Review K: Metadata and cascading metadata
Table of contents

1 DITA metadata...........................................................................................................................................3
  1.1 Metadata elements............................................................................................................................ 3
  1.2 Metadata attributes............................................................................................................................3
  1.3 Metadata in maps and topics.............................................................................................................4
  1.4 Context hooks and window metadata for user assistance.................................................................4

2 Map cascading...........................................................................................................................................6
  2.1 Cascading of metadata attributes in a DITA map..............................................................................6
    2.1.1 Merging of cascading attributes.................................................................................................7
    2.1.2 Processing cascading attributes in a map.................................................................................7
  2.2 Reconciling topic and map metadata elements.................................................................................8
  2.3 Map-to-map cascading behaviors....................................................................................................10
    2.3.1 Cascading of attributes from map to map................................................................................10
    2.3.2 Cascading of metadata elements from map to map.................................................................11
    2.3.3 Cascading of roles from map to map.......................................................................................11
  2.4 Examples of metadata cascading....................................................................................................12
    2.4.1 Example: How metadata elements cascade to topics.............................................................12
    2.4.2 Example: How elements cascade from one map to another...................................................13
    2.4.3 Example: How attributes cascade from one map to another..................................................13
    2.4.4 Example: How @cascade affects attribute cascading.............................................................14
    2.4.5 Example: How <topicref> roles cascade to referenced maps..................................................15

A Aggregated RFC-2119 statements..........................................................................................................16

Index..........................................................................................................................................................18
1 DITA metadata

Metadata can be applied in both DITA topics and DITA maps. Metadata that is assigned in DITA topics can be supplemented or overridden by metadata that is assigned in a DITA map; this design facilitates the reuse of DITA topics in different DITA maps and use-specific contexts.

DITA defines a core set of metadata elements to cover a variety of common scenarios. Because metadata requirements vary so widely, it is expected that few authors will use the full range of default elements. DITA also provides a few generic elements for use when the core elements are not enough.

Requirements for rendering metadata vary widely. For that reason, any rendering of metadata in published content is left up to implementations.

1.1 Metadata elements

The metadata elements are available in topics and DITA maps. This design enables authors and information architects to use identical metadata markup in both topics and maps.

When used in maps, metadata elements are located in the `<topicmeta>` element. When used in topics, metadata elements are located in the `<prolog>` element.

Comment by robander on 6 Feb 2022

In general, specifying metadata in a `<topicmeta>` element is equivalent to specifying it in the `<prolog>` element of a referenced topic. The value of specifying the metadata at the map level is that the topic then can be reused in other maps where different metadata might apply. Many items in the `<topicmeta>` element also cascade to nested `<topicref>` elements within the map.

1.2 Metadata attributes

The metadata attributes specify properties of the content that can be used to determine how the content is processed. Specialized metadata attributes can be defined to enable specific business-processing needs, such as semantic processing and data mining.

Metadata attributes typically are used for the following purposes:

- Filtering content based on the attribute values, for example, to suppress or publish profiled content
- Flagging content based on the attribute values, for example, to highlight specific content on output
- Performing custom processing, for example, to extract business-critical data and store it in a database

The base DITA vocabulary includes five specializations of the @props attribute as domains: @audience, @deliveryTarget, @platform, @product, and @otherprops. These five attributes are included in all map and topic document-type shells provided with the specification.

Metadata attributes fall into a few categories.

Filtering and flagging attributes

The @props attribute and its specializations are intended for filtering. This includes the five specializations added to the OASIS document-type shells: @audience, @deliveryTarget, @platform, @product, and @otherprops. The same attributes plus the @rev attribute are intended for flagging.
Translation and localization attributes
The @dir, @translate, and @xml:lang attributes are intended for use with translating and localizing content.

Architectural attributes
The @class, @DITAArchVersion, and @specializations attributes provide metadata about the DITA source itself, such as what version of the grammar is used. These attributes are not intended for use by authors.

Other metadata attributes
The @status and @importance attributes, as well as custom attributes specialized from @base, are intended for application-specific behavior, such as identifying metadata to aid in search and retrieval.

