery... -->
</topic>

OTHER002.dita:
<topic id="OTHER002">
  <title>OTHER002: Analyzer needs to be updated</title>
  <!-- explanation and recovery... -->
</topic>
```

Example: Using @chunk *to split nested documents*

Special attention is necessary when evaluating the map hierarchy that results from splitting a documents with their own nested topics.

Consider the following source map `input.ditamap`:

Figure 73: Input map without chunking

```
input.ditamap:
<map chunk="split">
  <title>Generation example</title>
  <topicref href="ancestor.dita">
    <topicref href="middle.dita">
      <topicref href="child.dita"/>
    </topicref>
  </topicref>
</map>
```

In this case, the `@chunk` attribute instructs a processor to render every topic in each of the three documents as its own document, while preserving any hierarchy from those documents. Now consider the following three topic documents, each of which includes nested or peer topics:

Figure 74: Source documents with nested structures

```
ancestor.dita:
<dita>
  <topic id="ancestor-first">
    <title>First major topic in ancestor composite doc</title>
    <!-- ...topic content... -->
  </topic>
  <!-- more topics in ancestor composite doc -->
  <topic id="ancestor-last">
    <title>Last major topic in ancestor composite doc</title>
    <!-- ...topic content... -->
    <topic id="ancestor-last-child">
      <title>Child of last major topic in ancestor composite doc</title>
      <!-- ...topic content... -->
    </topic>
  </topic>
</dita>

middle.dita:
<topic id="middle-root">
  <title>Root topic in middle doc</title>
  <body><!-- ... --></body>
  <topic id="middle-child">
    <title>Child of root topic in middle doc</title>
    <!-- body content -->
  </topic>
</topic>

child.dita:
<topic id="child">
  <title>Small child topic</title>
```

```
  <!-- small child topic content -->
</topic>
```

When `chunk="split"` is evaluated, both `ancestor.dita` and `middle.dita` are split and treated as multiple DITA topic documents. `child.dita` is only a single topic and has nothing to split.

- `ancestor.dita` has a root `<dita>` element, so it results in multiple peer topic references (or branches) in the map. Topic references nested within the original reference to `ancestor.dita` are now located within the reference to "ancestor-last" (the last topic child of the `<dita>` element).
- `middle.dita` has nested topics, so results in its own new hierarchy within the map. Content from the nested topic reference is now located within the reference to the root topic from `middle.dita`, but after any references to child topics.

Figure 75: Equivalent source content

```
input.ditamap:
<map chunk="split">
  <title>Generation example</title>
  <topicref href="ancestor-first.dita"/>
  <!-- more topics in ancestor composite doc -->
  <topicref href="ancestor-last.dita">
    <topicref href="ancestor-last-child.dita"/>
    <!-- middle.dita now located here, as final child of
         final topic child of <dita> in ancestor.dita -->
    <topicref href="middle-root.dita">
      <topicref href="middle-child.dita"/>
      <!-- child.dita now located here, as final topic
           child root topic in middle.dita ancestor.dita -->
      <topicref href="child.dita"/>
    </topicref>
  </topicref>
</map>
```

Example: When @chunk is ignored

The `@chunk` attribute is ignored in some cases, such as when `chunk="combine"` is already in effect or when `chunk="split"` is specified on a grouping element.

Ignoring @chunk when already combining topics

In the following example, evaluating `@chunk` results in one rendered document for each branche of the map. Any additional `@chunk` values within that branch are ignored (including `@chunk` values within any referenced maps).

Figure 76: Chunk within a combined branch

```
<map>
  <title>Ignoring chunking when already combined</title>

  <topicref href="branchOne.dita" chunk="combine">
    <!-- @chunk ignored for branchOneChild.dita -->
    <topicref href="branchOneChild.dita" chunk="split"/>
  </topicref>

  <topicref href="branchTwo.dita" chunk="combine">
    <!-- Any @chunk within submap.ditamap is ignored -->
    <topicref href="submap.ditamap" format="ditamap"/>
  </topicref>
```

Ignoring @chunk on a grouping element

In the following example, `chunk="split"` is specified on two grouping elements.

Figure 77: Chunk within a combined branch

```
<map>
  <title>Trying to "split" groups</title>
  <topicgroup chunk="split">
    <topicref href="ingroup1.dita">...</topicref>
    <topicref href="ingroup2.dita">...</topicref>
  </topicgroup>
  <topichead chunk="split">
    <topicmeta><navtitle>Heading for a branch</navtitle></topicmeta>
    <topicref href="inhead1.dita">...</topicref>
    <topicref href="inhead2.dita">...</topicref>
  </topichead>
</map>
```

- The `@chunk` attribute on the `<topicgroup>` is ignored; it does not cascade, and there is no referenced topic, so it has no effect.
- In some cases, an implementation might treat the `<topichead>` element as equivalent to a single title-only topic, while in other cases it might be ignored. In either case the `@chunk` value has no effect. If the `<topichead>` is treated as a title-only topic, it cannot be split further; if it is ignored for the current processing context, it is no different than the `<topicgroup>`.

Example: Combining topics within a split context

While `@chunk` attributes are ignored when a "combine" action is already in effect, it is possible to use `chunk="combine"` when `split` is otherwise in effect.

Assume the following map, where `chunk="split"` on the root element means that all topic documents within this map structure are split by default, but a branch within the map sets `chunk="combine"`.

Figure 78: Map with default "split" action, that also uses "combine"

```
<map chunk="split">
  <title>Split most, but not one branch</title>
  <topicref href="splitme.dita">...</topicref>
  <topicref href="exception.dita" chunk="combine">...</topicref>
  <topicref href="splitmetoo.dita">...</topicref>
</topicref>
```

Assume as well that no other `@chunk` attributes are specified in this map. The following points are true when `@chunk` is evaluated:

1. The document `splitme.dita` is treated as multiple split documents when it contains more than one topic. The same is true for any other document within that branch.
2. The second branch (beginning with `exception.dita`) is treated as a single DITA document, combining all topic documents within that branch.
3. The document `splitmetoo.dita` is treated as multiple split documents when it contains more than one topic. The same is true for any other document within that branch.

Example: Managing links when chunking

Link management with `@chunk` is often straightforward; in most cases where URI-based linking is ambiguous, using indirect links and `@keyref` will give the correct result.

Input topics for following examples

The following map and topics are used for all examples in this topic.

Figure 79: input.ditamap

```
<map>
  <title>Map with chunks and links</title>

  <keydef href="splitThis.dita" keys="splitThisKey"/>
  <keydef href="splitThis.dita#splitThisChild" keys="splitThisChildKey"/>

  <topicref href="splitThis.dita" chunk="split" keys="explicitSplitKey"/>
  <topicref href="combineThis.dita" keys="combineThisKey">
    <topicref href="combinedChild.dita" keys="combinedChildKey"/>
  </topicref>
</map>
```

Figure 80: Topics used by input.ditamap

```
splitThis.dita:
<topic id="splitThisRoot">
  <title>Root topic in split document</title>
  <!-- ... -->
  <topic id="splitThisChild">
    <title>Child topic in split document</title>
    <!-- ... -->
  </topic>
</topic>

combineThis.dita:
<topic id="combineThisRoot">
  <title>Root topic in combined document</title>
  <!-- ... -->
  <topic id="combineThisChild">
    <title>Child topic in combined document</title>
    <!-- ... -->
  </topic>
</topic>