1.3 Metadata in maps and topics
Information about topics can be specified using elements in the map, using attributes on the <map> or <topicref> elements, or using attributes or elements in the topic itself. By default, metadata in the map supplements or overrides metadata that is specified at the topic level.

Metadata elements and attributes in a map might apply to an individual topic, a set of topics, or globally for the entire document. Most metadata elements authored within a <topicmeta> element associate metadata with the parent element and its children. Because the topics in a branch of the hierarchy typically have some common subjects or properties, this is a convenient mechanism to define metadata for a set of topics.

Comment by rodaande on 8 Feb 2022
We should have a related link from this topic to the section on cascading; this is a conceptual topic about metadata and should not repeat the processing rules, but reading this I immediately want to know *which* elements cascade and how that works.

When the same metadata element or attribute is specified in both a map and a topic, by default the value in the map takes precedence; the assumption here is that the author of the map has more knowledge of the reusing context than the author of the topic.

1.4 Context hooks and window metadata for user assistance
Context hook information specified in the <resourceid> element in the DITA map or in a DITA topic enables processors to generate the header, map, alias and other types of support files that are required to integrate the user assistance with an application. Some user assistance topics might need to be displayed in a specific window or viewport, and this windowing metadata can be defined in the DITA map within the <ux-window> element.

Application metadata specified on <ux-window> is closely tied to that application, and it might be ignored when rendering content for other uses.

User interfaces for software application often are linked to user assistance (such as help systems and tool tips) through context hooks. Context hooks are identifiers that associate a part of the user interface with the location of a help topic. Context hooks can be direct links to URIs, but more often they are indirect links (numeric context identifiers and context strings) that can processed into external resource files. These external resource and mapping files are then used directly by context-sensitive help systems and other downstream applications.

Context hooks can define either one-to-one or one-to-many relationships between user interface controls and target help content.
The metadata that is available in `<resourceid>` and `<ux-window>` provides flexibility for content developers:

- You can overload maps and topics with all the metadata needed to support multiple target help systems. This supports single-sourcing of help content and help metadata.
- You can choose whether to add `<resourceid>` metadata to `<topicref>` elements, `<prolog>` elements, or both. Context-dependent metadata might be best be kept with maps, while persistent, context-independent metadata might best stay with topics in `<prolog>` elements.

Context hook information is defined within DITA topics and DITA maps through attributes of the `<resourceid>` element.

In some help systems, a topic might need to be displayed in a specifically sized or featured window. For example, a help topic might need to be displayed immediately adjacent to the user interface control that it supports in a window of a specific size that always remains on top, regardless of the focus within the operating system. Windowing metadata can be defined in the DITA map within the `<ux-window>` element.

**Related reference**

- `<resourceid>`
- `<ux-window>`
2 Map cascading

Metadata cascading is the process by which metadata elements and attributes specified for a map or for a topic reference cascade to nested references. This allows metadata properties to be set once and apply to an entire map or branch of a map.

2.1 Cascading of metadata attributes in a DITA map

Certain map-level attributes cascade throughout a map, which facilitates attribute and metadata management. When attributes cascade, they apply to the elements that are children of the element where the attributes were specified. Cascading applies to a containment hierarchy, as opposed to a element-type hierarchy.

The following attributes cascade when set on the `<map>` element or when set within a map:

- `@rev`
- `@props` and any attribute specialized from `@props` (including those integrated by default in OASIS shells: `@audience`, `@deliveryTarget`, `@platform`, `@product`, `@otherprops`)
- `@linking`, `@toc`, `@search`
- `@format`, `@scope`, `@type`
- `@xml:lang`, `@dir`, `@translate`
- `@processing-role`
- `@cascade`

Cascading is additive for attributes that accept multiple values, except when the `@cascade` attribute is set to avoid adding values to attributes. For attributes that take a single value, the closest value defined on a containing element takes effect.

In a relationship table, row-level metadata is considered more specific than column-level metadata, as shown in the following containment hierarchy:

- `<map>` (most general)
  - `<topicref>` container (more specific)
    - `<topicref>` (most specific)
  - `<reltable>` (more specific)
    - `<relcolspec>` (more specific)
      - `<relrow>` (more specific)
        - `<relcell>` (more specific)
          - `<topicref>` (most specific)

Related reference

`topicmeta`
2.1.1 Merging of cascading attributes

The @cascade attribute can be used to modify the additive nature of attribute cascading (though it does not turn off cascading altogether). The attribute has two predefined values: "merge" and "nomerge".