combinedChild.dita:
<topic id="combinedChildRoot">
  <title>Topic in child document, combined with parent</title>
  <!-- ... -->
</topic>
```

Topics that are rendered only once when publishing

Assume that the map above is a root map or is used by another map does not otherwise render the three topic documents. In that case, the following is true:

- `splitThis.dita` is rendered as two documents. For this example, assume a processor creates two documents with names taken from the topic ID, so that topic becomes `splitThisRoot.dita` and `splitThisChild.dita`.
- The branch with `combineThis.dita` is rendered as one document together with the content of `combinedChild.dita`. For this example, assume a processor merges the child topic into the file `combineThis.dita`.

- All links using `href="splitThis.dita"`, `keyref="splitThisKey"`, or `keyref="explicitSplitKey"` will resolve to `splitThisRoot.dita` (the only rendered instance of that topic).
- All links using `href="splitThis.dita#splitThisChild"` or `keyref="splitThisChildKey"` will resolve to `splitThisChild.dita` (the only rendered instance of that topic).
- All links using `href="combinedChild.dita"` or `keyref="combinedChildKey"` will resolve to that topic within `combineThis.dita` (the only rendered instance of that topic).

Topics that are rendered twice when publishing

Now assume that the map above is reused in another context that also renders all three topic documents as originally authored. As a result, each of the three documents in this map (`splitThis.dita`, `combineThis.dita`, and `combinedChild.dita`) are rendered more than once.

When each of these documents is rendered twice, the following is true:

- The original source document `splitThis.dita` is rendered twice. Based on the map above, assume a processor creates two documents with names taken from the topic ID, so that topic becomes `splitThisRoot.dita` and `splitThisChild.dita`. At the same time, `splitThis.dita` is rendered *in another context* as a single document, with a different name.
- Based on the map above, the branch that starts with the original source document `combineThis.dita` is rendered as one document combined with the content of `combinedChild.dita`. At the same time, those two documents are rendered in another context as individual documents. For this example, assume a processor generates the combined document using the generated name `combinThis-2.dita`, while the documents `combineThis.dita` and `combinedChild.dita` retain their names in their other context..
- All links in this map using the direct URI references `href="splitThis.dita"`, `href="splitThis.dita#splitThisChild"`, `href="combineThis.dita"`, or `href="combinedChild.dita"` are now ambiguous. They could go to the chunked instance from this map, or to the individual topics in the other context. Implementations will have to guess which topic to target: the split or combined instances from this map or versions in the alternate context from the root map.
- All links using indirect key-based references `keyref="splitThisKey"` or `keyref="splitThisChildKey"` are also ambiguous, because the key definitions are not associated explicitly with the chunked or not-chunked instance. If key scopes are used, applications might more reliably guess that the intended target is the split copy in this map, but this is not guaranteed.
- All links using `keyref="explicitSplitKey"`, `keyref="combinedThisKey"`, or `keyref="combinedChildKey"` are unambiguous; they can only resolve to the chunked instance from this submap, because they are defined directly within the chunk context.
- There is no way to unambiguously link to the child document that will result from splitting `splitThis.dita`. This is because it is only possible for the element using `@chunk` to associate a key definition with the first or root topic in the document. While other key definition elements can be used to associate keys with other topics in the same document, that can only be done outside of the navigation context that uses `@chunk`; as a result, a processor cannot guarantee whether the intended link target is the split topic from the `@chunk` context, or a use of the same topic in the second context. It is possible for an implementation to define its own way to resolve this ambiguity; however, if a situation requires both multiple instances of split topics and unambiguous cross-implementation links to those split topics, alternate reuse mechanisms need to be considered.

3.1.9 Stage 3: #253 Indexing changes

Refactor indexing to remove redundant elements and reduce complexity

Champion

Kristen James Eberlein, Eberlein Consulting LLC

Tracking information

Event	Date	Links
Stage 1 proposal accepted	11 June 2019	Minutes, 11 June 2019
Stage 2 proposal submitted	14 June 2019	DITA PDF
Stage 2 proposal discussed	18 June 2019	Minutes, 18 June 2019
Stage 2 proposal approved	02 July 2019	Minutes, 02 July 2019
Stage 3 proposal submitted to reviewers	02 July 2019	Bill Burns Eliot Kimber Dawn Stevens
Stage 3 proposal (this document) submitted to TC	18 July 2019	DITA PDF

Approved technical requirements

- Remove the indexing domain, and add `<index-see>` and `<index-see-also>` to the base
- Remove `<index-base>`
- Remove `<index-sort-as>`

Dependencies or interrelated proposals

None

Modified grammar files

The following files must be modified:

DTDs

- `basemap.dtd`
- `basetopic.dtd`
- `commonElementsMod.dtd`
- `commonElementsMod.ent`

RNG

- `basemap.rng`
- `basetopic.rng`
- `commonElementsMod.rng`

In the content below, the following conventions are used:

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

Figure 81: Changes to basemap.dtd

```
<!-- ============================================================= -->
<!--              DOMAIN ENTITY DECLARATIONS                       -->
<!-- ============================================================= -->
...
<!ENTITY % indexing-d-dec
  PUBLIC "-//OASIS//ENTITIES DITA 1.3 Indexing Domain//EN"
         "indexingDomain.ent"
>%indexing-d-dec;
...
<!-- ============================================================= -->
<!--                     DOMAIN EXTENSIONS                         -->
<!-- ============================================================= -->
<!--                     One for each extended base element, with
                         the name of the domain(s) in which the
                         extension was declared                   -->
...
<!ENTITY % index-base    "index-base |
                          %indexing-d-index-base;
                         ">
...
<!-- ============================================================= -->
<!--                     DOMAINS ATTRIBUTE OVERRIDE                -->
<!-- ============================================================= -->

<!ENTITY included-domains
                           "&mapgroup-d-att;
                            &delay-d-att;
                            &deliveryTargetAtt-d-att;
                            &ditavalref-d-att;
                            &indexing-d-att;
                            &hazard-d-att;
                            &hi-d-att;
                            &ut-d-att;
  "
>
...
<!-- ============================================================= -->
<!--                     DOMAIN ELEMENT INTEGRATION                -->
<!-- ============================================================= -->
...

<!ENTITY % indexing-d-def
  PUBLIC "-//OASIS//ELEMENTS DITA 1.3 Indexing Domain//EN"
         "indexingDomain.mod"
>%indexing-d-def;
```

Figure 82: Changes to basetopic.dtd

```
<!-- ============================================================= -->
<!--              DOMAIN ENTITY DECLARATIONS                       -->
<!-- ============================================================= -->
...
<!ENTITY % indexing-d-dec
  PUBLIC "-//OASIS//ENTITIES DITA 1.3 Indexing Domain//EN"
         "indexingDomain.ent"
>%indexing-d-dec;
...
<!-- ============================================================= -->
<!--                     DOMAIN EXTENSIONS                         -->
<!-- ============================================================= -->
<!--                     One for each extended base element, with
                         the name of the domain(s) in which the
```

```
                          extension was declared                      -->

<!ENTITY % index-base    "index-base |
                          %indexing-d-index-base;
                         ">
...
<!-- ============================================================= -->
<!--                    DOMAINS ATTRIBUTE OVERRIDE                 -->
<!-- ============================================================= -->

<!ENTITY included-domains
                           "&deliveryTargetAtt-d-att;
                            &hazard-d-att;
                            &hi-d-att;
                            &indexing-d-att;
                            &ut-d-att;
  "
>
...
<!-- ============================================================= -->
<!--                    DOMAIN ELEMENT INTEGRATION                 -->
<!-- ============================================================= -->
...
<!ENTITY % indexing-d-def
  PUBLIC "-//OASIS//ELEMENTS DITA 1.3 Indexing Domain//EN"
         "indexingDomain.mod"
>%indexing-d-def;
...
```

Figure 83: Changes to commonElements.ent

```
...
<!-- ============================================================= -->
<!--                    ELEMENT NAME ENTITIES                      -->
<!-- ============================================================= -->
...
<!ENTITY % index-base   "index-base"                                >
...
```

Figure 84: Changes to commonElements.mod

```
...
<!--                    LONG NAME: Index Term                      -->
<!ENTITY % indexterm.content
                         "(%words.cnt; |
                           %ph; |
                           %indexterm; |
                           %index-see;
                           %index-see-also;
                           %index-base;)*"
>
...
<!--                    LONG NAME: Index Base                      -->
<!ENTITY % index-base.content
                         "(%words.cnt; |
                           %indexterm;)*"
>
<!ENTITY % index-base.attributes
             "keyref
                          CDATA
                                      #IMPLIED
              %univ-atts;"
>
<!ELEMENT  index-base %index-base.content;>
<!ATTLIST  index-base %index-base.attributes;>
<!--                    LONG NAME: Index See                       -->
<!ENTITY % index-see.content
                         "(%words.cnt; |
                           %ph; |
                           %indexterm;)*"
>
```

```
<!ENTITY % index-see.attributes
              "keyref
                          CDATA
                                        #IMPLIED
               %univ-atts;"
>
<!ELEMENT  index-see %index-see.content;>
<!ATTLIST  index-see %index-see.attributes;>