`cascade="merge"

The metadata attributes cascade; the values of the metadata attributes are additive. This is the processing default for the @cascade attribute.

`cascade="nomerge"

The metadata attributes cascade; however, they are not additive for <topicref> elements that specify a different value for a specific metadata attribute. If the cascading value for an attribute is already merged based on multiple ancestor elements, that merged value continues to cascade until a new value is encountered (that is, setting cascade="nomerge" does not undo merging that took place on ancestors).

Implementers MAY define their own custom, implementation-specific tokens for the @merge attribute. To avoid name conflicts between implementations or with future additions to the standard, implementation-specific tokens SHOULD consist of a prefix that gives the name or an abbreviation for the implementation followed by a colon followed by the token or method name.

For example, a processor might define the token "appToken:audience" in order to specify cascading and merging behaviors for only the @audience attribute.

Comment by rodaande on 8 Feb 2022
It has been many years now since we defined this behavior. I’m not aware of anyone implementing their own tokens, and at this point the following paragraph defining a grouping syntax seems way too complex; can we just remove it and leave the previous paragraph?

Tokens can apply to a set of attributes, specified as part of the @cascade value. In that case, the syntax for specifying those values consists of the implementation-specific token, followed by a parenthetical group that uses the same syntax as groups within the @audience, @platform, @product, and @otherprops attributes. For example, a token that applies to only @platform and @product could be specified as cascade="appname:token(platform product)".

The predefined values for the @cascade attribute MUST precede any implementation-specific tokens, for example, cascade="merge appToken:audience".

2.1.2 Processing cascading attributes in a map

Certain rules apply to processors when they process cascading attributes in a map.

When determining the value of an attribute, processors MUST evaluate each attribute on each individual element in a specific order; this order is specified in the following list. Applications MUST continue through the list until a value is established or until the end of the list is reached (at which point no value is established for the attribute). In essence, the list provides instructions on how processors can construct a map where all attribute values are set and all cascading is complete.

For attributes within a map, the following processing order MUST occur:

1. The @conref and @keyref attributes are evaluated.
2. The explicit values specified in the document instance are evaluated. For example, a `<topicref>` element with the `@toc` attribute set to "no" will use that value.

3. The default or fixed attribute values are evaluated. For example, the `@toc` attribute on the `<reltable>` element has a default value of "no".

4. The default values that are supplied by a controlled values file are evaluated.

5. The attributes cascade.

6. The processing-supplied default values are applied.

7. After the attributes are resolved within the map, *any values that do not come from processing-supplied defaults* will cascade to referenced maps.

   For example, most processors will supply a default value of `toc="yes"` when no `@toc` attribute is specified. However, a processor-supplied default of `toc="yes"` does not override a value of `toc="no"` that is set on a referenced map. If the `toc="yes"` value is explicitly specified, is given as a default through a DTD, RNG, or controlled values file, or cascades from a containing element in the map, it will override a `toc="no"` setting on the referenced map. See 2.3 Map-to-map cascading behaviors (10) for more details.

8. Repeat steps 1 (7) to 4 (8) for each referenced map.

9. The attributes cascade within each referenced map.

10. The processing-supplied default values are applied within each referenced map.

11. Repeat the process for maps referenced within the referenced maps.

For example, in the case of `<topicref toc="yes">`, applications must stop at item 2 (8) in the list; a value is specified for `@toc` in the document instance, so `@toc` values from containing elements will not cascade to that specific `<topicref>` element. The `toc="yes"` setting on that `<topicref>` element will cascade to contained elements, provided those elements reach item 5 (8) when evaluating the `@toc` attribute.

### 2.2 Reconciling topic and map metadata elements

The `<topicmeta>` element in maps contains numerous elements that can be used to declare metadata. These metadata elements have an effect on the parent `<topicref>` element, any child `<topicref>` elements, and – if a direct child of the `<map>` element – on the map as a whole.

For each element that can be contained in the `<topicmeta>` element, the following table addresses the following questions:

**How does it apply to the topic?**

This column describes how the metadata specified within the `<topicmeta>` element interacts with the metadata specified in the topic. In most cases, the properties are additive. For example, when the `<audience>` element specifies an audience type of "user" at the map level, the type "user" is added during processing to any audience metadata that is specified within the topic.

**Does it cascade to other topics in the map?**

This column indicates whether the specified metadata value cascades to nested `<topicref>` elements. For example, when an `<audience>` element is specifies an audience type of "user" at the map level, all child `<topicref>` elements implicitly have an `<audience>` element with a type set to "user". Elements that can apply only to the specific `<topicref>` element, such as `<titlealt>` or `<keytext>`, do not cascade.
What is the purpose when specified on the `<map>` element?

The map element allows metadata to be specified for the entire map. This column describes what effect, if any, an element has when specified at this level.

Table 1: `<topicmeta>` elements and their properties

<table>
<thead>
<tr>
<th>Element</th>
<th>How does it apply to the topic?</th>
<th>Does it cascade to child <code>&lt;topicref&gt;</code> elements?</th>
<th>What is the purpose when set on the <code>&lt;map&gt;</code> element?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;audience&gt;</code></td>
<td>Add to the topic</td>
<td>Yes</td>
<td>Specify an audience for the entire map</td>
</tr>
<tr>
<td><code>&lt;author&gt;</code></td>
<td>Add to the topic</td>
<td>Yes</td>
<td>Specify an author for the entire map</td>
</tr>
<tr>
<td><code>&lt;category&gt;</code></td>
<td>Add to the topic</td>
<td>Yes</td>
<td>Specify a category for the entire map</td>
</tr>
<tr>
<td><code>&lt;copyright&gt;</code></td>
<td>Add to the topic</td>
<td>Yes</td>
<td>Specify a copyright for the entire map</td>
</tr>
<tr>
<td><code>&lt;critdates&gt;</code></td>
<td>Add to the topic</td>
<td>Yes</td>
<td>Specify critical dates for the entire map</td>
</tr>
<tr>
<td><code>&lt;data&gt;</code></td>
<td>Add to the topic</td>
<td>No, unless specialized for a purpose that cascades</td>
<td>No stated purpose</td>
</tr>
<tr>
<td><code>&lt;foreign&gt;</code></td>
<td>Add to the topic</td>
<td>No, unless specialized for a purpose that cascades</td>
<td>No stated purpose</td>
</tr>
<tr>
<td><code>&lt;keytext&gt;</code></td>
<td>Not added to the topic</td>
<td>No</td>
<td>No stated purpose</td>
</tr>
<tr>
<td><code>&lt;keywords&gt;</code></td>
<td>Add to the topic</td>
<td>No</td>
<td>No stated purpose</td>
</tr>
<tr>
<td><code>&lt;metadata&gt;</code></td>
<td>Add to the topic</td>
<td>Yes</td>
<td>Define metadata for the entire map</td>
</tr>
<tr>
<td><code>&lt;othermeta&gt;</code></td>
<td>Add to the topic</td>
<td>No</td>
<td>Define metadata for the entire map</td>
</tr>
<tr>
<td><code>&lt;permissions&gt;</code></td>
<td>Add to the topic</td>
<td>Yes</td>
<td>Specify permissions for the entire map</td>
</tr>
<tr>
<td><code>&lt;prodinfo&gt;</code></td>
<td>Add to the topic</td>
<td>Yes</td>
<td>Specify product info for the entire map</td>
</tr>
<tr>
<td><code>&lt;publisher&gt;</code></td>
<td>Add to the topic</td>
<td>Yes</td>
<td>Specify a publisher for the map</td>
</tr>
<tr>
<td><code>&lt;resourceid&gt;</code></td>
<td>Add to the topic</td>
<td>No</td>
<td>Specify a resource ID for the map itself</td>
</tr>
<tr>
<td><code>&lt;shortdesc&gt;</code></td>
<td>Applies only to links created based on this occurrence in the map</td>
<td>No</td>
<td>Provide a description of the map</td>
</tr>
<tr>
<td><code>&lt;source&gt;</code></td>
<td>Add to the topic</td>
<td>No</td>
<td>Specify a source for the map</td>
</tr>
<tr>
<td><code>&lt;titlealt&gt;</code></td>
<td>Add to the topic before its <code>&lt;titlealt&gt;</code> elements</td>
<td>No</td>
<td>Specify an alternative title for the map</td>
</tr>
<tr>
<td>Element</td>
<td>How does it apply to the topic?</td>
<td>Does it cascade to child &lt;topicref&gt; elements?</td>
<td>What is the purpose when set on the &lt;map&gt; element?</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------</td>
<td>-----------------------------------------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>&lt;unknown&gt;</td>
<td>Add to the topic</td>
<td>No, unless specialized for a purpose that cascades</td>
<td>No stated purpose</td>
</tr>
<tr>
<td>&lt;ux-window&gt;</td>
<td>Not added to the topic</td>
<td>No</td>
<td>Definitions are global, so setting at map level is equivalent to setting anywhere else.</td>
</tr>
</tbody>
</table>