<!--                    LONG NAME: Index See Also                  -->
<!ENTITY % index-see-also.content
                       "(%words.cnt; |
                         %ph; |
                         %indexterm;)*"
>
<!ENTITY % index-see-also.attributes
              "keyref
                          CDATA
                                        #IMPLIED
               %univ-atts;"
>
<!ELEMENT  index-see-also %index-see-also.content;>
<!ATTLIST  index-see-also %index-see-also.attributes;>
...
<!-- ============================================================= -->
<!--             SPECIALIZATION ATTRIBUTE DECLARATIONS             -->
<!-- ============================================================= -->
...
<!ATTLIST  index-base    %global-atts;  class CDATA "- topic/index-base " >
<!ATTLIST  index-see     %global-atts;  class CDATA "- topic/index-see ">
<!ATTLIST  index-see-also %global-atts;  class CDATA "- topic/index-see-also ">
...
```

Figure 85: Changes to basemap.rng

```
...
  <div>
      <a:documentation>DITA DOMAINS ATTRIBUTE</a:documentation>

      <define name="domains-att">
         <optional>
            <attribute name="domains"
                       a:defaultValue="
                         (topic delay-d)
                         (map ditavalref-d)
                         (topic hazard-d)
                         (topic hi-d)
                         (topic indexing-d)
                         (map mapgroup-d)
                         (topic ut-d)
                         a(props deliveryTarget)"/>
         </optional>
      </define>


  </div>
  <div>
      <a:documentation>MODULE INCLUSIONS</a:documentation>

      <include href="mapMod.rng"/>
      <include href="mapGroupDomain.rng"/>

      <include href="delayResolutionDomain.rng"/>
      <include href="deliveryTargetAttDomain.rng" dita:since="1.3"/>
      <include href="ditavalrefDomain.rng" dita:since="1.3"/>
      <include href="indexingDomain.rng"/>
      <include href="hazardstatementDomain.rng"/>
      <include href="highlightDomain.rng"/>
      <include href="utilitiesDomain.rng"/>
```

```
  </div>
...
```

Figure 86: Changes to basetopic.rng

```
...
  <div>
    <a:documentation>DOMAINS ATTRIBUTE</a:documentation>
    <define name="domains-att">
      <optional>
        <attribute name="domains"
          a:defaultValue="(topic hazard-d)
                          (topic hi-d)
                          (topic indexing-d)
                          (topic ut-d)
                          a(props deliveryTarget)"
        />
      </optional>
    </define>
  </div>
  <div>
    <a:documentation>MODULE INCLUSIONS</a:documentation>
    <include href="topicMod.rng">
      <define name="topic-info-types">
        <ref name="topic.element"/>
      </define>

    </include>
    <include href="deliveryTargetAttDomain.rng" dita:since="1.3"/>
    <include href="hazardstatementDomain.rng" dita:since="1.3"/>
    <include href="highlightDomain.rng"/>
    <include href="indexingDomain.rng"/>
    <include href="utilitiesDomain.rng"/>
  </div>
...
```

Figure 87: Changes to commonElementsMod.rng

```
...
    <define name="index-base">
      <ref name="index-base.element"/>
    </define>
    <define name="index-see">
      <ref name="index-see.element"/>
    </define>
    <define name="index-see-also">
      <ref name="index-see-also.element"/>
    </define>
...
<div>
        <a:documentation>LONG NAME: Index Term</a:documentation>
        <define name="indexterm.content">
          <zeroOrMore>
            <choice>
              <ref name="words.cnt"/>
              <ref name="ph" dita:since="1.3"/>
              <ref name="indexterm"/>
              <ref name="index-base"/>
              <ref name="index-see"/>
              <ref name="index-see-also"/>
            </choice>
          </zeroOrMore>
        </define>
        <define name="indexterm.attributes">
          <optional>
            <attribute name="keyref"/>
          </optional>
          <optional>
            <attribute name="start"/>
          </optional>
          <optional>
```

```
            <attribute name="end"/>
          </optional>
          <ref name="univ-atts"/>
        </define>
        <define name="indexterm.element">
          <element name="indexterm" dita:longName="Index Term">
            <a:documentation>An &lt;indexterm> element allows the author to indicate that a
certain word or phrase should produce an index entry in the generated index. Category:
Miscellaneous
              elements</a:documentation>
            <ref name="indexterm.attlist"/>
            <ref name="indexterm.content"/>
          </element>
        </define>
        <define name="indexterm.attlist" combine="interleave">
          <ref name="indexterm.attributes"/>
        </define>
      </div>
   <div>
      <a:documentation>LONG NAME: Index See</a:documentation>
      <define name="index-see.content">
        <zeroOrMore>
          <choice>
            <ref name="words.cnt"/>
            <ref name="ph" dita:since="1.3"/>
            <ref name="indexterm"/>
          </choice>
        </zeroOrMore>
      </define>
      <define name="index-see.attributes">
        <optional>
          <attribute name="keyref"/>
        </optional>
        <ref name="univ-atts"/>
      </define>
      <define name="index-see.element">
        <element name="index-see" dita:longName="Index See">
          <a:documentation>An &lt;index-see> element within an &lt;indexterm> directs the
reader to another index entry that the reader should reference instead of the current one.
          </a:documentation>
          <ref name="index-see.attlist"/>
          <ref name="index-see.content"/>
        </element>
      </define>
      <define name="index-see.attlist" combine="interleave">
        <ref name="index-see.attributes"/>
      </define>
    </div>
    <div>
      <a:documentation>LONG NAME: Index See Also</a:documentation>
      <define name="index-see-also.content">
        <zeroOrMore>
          <choice>
            <ref name="words.cnt"/>
            <ref name="ph" dita:since="1.3"/>
            <ref name="indexterm"/>
          </choice>
        </zeroOrMore>
      </define>
      <define name="index-see-also.attributes">
        <optional>
          <attribute name="keyref"/>
        </optional>
        <ref name="univ-atts"/>
      </define>
      <define name="index-see-also.element">
        <element name="index-see-also" dita:longName="Index See Also">
          <a:documentation>An &lt;index-see-also> element ... </a:documentation>
          <ref name="index-see-also.attlist"/>
          <ref name="index-see-also.content"/>
        </element>
      </define>
      <define name="index-see-also.attlist" combine="interleave">
        <ref name="index-see-also.attributes"/>
```

```
      </define>
    </div>
...
      <div>
        <a:documentation>LONG NAME: Index Base</a:documentation>
        <define name="index-base.content">
          <zeroOrMore>
            <choice>
              <ref name="words.cnt"/>
              <ref name="indexterm"/>
            </choice>
          </zeroOrMore>
        </define>
        <define name="index-base.attributes">
          <optional>
            <attribute name="keyref"/>
          </optional>
          <ref name="univ-atts"/>
        </define>
        <define name="index-base.element">
          <element name="index-base" dita:longName="Index Base">
            <a:documentation>The &lt;index-base> element allows indexing extensions to be
added by specializing off this element. It does not in itself have any meaning and should be
ignored in
              processing. Category: Miscellaneous elements</a:documentation>
            <ref name="index-base.attlist"/>
            <ref name="index-base.content"/>
          </element>
        </define>
        <define name="index-base.attlist" combine="interleave">
          <ref name="index-base.attributes"/>
        </define>
_
      </div>
...
  <div>
    <a:documentation>SPECIALIZATION ATTRIBUTE DECLARATIONS</a:documentation>
...
    <define name="index-base.attlist" combine="interleave">
      <ref name="global-atts"/>
      <optional>
        <attribute name="class" a:defaultValue="- topic/index-base "/>
      </optional>
    </define>