Related reference
topicmeta

2.3 Map-to-map cascading behaviors

When a DITA map (or branch of a DITA map) is referenced by another DITA map, by default, certain rules apply. These rules pertain to the cascading behaviors of attributes, metadata elements, and roles assigned to content (for example, the role of “Chapter” assigned by a `<chapter>` element). Attributes and elements that cascade within a map generally follow the same rules when cascading from one map to another map, but there are some exceptions and additional rules that apply.

2.3.1 Cascading of attributes from map to map

Certain elements cascade from map to map, although some of the attributes that cascade within a map do not cascade from map to map.

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`)
- `@linking`, `@toc`, `@search`
- `@type`
- `@translate`
- `@processing-role`
- `@cascade`

Note that the above list excludes the following attributes:

* `@format`
  The `@format` attribute is set to “ditamap” when referencing a map or a branch of a map, so it cannot cascade through to the referenced map.

* `@xml:lang` and `@dir`
  Cascading behavior for `@xml:lang` is defined in The `xml:lang` attribute. The `@dir` attribute follows the same rules as `@xml:lang`.

* `@scope`
  The value of the `@scope` attribute describes the map itself, rather than the content. For example, when the `@scope` attribute is set to “external”, it indicates that the referenced map itself is external and unavailable, so the value cannot cascade into that referenced map.

The `@class` attribute is used to determine the processing roles that cascade from map to map. See 2.3.3 Cascading of roles from map to map (11) for more information.
As with values that cascade within a map, the cascading is additive if the attribute permits multiple values (such as @audience). When the attribute only permits one value, the cascading value overrides the top-level element.

### 2.3.2 Cascading of metadata elements from map to map

Elements that are contained within `<topicmeta>` elements follow the same rules for cascading from map to map as the rules that apply within a single DITA map.

For a complete list of which elements cascade within a map, see the column "Does it cascade to child `<topicref>` elements?" in the topic 2.2 Reconciling topic and map metadata elements (8).

**Comment by rodaande on 8 Feb 2022**

I don't like saying "DITA does not currently support" for a couple reasons. 1) Currently implies it's coming, and at this point it's not. 2) I think what we mean is that an application could say "This specialized `<data>` element cascades in a way we recognize" but DITA does not define a way to specify that behavior. I think the language should reflect 2) better than it does today. Perhaps:

<table>
<thead>
<tr>
<th>Not e</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is possible that a specialization might define metadata that is intended to replace rather than add to metadata in the referenced map, but DITA (by default) does not have a mechanism to specify this behavior.</td>
</tr>
</tbody>
</table>

**Note**

It is possible that a specialization might define metadata that is intended to replace rather than add to metadata in the referenced map, but DITA (by default) does not currently support this behavior.

### 2.3.3 Cascading of roles from map to map

When specialized `<topicref>` elements (such as `<chapter>` or `<mapref>`) reference a map, they typically imply a semantic role for the referenced content.

**Comment by Kristen J Eberlein on 04 July 2019**

We need to look at the instances of "should" in this topic. Can they be recast? Do we need to introduce RFC-2119 language?