        <define name="index-see.attlist" combine="interleave">
      <ref name="global-atts"/>
      <optional>
        <attribute name="class" a:defaultValue="- topic/index-see "/>
      </optional>
    </define>
    <define name="index-see-also.attlist" combine="interleave">
      <ref name="global-atts"/>
      <optional>
        <attribute name="class" a:defaultValue="- topic/index-see-also "/>
      </optional>
    </define>
```

Modified terminology

None

Modified specification documentation

Note All numeric references here are to the DITA 2.0 specification, working draft 04.

The following topic cluster needs to be modified and relocated:

- 9.5.4 Indexing-group domain elements

- 9.5.4.1 <indexterm>
- 9.5.4.2 <index-see>
- 9.5.4.3 <index-see-also>
- ~~9.5.4.4 <index-sort-as>~~

The topic cluster should be re-named "Indexing elements"; the <index-sort-as> topic removed; and the cluster relocated to be a child of "9.2 "Body elements".

The cross reference to the <index-sort-as> topic should be removed from "9.1 DITA elements, A to Z."

The relationship-table link from 6.8 "Sorting" to 9.5.44.4 <index-sort-as> should be removed.

Any index entries to <index-sort-as> need to be removed.

Content from 9.5.44.4 <index-sort-as> is either incorporated into a new Index sorting (215) topic or moved into 9.5.6.5 <sort-as>.

The following table contains precise suggestions for changes to be made to other topics. The following conventions are used to indicate textual changes:

- Deletions are indicated with line through and red text, for example, ~~deletion~~.
- Insertions are indicated with underlining and green text, for example, insertion.

Topic	DITA 2.0, working draft 04 content	New content
6 DITA processing	DITA processing is affected by a number of factors, including attributes that indicate the set of vocabulary and constraint modules on which a DITA document depends; navigation; linking; content reuse (using direct or indirect addressing); conditional processing; branch filtering; chunking; and more. In addition, translation of DITA content is expedited through the use of the @dir, @translate, and @xml:lang attributes, and the <index-sort-as> element.	DITA processing is affected by a number of factors, including attributes that indicate the set of vocabulary and constraint modules on which a DITA document depends; navigation; linking; content reuse (using direct or indirect addressing); conditional processing; branch filtering; chunking; and more. In addition, translation of DITA content is expedited through the use of the @dir, @translate, and @xml:lang attributes~~, and the <index-sort-as> element~~.
6.6 Translation and localization	DITA has features that facilitate preparing content for translation and working with multilingual content, including the @xml:lang attribute, the @dir attribute, and the @translate attribute. In addition, the <sort-as> and <index-sort-as> elements provide support for sorting in languages in which the correct sorting of an element requires text that is different from the base content of the element.	DITA has features that facilitate preparing content for translation and working with multilingual content, including the @xml:lang attribute, the @dir attribute, and the @translate attribute. In addition, the <sort-as>~~ and <index-sort-as> elements provide~~ element provides support for sorting in languages in which the correct sorting of an element requires text that is different from the base content of the element.
6.8 Sorting	See the Sorting (215) topic below.	See the modified Sorting (215) topic below. I want TC members to view

Topic	DITA 2.0, working draft 04 content	New content
		the changes within the context of the entire topic.
9.5.6.5 <sort-as>	See the sort-as (216) topic below.	See the modified sort-as (216) topic below. I want TC members to view the changes within the context of the entire topic.

Migration plans for backwards incompatibilities

See the "Migration plan" section of the stage two proposal.

Index sorting

~~The <index-sort-as> element~~ The combination of an <indexterm> and <sort-as> element specifies a sort phrase under which an index entry ~~would be~~ is sorted.

This ~~element~~ gives an author the flexibility to sort an index entry in an index differently from how its text normally would be sorted. The common use for this is to disregard insignificant leading text, such as punctuation or words like "the" or "a". For example, the author might want <data> to be sorted under the letter D rather than the left angle bracket (<). An author might want to include such an entry under both the punctuation heading and the letter D, in which case there can be two index entry directives differentiated only by the sort order.

Comment by Kristen J Eberlein on 18 July 2019

Comment from Dawn Stevens during the stage three review of proposal #253, "Indexing changes":

“So .. an author might want to sort <data> under both D and <. If that was the case, they would need two separate indexterm elements, each with its own <sort-as>?”

I think we need to clarify the above paragraph, but I think it will be better done as part of a targeted review of all spec content about indexing, rather than as part of the review for this proposal.

Certain languages have special sort order needs. For example, Japanese index entries might be written partially or wholly in kanji, but need to be sorted in phonetic order according to its hiragana/katakana rendition. There is no reliable automated way to map written to phonetic text: for kanji text, there can be multiple phonetic possibilities depending on the context. The only way to correctly sort Japanese index entries is to keep the phonetic counterparts with the written forms. The phonetic text would be presented as the sort order text for indexing purposes.

Sorting

Processors can be configured to sort elements. Typical processing includes sorting glossary entries, index entries, lists of parameters or reference entries in custom navigation structures, and tables based on the contents of cells in specific columns or rows.

Each element to be sorted must have some inherent text on which it will be sorted. This text is the *base sort phrase* for the element. For elements that have titles, the base sort phrase usually is the content of the <title> element. For elements that do not have titles, the base sort phrase might be literal content in the DITA source, or it might be generated or constructed based on the semantics of the element involved; for example, it could be constructed from various attribute or metadata values.

Processors that perform sorting **SHOULD** explicitly document how the base sort phrase is determined for a given element.

The <sort-as> element can be used to specify an effective sort phrase when the base sort phrase is not appropriate for sorting. For index terms, the ~~<index-sort-as>~~ <sort-as> element ~~can be used to specify~~ specifies the effective sort phrase for an index entry.

The details of sorting and grouping are implementation specific. Processors might provide different mechanisms for defining or configuring collation and grouping details. Even where the <sort-as> element is specified, two processors might produce different sorted and grouped results because they might use different collation and grouping rules. For example, one processor might be configured to sort English terms before non-English terms, while another might be configured to sort them after. The grouping and sorting of content is subject to local editorial rules.

When a <sort-as> element is specified, processors that sort the containing element **MUST** construct the effective sort phrase by prepending the content of the <sort-as> element to the base sort phrase. This ensures that two items with the same <sort-as> element but different base sort phrases will sort in the appropriate order.

For example, if a processor uses the content of the <title> element as the base sort phrase, and 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 phrase would be "twenty-four hour24 Hour Support Hotline".

<sort-as>

For elements that are sorted, the <sort-as> element provides text that is combined with the base sort phrase to construct the effective sort phrase. The text can be specified in the content of the <sort-as> element or in the value attribute on the <sort-as> element. The <sort-as element> element also is useful for elements where the base sort phrase is inadequate or non-existent, for example, a glossary or index entry for a Japanese Kanji phrase.

Usage information

The <sort-as> element can contain <text> and <keyword> elements in order to enable content referencing. If a <keyword> element is used within <sort-as>, the @keyref attribute can be used to set the sort phrase. If a <keyword> uses @keyref and would otherwise also act as a navigation link, the link aspect of the @keyref attribute is ignored.

Some elements in the base DITA vocabulary are natural candidates for sorting, including topics, definition list entries, index entries, and rows in tables and simple tables. Authors are likely to include <sort-as> elements in the following locations:

- For topics, the <sort-as> element can be included directly in <title>, <searchtitle>, or <navtitle> when the different forms of title need different effective sort phrases. If the effective sort phrase is common to all the titles for a topic, the <sort-as> element can be included as a direct child of ~~the topic prolog anywhere <data> is allowed~~ the <prolog> element in the topic.
- For glossary entry topics, the <sort-as> element can be included directly in <glossterm> or as a direct child of the ~~<prolog>~~ <prolog> element.
- For topic references, the <sort-as> element can be included directly in the <navtitle> or <title> element within <topicmeta> or as a child of <topicmeta>.
- For definition list items, ~~include~~ the <sort-as> element can be included in the <dt> element.
- For index entries, the <sort-as> can be included as a child of <indexterm>. In a multilevel index entry, the <sort-as> element only affects the level in which it occurs.

Processing expectations

As a specialization of <data>, the <sort-as> element is allowed in any context where <data> is allowed. However, the presence of <sort-as> within an element does not, by itself, indicate that the containing element should be sorted. Processors can choose to sort any DITA elements for any reason. Likewise, processors are not required to sort any elements. See [sorting topic] for more information on sorting.

Processors **SHOULD** expect to encounter <sort-as> elements in the above locations. Processors that sort **SHOULD** use the following precedence rules:

- A <sort-as> element that is specified in a title takes precedence over a <sort-as> element that is specified as a child of the topic prolog.
- Except for instances in the topic prolog, processors only apply <sort-as> elements that are either a direct child of the element to be sorted or a direct child of the title- or label-defining element of the element to be sorted.
- When an element contains multiple, direct-child, <sort-as> elements, the first direct-child <sort-as> element in document order takes precedence.
- ~~When located within the <indexterm> element, the <sort-as> element is equivalent to <index-sort-as>. It is an error for an <indexterm> element to directly contain both <sort-as> and <index-sort-as> elements.~~
- It is an error if there is more than one <sort-as> child for a given <indexterm>. An implementation might give an error message.
- Sort phrases are determined after filtering and content reference resolution occur.

When a <sort-as> element is specified, processors that sort the containing element **MUST** construct the effective sort phrase by prepending the content of the <sort-as> element to the base sort phrase. This ensures that two items with the same <sort-as> element but different base sort phrases will sort in the appropriate order.

For example, if a processor uses the content of the <title> element as the base sort phrase, and 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 phrase would be "twenty-four hour24 Hour Support Hotline".

Specialization hierarchy

The <sort-as> element is specialized from <data>. It in defined in the utilities-domain module.

Attributes

The following attributes are available on this element: Universal attribute group and the attributes defined below.

@name
: Names the metadata item that the element represents. The default value is "sort-as". Specializations of <sort-as> can set the default value of the @name attribute to reflect the tag name of the specialized element.

@value
: The value of the metadata item. When the <sort-as> element has content and the @value attribute is specified, the @value attribute takes precedence. If the @value attribute is not specified and the <sort-as> element does not contain content, then the <sort-as> element has no effect.

Examples

Intro goes here ...

Comment by Kristen J Eberlein on 16 June 2019

I have not attempted to make changes in this section with bold and line-through. The examples have not changed since DITA 1.3, although the markup for this example section has been modified.

Figure 88: Sorting glossary entries

The following code samples show how a glossary entry for the Chinese ideographic character for "big" might specify an effective sort phrase of "dada" (the Pin-Yin transliteration for Mandarin):

The `<sort-as>` element can be located within `<glossterm>`:

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

Or the `<sort-as>` element can be located within `<prolog>`:

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

Figure 89: Sorting index entries

This following code sample shows an index entry for `<data>` that will be sorted as "data":

```
<indexterm>&lt;data&gt;<sort-as>data</sort-as></indexterm>
```

3.1.10 Stage 3: #277 Change specialization base for <imagemap>

Change the specialization base for `<imagemap>` to enable image maps to be treated as images.

Champion

Kristen James Eberlein, Eberlein Consulting LLC

Tracking information

Event	Date	Links
Stage 1 proposal accepted	16 July 2019	Minutes, 16 July 2019
Stage 2 proposal submitted	17 July 2019	DITA PDF
Stage 2 proposal discussed	06 August 2019	Minutes, 06 August 2019
Stage 2 proposal approved	13 August 2019	Minutes, 13 August 2019
Stage 3 proposal submitted to reviewers	13 August 2019	Scott Hudson

Event	Date	Links
		Eliot Kimber Zoe Lawson
Stage 3 proposal (this document) submitted to TC	18 August 2019; updated 20 August with link to minutes when stage 2 proposal approved	

Approved technical requirements

Change the specialization base of `<imagemap>` and `<area>` to `<div>`

Dependencies or interrelated proposals

None

Modified grammar files

The following files must be modified:

DTDs

- `basemap.dtd`
- `basetopic.dtd`
- `utilitiesDomain.ent`
- `utilitiesDomain.mod`

RNG

`utilitiesDomain.rng`

In the content below, the following conventions are used:

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

Figure 90: Changes to basemap.dtd

```
...
<!-- ============================================================= -->
<!--                    DOMAIN EXTENSIONS                          -->
<!-- ============================================================= -->
<!--                    One for each extended base element, with
                        the name of the domain(s) in which the
                        extension was declared                     -->

<!ENTITY % topicref     "topicref |
                         %mapgroup-d-topicref; |
                         %ditavalref-d-topicref;
                        ">
<!ENTITY % keywords     "keywords |
                         %delay-d-keywords;
                        ">
<!ENTITY % index-base   "index-base |
                         %indexing-d-index-base;
                        ">
<!ENTITY % note         "note |
                         %hazard-d-note;
                        ">
<!ENTITY % ph           "ph |
```

```
                          %hi-d-ph;
                         ">
<!ENTITY % fig           "fig |
                          %ut-d-fig;
                         ">
<!ENTITY % data          "data |
                          %ut-d-data;
                         ">
<!ENTITY % div           "div |
                          %ut-d-div;
                         ">
...
```

Figure 91: Changes to basetopic.dtd

```
...
<!-- ============================================================= -->
<!--                    DOMAIN EXTENSIONS                          -->
<!-- ============================================================= -->
<!--                    One for each extended base element, with
                        the name of the domain(s) in which the
                        extension was declared                     -->

<!ENTITY % index-base    "index-base |
                          %indexing-d-index-base;
                         ">
<!ENTITY % note          "note |
                          %hazard-d-note;
                         ">
<!ENTITY % ph            "ph |
                          %hi-d-ph;
                         ">
<!ENTITY % fig           "fig |
                          %ut-d-fig;
                         ">
<!ENTITY % data          "data |
                          %ut-d-data;
                         ">
<!ENTITY % div           "div |
                          %ut-d-div;
                         ">
...
```

Figure 92: Changes to utilitiesDomain.ent

```
...<!-- ============================================================= -->
<!--                    ELEMENT EXTENSION ENTITY DECLARATIONS      -->
<!-- ============================================================= -->

<!ENTITY % ut-d-fig
  "imagemap"
>
<!ENTITY % ut-d-div
  "imagemap"
>
<!ENTITY % ut-d-data
  "sort-as"
>
...
```

Figure 93: Changes to utilitiesDomain.mod

```
...
<!-- ============================================================= -->
<!--              SPECIALIZATION ATTRIBUTE DECLARATIONS            -->
<!-- ============================================================= -->

<!ATTLIST  imagemap      %global-atts;  class CDATA "+ topic/fig ut-d/imagemap ">
<!ATTLIST  imagemap      %global-atts;  class CDATA "+ topic/div ut-d/imagemap ">
```

```
<!ATTLIST  area          %global-atts;  class CDATA "+ topic/figgroup ut-d/area ">
<!ATTLIST  area          %global-atts;  class CDATA "+ topic/div ut-d/area ">
<!ATTLIST  shape         %global-atts;  class CDATA "+ topic/keyword ut-d/shape ">
<!ATTLIST  coords        %global-atts;  class CDATA "+ topic/ph ut-d/coords ">
<!ATTLIST  sort-as       %global-atts;  class CDATA "+ topic/data ut-d/sort-as ">

<!-- ================== End of DITA Utilities Domain ==================== -->
```

Figure 94: Changes to utiltiesDomain.rng

```
...
  <div>
    <a:documentation>Define domain extension patterns</a:documentation>
    <define name="ut-d-fig">
      <ref name="imagemap.element"/>
    </define>
    <define name="fig" combine="choice">
      <ref name="ut-d-fig"/>
    </define>
    <define name="ut-d-div">
      <ref name="imagemap.element"/>
    </define>
    <define name="div" combine="choice">
      <ref name="ut-d-div"/>
    </define>
    <define name="ut-d-data">
      <ref name="sort-as.element"/>
    </define>
    <define combine="choice" name="data">
      <ref name="ut-d-data"/>
    </define>
  </div>
...
  <div>
    <a:documentation>SPECIALIZATION ATTRIBUTE DECLARATIONS</a:documentation>

    <define combine="interleave" name="imagemap.attlist">
      <ref name="global-atts"/>
      <optional>
        <attribute a:defaultValue="+ topic/fig ut-d/imagemap " name="class"/>
        <attribute a:defaultValue="+ topic/div ut-d/imagemap " name="class"/>
      </optional>
    </define>
    <define combine="interleave" name="area.attlist">
      <ref name="global-atts"/>
      <optional>
        <attribute a:defaultValue="+ topic/figgroup ut-d/area " name="class"/>
        <attribute a:defaultValue="+ topic/div ut-d/area " name="class"/>
      </optional>
    </define>
    <define combine="interleave" name="shape.attlist">
      <ref name="global-atts"/>
      <optional>
        <attribute a:defaultValue="+ topic/keyword ut-d/shape " name="class"/>
      </optional>
    </define>
    <define combine="interleave" name="coords.attlist">
      <ref name="global-atts"/>
      <optional>
        <attribute a:defaultValue="+ topic/ph ut-d/coords " name="class"/>
      </optional>
    </define>
    <define combine="interleave" name="sort-as.attlist">
      <ref name="global-atts"/>
      <optional>
        <attribute a:defaultValue="+ topic/data ut-d/sort-as " name="class"/>
      </optional>
    </define>