The semantic role reflects the @class hierarchy of the referencing `<topicref>` element; it is equivalent to having the @class attribute from the referencing `<topicref>` cascade to the top-level `<topicref>` elements in the referenced map. Although this cascade behavior is not universal, there are general guidelines for when a role based on the @class attribute cascades.

When a `<topicref>` element or a specialization of a `<topicref>` element references a DITA resource, it defines a role for that resource. In some cases this role is straightforward, such as when a `<topicref>` element references a DITA topic (giving it the already known role of "topic"), or when a `<mapref>` element references a DITA map (giving it the role of "DITA map").

**Comment by rodaande on 8 Feb 2022**

The following paragraph includes the statement: "the non-default behavior should be clearly specified"

We do not say how or where. For mapgroup, I believe it is only specified here in this topic. I think either we need this as part of every mapgroup element definition, in "Processing expectations", or we need a clear table here listing every element where this behavior does not apply.
Unless otherwise instructed, a specialized `<topicref>` element that references a map supplies a role for the referenced content. This means that, in effect, the `@class` attribute of the referencing element cascades to top-level `<topicref>` elements in the referenced map. In situations where this should not happen—such as all elements from the mapgroup domain—the non-default behavior should be clearly specified.

For example, when a `<chapter>` element from the bookmap specialization references a map, it supplies a role of "chapter" for each top-level `<topicref>` element in the referenced map. When the `<chapter>` element references a branch in another map, it supplies a role of "chapter" for that branch. In effect, the `@class` attribute for `<chapter>` ("- map/topicref bookmap/chapter ") cascades to the top-level `<topicref>` elements in the nested map, although it does not cascade any further.

Because the `<mapref>` element is a convenience element, the top-level `<topicref>` elements in the map referenced by a `<mapref>` element MUST NOT be processed as if they are `<mapref>` elements. The `@class` attribute from the `<mapref>` element (+ map/topicref mapgroup-d/mapref ) does not cascade to the referenced map.

In some cases, preserving the role of the referencing element might result in out-of-context content. For example, a `<chapter>` element that references a bookmap might pull in `<part>` elements that contain nested `<chapter>` elements. Treating the `<part>` element as a `<chapter>` will result in a chapter that nests other chapters, which is not valid in bookmap and might not be understandable by processors. The result is implementation specific; processors MAY choose to treat this as an error, issue a warning, or simply assign new roles to the problematic elements.

### 2.4 Examples of metadata cascading

These examples illustrate the processing expectations for cascading metadata. The processing examples use either before and after sample markup or expanded syntax that shows the equivalent markup without cascading.

#### 2.4.1 Example: How metadata elements cascade to topics

In this scenario, elements in `<topicmeta>` cascade within a map and from maps to topics.

The following code sample illustrates how an information architect can apply certain metadata to all the DITA topics in a map:

```xml
<map xml:lang="en-us">
  <title>DITA maps</title>
  <topicmeta>
    <author>Kristen James Eberlein</author>
    <copyright>
      <copyryear year="2020"/>
      <copyrholder>OASIS</copyrholder>
    </copyright>
  </topicmeta>
  <topicref href="dita_maps.dita">
    <topicref href="definition_ditamaps.dita"/>
    <topicref href="purpose_ditamaps.dita"/>
    <!-- ... -->
  </topicref>
</map>
```

The author and copyright information cascades to each of the DITA topics referenced in the DITA map. When the DITA map is processed to HTML5, for example, the author and copyright metadata apply to each HTML5 file.
2.4.2 Example: How elements cascade from one map to another
In this scenario, metadata elements in one map cascade to a nested map.
Consider the following code examples:

Figure 1: test-2.ditamap

```xml
<map>
  <topicref href="a.ditamap" format="ditamap">
    <topicmeta>
      <shortdesc>This map contains information about Acme defects.</shortdesc>
    </topicmeta>
  </topicref>
  <topicref href="b.ditamap" format="ditamap">
    <topicmeta>
      <audience type="programmer"/>
    </topicmeta>
  </topicref>
  <mapref href="c.ditamap" format="ditamap"/>
  <mapref href="d.ditamap" format="ditamap"/>
</map>
```