  </div>
...
```

Modified terminology

None

Modified specification documentation

The following table lists the changes that need to be made to element reference topics.

Topic	Current GitHub source	Post-accepted proposal DITA source
`<area>`	The `<area>` element is specialized from `<figgroup>`. It in defined in the utilities-domain module.	The `<area>` element is specialized from `<div>`. It is defined in the utilities-domain module.
`<imagemap>`	The `<imagemap>` element is specialized from `<fig>`. It in defined in the utilities-domain module.	The `<imagemap>` element is specialized from `<div>`. It is defined in the utilities-domain module.

Migration plans for backwards incompatibilities

See the "Migration plan" section of the stage two proposal.

3.2 Technical Content edition

3.2.1 Stage 3: #85 Add @sub and <sup> to glossary elements

The `<sub>` and `<sup>` elements are not allowed in several elements from glossentry, which means that terms cannot be used properly in certain contexts. Since `<sub>` and `<sup>` are specialized from `<ph>`, adding the `<ph>` element solves the issue.

Champion

Scott Hudson

Tracking information

Event	Date	Links
Stage 1 proposal accepted	24 Oct 2017	https://lists.oasis-open.org/archives/dita/201711/msg00009.html
Stage 2 proposal submitted	5 Feb 2018	https://markmail.org/message/pk43srtwslyj3trd?q=Issue+85+list:org%2Eoasis-open%2Elists%2Edita
Stage 2 proposal discussed	13 Feb 2018	https://lists.oasis-open.org/archives/dita/201802/msg00050.html
Stage 2 proposal approved	20 Feb 2018	https://lists.oasis-open.org/archives/dita/201802/msg00071.html
Stage 3 proposal submitted to reviewers	22 May 2018	Bill Burns, Nancy Harrison
Stage 3 proposal (this document) submitted to TC		

Approved technical requirements

Modify the content models to allow the <ph> element within the following elements:

- <glossSurfaceForm>
- <glossAcronym>
- <glossSynonym>
- <glossShortForm>
- <glossAbbreviation>

Dependencies or interrelated proposals

None.

Modified grammar files

Renaming or refactoring elements and attributes

Content model name / location	Original model	Proposed model
glossSurfaceForm.content (glossentry.mod)	<!ENTITY % glossSurfaceForm.content "(#PCDATA | %keyword; | %term; | %text; | %tm;)*" >	<!ENTITY % glossSurfaceForm.content "(#PCDATA | %keyword; | %term; | %text; | %tm; | %ph;)*" >
glossSurfaceForm.content (glossentryMod.rng)	<define name="glossSurfaceForm.content"> <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> </choice> </zeroOrMore> </define>	<define name="glossSurfaceForm.content"> <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> <ref name="ph" dita:since="2.0" /> </choice> </zeroOrMore> </define>
glossAcronym.content (glossentry.mod)	<!ENTITY % glossAcronym.content "(#PCDATA |	<!ENTITY % glossAcronym.content "(#PCDATA |

Content model name / location	Original model	Proposed model
	%keyword; | %term; | %text; | %tm;)*" >	%keyword; | %term; | %text; | %tm; | %ph;)*" >
glossAcronym.content (glossentryMod.rng)	<define name="glossAcronym.content"> > <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> </choice> </zeroOrMore> </define>	<define name="glossAcronym.content"> > <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> <ref name="ph" dita:since="2.0""/> </choice> </zeroOrMore> </define>
glossSynonym.content (glossentry.mod)	<!ENTITY % glossSynonym.content "(#PCDATA | %keyword; | %term; | %text; | %tm;)*" >	<!ENTITY % glossSynonym.content "(#PCDATA | %keyword; | %term; | %text; | %tm; | %ph;)*" >
glossSynonym.content (glossentryMod.rng)	<define name="glossSynonym.content"> > <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> </choice> </zeroOrMore> </define>	<define name="glossSynonym.content"> > <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> <ref name="ph" dita:since="2.0" /> </choice> </zeroOrMore> </define>
glossShortForm.content (glossentry.mod)	<!ENTITY % glossShortForm.content	<!ENTITY % glossShortForm.content "(#PCDATA |

Content model name / location	Original model	Proposed model
	"(#PCDATA | %keyword; | %term; | %text; | %tm;)*" >	%keyword; | %term; | %text; | %tm; | %ph;)*" >
glossShortForm.content (glossentryMod.rng)	<define name="glossShortForm.content"> <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> </choice> </zeroOrMore> </define>	<define name="glossShortForm.content"> <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> <ref name="ph" dita:since="2.0" /> </choice> </zeroOrMore> </define>
glossAbbreviation.content (glossentry.mod)	<!ENTITY % glossAbbreviation.content "(#PCDATA | %keyword; | %term; | %text; | %tm;)*" >	<!ENTITY % glossAbbreviation.content "(#PCDATA | %keyword; | %term; | %text; | %tm; | %ph;)*" >
glossAbbreviation.content (glossentryMod.rng)	<define name="glossAbbreviation.content"> <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> </choice> </zeroOrMore> </define>	<define name="glossAbbreviation.content"> <zeroOrMore> <choice> <text/> <ref name="keyword"/> <ref name="term"/> <ref name="text" dita:since="1.3"/> <ref name="tm"/> <ref name="ph" dita:since="2.0" /> </choice> </zeroOrMore> </define>

Modified terminology

None.

Modified specification documentation

The automatically generated content model descriptions include the <ph> for the affected content models:

- <glossSurfaceForm>
- <glossAcronym>
- <glossSynonym>
- <glossShortForm>
- <glossAbbreviation>

No further topic changes are expected.

Migration plans for backward incompatibilities

There are no backward compatibility issues, since the <ph> element is an addition to existing content models.

No existing content needs to be migrated, since the <ph> element is an addition to existing content models.

Existing constraints or specializations might need to be adjusted or updated to remove unwanted elements from the glossary content models. One benefit, is that any elements or domains constrained in the shell DTDs (particularly from the equation, highlight, programming, software, user interface, and XML mention domains) are not allowed in the glossary models of the constrained shell.

All @domains- and @class-based processing of specialized elements and attributes continue to function exactly as before. Processors might need to be adjusted to account for the allowed children and specializations of <ph>, such as <xref>.

3.2.2 Stage 3: #106 Nest steps

Allow the <steps> element to nest, in order to address one of our more common authoring pain points, simplify reuse, and reduce overall element count.

Champion

Robert D. Anderson

Tracking information

Event	Date	Links
Stage 1 proposal accepted	13 March 2018	https://lists.oasis-open.org/archives/dita/201803/msg00052.html
Stage 2 proposal submitted	19 March 2018	HTML, DITA
Stage 2 proposal discussed	20 March 2018	Meeting minutes
Stage 2 proposal approved	27 March 2018	Meeting minutes
Stage 3 proposal submitted to reviewers	17 April 2018	Nancy Harrison, Bob Thomas, Carsten Brennecke
Stage 3 proposal (this document) submitted to TC	4 September 2018	DITA version in SVN

Approved technical requirements

1. Delete the <substeps> and <substep> elements.
2. Add <steps> and <steps-unordered> to the content model of <step>, after the command (where <substeps> is currently allowed)

Dependencies or interrelated proposals

N/A

Modified grammar files

task.mod (before)

```
<!ENTITY % substeps
"substeps"
  >
<!ENTITY % substep
"substep"
  >
```

task.mod (after)

```
<!ENTITY % substeps
"substeps"
-->
<!ENTITY % substep
"substep"
-->
```

task.mod (before)

```
<!ENTITY % step.content
                      "((%note;)*,
                        (%cmd;),
                        (%choices; |
                         %choicetable; |
                         %info; |
                         %itemgroup; |
                         %stepxmp; |
                         %substeps; |
                         %tutorialinfo;)*,
                        (%stepresult;)?,

(%steptroubleshooting;)?)"
>
```

task.mod (after)

```
<!ENTITY % step.content
                      "((%note;)*,
                        (%cmd;),
                        (%choices; |
                         %choicetable; |
                         %info; |
                         %itemgroup; |
                         %stepxmp; |
                         %substeps; |
                         %steps; |
                         %steps-unordered;
|
                         %tutorialinfo;)*,
                        (%stepresult;)?,

(%steptroubleshooting;)?)"
>
```

task.mod (before)

```
<!--                    LONG NAME: Sub-
steps                      -->
<!ENTITY % substeps.content
                      "((%data; |
                         %data-about;)*,
                        (%substep;)+)"
>
<!ENTITY % substeps.attributes
             "%univ-atts;
              outputclass
                          CDATA

#IMPLIED"
>
<!ELEMENT  substeps %substeps.content;>
<!ATTLIST  substeps %substeps.attributes;>


<!--                    LONG NAME: Sub-
step                       -->
<!ENTITY % substep.content
                      "((%note;)*,
                        (%cmd;),
                        (%info; |
                         %itemgroup; |
```

task.mod (after)

```
<!--                    LONG NAME: Sub-
steps                      -->
<!ENTITY % substeps.content
                      "((%data; |
                         %data-about;)*,
                        (%substep;)+)"
>
<!ENTITY % substeps.attributes
             "%univ-atts;
              outputclass
                          CDATA

#IMPLIED"
>
<!ELEMENT  substeps %substeps.content;>
<!ATTLIST  substeps %substeps.attributes;>


<!--                    LONG NAME: Sub-
step                       -->
<!ENTITY % substep.content
                      "((%note;)*,
                        (%cmd;),
                        (%info; |
                         %itemgroup; |
```

task.mod (before)

```
                          %stepxmp; |
                          %tutorialinfo;)*,
                         (%stepresult;)?)"
>
<!ENTITY % substep.attributes
             "importance
                          (optional |
                           required |
                           -dita-use-conref-
target)
                                        #IMPLIED
              %univ-atts-no-importance-
task;
              outputclass
                          CDATA

#IMPLIED"
>
<!ELEMENT  substep %substep.content;>
<!ATTLIST  substep %substep.attributes;>
```

task.mod (after)

```
                          %stepxmp; |
                          %tutorialinfo;)*,
                         (%stepresult;)?)"
>
<!ENTITY % substep.attributes
             "importance
                          (optional |
                           required |
                           -dita-use-conref-
target)
                                        #IMPLIED
              %univ-atts-no-importance-
task;
              outputclass
                          CDATA

#IMPLIED"
>
<!ELEMENT  substep %substep.content;>
<!ATTLIST  substep %substep.attributes;>
```

```
<!ATTLIST  substeps     %global-atts;
class CDATA "- topic/ol task/substeps ">
<!ATTLIST  substep      %global-atts;
class CDATA "- topic/li task/substep ">
```

```
<!ATTLIST  substeps     %global-atts;
class CDATA "- topic/ol task/substeps ">
<!ATTLIST  substep      %global-atts;
class CDATA "- topic/li task/substep ">
```

taskMod.rng (before)

```
<define name="substeps">
      <ref name="substeps.element"/>
    </define>
    <define name="substep">
      <ref name="substep.element"/>
    </define>
```

taskMod.rng (after)

```
<define name="substeps">
      <ref name="substeps.element"/>
    </define>
    <define name="substep">
      <ref name="substep.element"/>
    </define>
```

```
<define name="step.content">
        <zeroOrMore>
          <ref name="note"/>
        </zeroOrMore>
        <ref name="cmd"/>
        <zeroOrMore>
          <choice>
            <ref name="choices"/>
            <ref name="choicetable"/>
            <ref name="info"/>
            <ref name="itemgroup"/>
            <ref name="stepxmp"/>
            <ref name="substeps"/>
            <ref name="tutorialinfo"/>
          </choice>
        </zeroOrMore>
```

```
<define name="step.content">
        <zeroOrMore>
          <ref name="note"/>
        </zeroOrMore>
        <ref name="cmd"/>
        <zeroOrMore>
          <choice>
            <ref name="choices"/>
            <ref name="choicetable"/>
            <ref name="info"/>
            <ref name="itemgroup"/>
            <ref name="stepxmp"/>
            <ref name="substeps"/>
            <ref name="steps"/>
            <ref name="steps-unordered"/>
            <ref name="tutorialinfo"/>
          </choice>
        </zeroOrMore>
```

```
    <div>
      <a:documentation> LONG NAME: Sub-
steps </a:documentation>
      <define name="substeps.content">
        <zeroOrMore dita:since="1.3">
          <choice>
            <ref name="data"/>
            <ref name="data-about"/>
          </choice>
```

```
    <div>
      <a:documentation> LONG NAME: Sub-
steps </a:documentation>
      <define name="substeps.content">
        <zeroOrMore dita:since="1.3">
          <choice>
            <ref name="data"/>
            <ref name="data-about"/>
          </choice>
```

taskMod.rng (before)

```
        </zeroOrMore>
        <oneOrMore>
          <ref name="substep"/>
        </oneOrMore>
      </define>
      <define name="substeps.attributes">
        <ref name="univ-atts"/>
        <optional>
          <attribute name="outputclass"/>
        </optional>
      </define>
      <define name="substeps.element">
        <element name="substeps"
dita:longName="Sub-steps">
          <a:documentation>
            <![CDATA[The <substeps> element
allows you to break a step down
into a series of separate actions, and
should be used only if necessary.
Try to describe the steps of a task in a
single level of steps.
If you need to use more than one level of
substep nesting, you should
probably rewrite the task to simplify it.
    Category: Task elements
  ]]></a:documentation>
          <ref name="substeps.attlist"/>
          <ref name="substeps.content"/>
        </element>
      </define>
      <define name="substeps.attlist"
combine="interleave">
        <ref name="substeps.attributes"/>
      </define>
    </div>
    <div>
      <a:documentation> LONG NAME: Sub-step
</a:documentation>
      <define name="substep.content">
        <zeroOrMore>
          <ref name="note"/>
        </zeroOrMore>
        <ref name="cmd"/>
        <zeroOrMore>
          <choice>
            <ref name="info"/>
            <ref name="itemgroup"/>
            <ref name="stepxmp"/>
            <ref name="tutorialinfo"/>
          </choice>
        </zeroOrMore>
        <optional>
          <ref name="stepresult"/>
        </optional>
      </define>
      <define name="substep.attributes">
        <optional>
          <attribute name="importance">
            <choice>
              <value>optional</value>
              <value>required</value>
              <value>-dita-use-conref-
target</value>
            </choice>
          </attribute>
        </optional>
        <ref name="univ-atts-no-importance-
task"/>
        <optional>
          <attribute name="outputclass"/>
        </optional>
```

taskMod.rng (after)

```
        </zeroOrMore>
        <oneOrMore>
          <ref name="substep"/>
        </oneOrMore>
      </define>
      <define name="substeps.attributes">
        <ref name="univ-atts"/>
        <optional>
          <attribute name="outputclass"/>
        </optional>
      </define>
      <define name="substeps.element">
        <element name="substeps"
dita:longName="Sub-steps">
          <a:documentation>
            <![CDATA[The <substeps> element
allows you to break a step down
into a series of separate actions, and
should be used only if necessary.
Try to describe the steps of a task in a
single level of steps.
If you need to use more than one level of
substep nesting, you should
probably rewrite the task to simplify it.
    Category: Task elements
  ]]></a:documentation>
          <ref name="substeps.attlist"/>
          <ref name="substeps.content"/>
        </element>
      </define>
      <define name="substeps.attlist"
combine="interleave">
        <ref name="substeps.attributes"/>
      </define>
    </div>
    <div>
      <a:documentation> LONG NAME: Sub-step
</a:documentation>
      <define name="substep.content">
        <zeroOrMore>
          <ref name="note"/>
        </zeroOrMore>
        <ref name="cmd"/>
        <zeroOrMore>
          <choice>
            <ref name="info"/>
            <ref name="itemgroup"/>
            <ref name="stepxmp"/>
            <ref name="tutorialinfo"/>
          </choice>
        </zeroOrMore>
        <optional>
          <ref name="stepresult"/>
        </optional>
      </define>
      <define name="substep.attributes">
        <optional>
          <attribute name="importance">
            <choice>
              <value>optional</value>
              <value>required</value>
              <value>-dita-use-conref-
target</value>
            </choice>
          </attribute>
        </optional>
        <ref name="univ-atts-no-importance-
task"/>
        <optional>
          <attribute name="outputclass"/>
        </optional>
```

taskMod.rng (before)	**taskMod.rng (after)**
</define> <define name="substep.element"> <element name="substep" dita:longName="Sub-step"> <a:documentation> <![CDATA[A <substep> element has the same structure as a <step>, except that it does not allow lists of choices or substeps within it, in order to prevent unlimited nesting of steps. Category: Task elements]]></a:documentation> <ref name="substep.attlist"/> <ref name="substep.content"/> </element> </define> <define name="substep.attlist" combine="interleave"> <ref name="substep.attributes"/> </define> 	~~</define>~~ ~~<define name="substep.element">~~ ~~<element name="substep"~~ ~~dita:longName="Sub-step">~~ ~~<a:documentation>~~ ~~<![CDATA[A <substep> element~~ ~~has the same structure as a <step>,~~ ~~except that it does not allow lists of~~ ~~choices or substeps within it,~~ ~~in order to prevent unlimited nesting of~~ ~~steps.~~ ~~Category: Task elements~~ ~~]]></a:documentation>~~ ~~<ref name="substep.attlist"/>~~ ~~<ref name="substep.content"/>~~ ~~</element>~~ ~~</define>~~ ~~<define name="substep.attlist"~~ ~~combine="interleave">~~ ~~<ref name="substep.attributes"/>~~ ~~</define>~~ ~~ ~~
<define name="substeps.attlist" combine="interleave"> <ref name="global-atts"/> <optional> <attribute name="class" a:defaultValue="- topic/ol task/substeps "/> </optional> </define> <define name="substep.attlist" combine="interleave"> <ref name="global-atts"/> <optional> <attribute name="class" a:defaultValue="- topic/li task/substep "/> </optional> </define>	~~<define name="substeps.attlist"~~ ~~combine="interleave">~~ ~~<ref name="global-atts"/>~~ ~~<optional>~~ ~~<attribute name="class"~~ ~~a:defaultValue="- topic/ol task/substeps "/>~~ ~~</optional>~~ ~~</define>~~ ~~<define name="substep.attlist"~~ ~~combine="interleave">~~ ~~<ref name="global-atts"/>~~ ~~<optional>~~ ~~<attribute name="class"~~ ~~a:defaultValue="- topic/li task/substep "/>~~ ~~</optional>~~ ~~</define>~~

Modified terminology

N/A

Modified specification documentation

Content models will not be published in the same document as the specification with DITA 2.0, so there are no changes required based on the DITA 1.3 content model appendices.

Existing element reference topics for `<substeps>` and `<substep>` will be deleted.

The task table in topic B.6 Element-by-element recommendations for translators: Technical content edition needs to delete the rows for `<substep>` and `<substeps>`.

In addition, the following changes are required:

<step> element reference topic (before)	**<step> element reference topic (after)**
Definition of `@importance`: **importance** Describes whether the current `<step>` or `<substep>` is optional or required. Output	Definition of `@importance`: **importance** Describes whether the current `<step>` ~~or~~ ~~`<substep>`~~ is optional or required. Output

<step> element reference topic (before)	**<step> element reference topic (after)**
processors might highlight steps that are optional or required. Allowed values are "optional", "required", or -dita-use-conref-target.	processors might highlight steps that are optional or required. Allowed values are "optional", "required", or -dita-use-conref-target.

Task topic (strict task) (before)	**Task topic (strict task) (after)**
From topic 2.7.1.4 in DITA 1.3, the description of <steps>: The <step> element represents an action that a user must follow to accomplish a task. Each <step> in a <task> must contain a command <cmd> element which describes the particular action the user must perform to accomplish the overall task. The <step> element can also contain information <info>, substeps <substeps>, tutorial information <tutorialinfo>, a step example <stepxmp>, choices <choices>, a step result <stepresult> , or troubleshooting <steptroubleshooting> , although these are optional.	The <step> element represents an action that a user must follow to accomplish a task. Each <step> in a <task> must contain a command <cmd> element which describes the particular action the user must perform to accomplish the overall task. The <step> element can also contain information <info>, ~~substeps <substeps>,~~ tutorial information <tutorialinfo>, a step example <stepxmp>, choices <choices>, a step result <stepresult> , or troubleshooting <steptroubleshooting> , although these are optional.

From topic 2.7.1.4 Task topic (strict task) in DITA 1.3, the description of <steps>:

Before	**After**
The <step> element represents an action that a user must follow to accomplish a task. Each <step> in a <task> must contain a command <cmd> element which describes the particular action the user must perform to accomplish the overall task. The <step> element can also contain information <info>, substeps <substeps>, tutorial information <tutorialinfo>, a step example <stepxmp>, choices <choices>, a step result <stepresult> , or troubleshooting <steptroubleshooting> , although these are optional.	The <step> element represents an action that a user must follow to accomplish a task. Each <step> in a <task> must contain a command <cmd> element which describes the particular action the user must perform to accomplish the overall task. The <step> element can also contain information <info>, ~~substeps <substeps>,~~ tutorial information <tutorialinfo>, a step example <stepxmp>, choices <choices>, a step result <stepresult> , or troubleshooting <steptroubleshooting> , although these are optional.

The following changes are required in TroubleshootingElements.dita

“Troubleshooting information” concept topic (before)	**“Troubleshooting information” concept topic (after)**
The <steptroubleshooting> element can occur following the <cmd> element in the <step> or <substep> element. The <steptroubleshooting>element ends the <step> or <substep> element. Another element, such as <info> or <stepxmp>, cannot be added after the <steptroubleshooting> element.	The <steptroubleshooting> element can occur following the <cmd> element in the <step>~~or <substep>~~ element. The <steptroubleshooting>element ends the <step>~~or <substep>~~ element. Another element, such as <info> or <stepxmp>, cannot be added after the <steptroubleshooting> element.

The following changes are required to the examples in the element reference topics for <choicetable>, <chrow>, <chdesc>, and <chhead>:

Example (before)	Example (after)
<step><cmd>Then this</cmd> <substeps> <substep importance="optional"><cmd>which is done by doing this</cmd></substep> <substep importance="required"><cmd>and then this.</ cmd></substep> </substeps> *[choicetable elided]*	<step><cmd>Then this</cmd> <steps> <step importance="optional"><cmd>which is done by doing this</cmd></step> <step importance="required"><cmd>and then this.</ cmd></step> </steps> *[choicetable elided]*

Migration plans for backwards incompatibilities

1. Because any element allowed in <substep> is already allowed in <step>, content migration is most easily handled with a search/replace tool to change element names.
    - Replace <substep with <step (this will change the start tag for <substeps> to <steps> and also change <substep> to <step>, regardless of what attributes are specified).
    - Replace </substep with </step (this will change the end tag for </substeps> to </steps> and also change </substep> to </step>)
2. Because constraint or specialization modules may use these elements in ways that do not follow normal patterns, it is not possible to handle all possible extensions with a tool or common search/replace expression. The following incompatibilities are possible, though likely to be less common:
    - Specializations of <substeps> and <substep> will need to change ancestry in their @class attributes (replacing the token task/substeps with task/steps and task/substep with task/step)
    - Constraints or specialized elements that explicitly include either <substeps> or <substep> will need to be updated to use <steps> or <step>
3. Rendering tools that automatically add headings to either <steps> or <steps-unordered> may need a conditional test to only render the heading when those elements appear as a child of <taskbody>. Rendering processes can be written in any number of tools or languages, so it is not possible to automate this migration; for tools that need this update, it should require a test such as "If @class of parent element contains task/taskbody " before generating any heading.

4 Revision history

The following table contains information about revisions to this document.

Revision	Date	Editor	Description of changes
01	15 July 2019	Kristen James Eberlein	Generated working draft #01.
02	01 October 2019	Kristen James Eberlein	Generated working draft #02.

Revision	Date	Editor	Description of changes
			Added the following proposals: **Stage two** • #16: Add `<titlealts>` element to map • #21 Resolve inconsistent class values for `<shortdesc>`, `<linktext>`, and `<searchtitle>` • #34 Remove `<topicset>` and `<topicsetref>` • #252 Add `@outputclass` to DITAVAL • #277 Change specialization base for `<imagemap>` • #279 Remove `@lockmeta` attribute **Stage three** • #253: Indexing changes • #277 Change specialization base for `<imagemap>`