Figure 2: b.ditamap

```xml
<map>
  <topicmeta>
    <audience type="writer"/>
  </topicmeta>
</map>
```

When test-2.ditamap is processed, the following behavior occurs:

- Because the `<shortdesc>` element does not cascade, it does not apply to the DITA topics that are referenced in a.ditamap.
- Because the `<audience>` element cascades, the `<audience>` element in the reference to b.ditamap combines with the `<audience>` element that is specified at the top level of b.ditamap. The result is that the b-1.dita topic and b-2.dita topic are processed as though they each contained the following child `<topicmeta>` element:

```xml
<topicmeta>
  <audience type="programmer"/>
  <audience type="writer"/>
</topicmeta>
```

2.4.3 Example: How attributes cascade from one map to another
In this scenario, attributes in one map cascade to a nested map.
Assume the following references in test.ditamap:

```xml
<map>
  <topicref href="a.ditamap" format="ditamap" toc="no"/>
  <mapref href="b.ditamap" audience="developer"/>
  <mapref href="c.ditamap#branch2" platform="myPlatform"/>
</map>
```

- The map a.ditamap is treated as if toc="no" is specified on the root `<map>` element. This means that the topics that are referenced by a.ditamap do not appear in the navigation generated by test.ditamap (except for branches within the map that explicitly set toc="yes").
• The map b.ditamap is treated as if audience="developer" is set on the root <map> element. If the @audience attribute is already set on the root <map> element within b.ditamap, the value "developer" is added to any existing values.

• The element with id="branch2" within the map c.ditamap is treated as if platform="myPlatform" is specified on that element. If the @platform attribute is already specified on the element with id="branch", the value "myPlatform" is added to existing values.

2.4.4 Example: How @cascade affects attribute cascading

In this scenario, the @cascade attribute is used to modify how metadata attributes cascade within a map.

**Figure 3: Example of cascade="merge"**

Consider the following code example:

```xml
<map audience="a b" cascade="merge">
  <topicref href="topic.dita" audience="c"/>
</map>
```

In this map, the cascade="merge" attribute instructs a processor to merge attribute values while cascading. With @audience specified on both the <map> element and the <topicref> element, the effective @audience attribute value for the reference to topic.dita is "a b c".

**Figure 4: Example of cascade="nomerge"**

Consider the following code example:

```xml
<map audience="a b" cascade="nomerge">
  <topicref href="topic.dita" audience="c"/>
</map>
```

In this map, the cascade="nomerge" attribute instructs a processor not to merge attribute values while cascading. With @audience specified on both the <map> element and the <topicref> element, the effective @audience attribute value on the reference to topic.dita is not merged with the value from the map and remains "c".

**Figure 5: Example: changing the @cascade value within the map**

Consider the following code example:

```xml
<map platform="a" product="x" cascade="merge">
  <topicref href="one.dita" platform="b" product="y">
    <topicref href="two.dita" cascade="nomerge" product="z"/>
  </topicref>
</map>
```

In this map, the @cascade attribute is set to "merge" at the map level but changes to "nomerge" on a topic reference.

• For the reference to one.dita, the cascade="merge" attribute is in used. This results in an effective @platform value of "a b" and an effective @product value of "y z".

• The reference to two.dita specifies cascade="nomerge", so attribute values from other elements do not merge with anything specified on this reference. The @platform attribute is not specified, so the effective value is "a b", which still cascades from the parent element. The @product value does not merge with values from the parent, so the effective value is "z".
2.4.5 Example: How <topicref> roles cascade to referenced maps

In this scenario, a specialized topicref references content in another map.

Consider the scenario of a <chapter> element from the Book map specialization that references a DITA map. This scenario could take several forms:

**Referenced map contains a single top-level <topicref> element**
- The entire branch functions as if it were included in the bookmap; the top-level <topicref> element is processed as if it were the <chapter> element.

**Referenced map contains multiple top-level <topicref> elements**
- Each top-level <topicref> element is processed as if it were a <chapter> element (the referencing element).

**Referenced map contains a single <appendix> element**
- The <appendix> element is processed as it were a <chapter> element.

**Referenced map contains a single <part> element, with nested <chapter> elements.**
- The <part> element is processed as it were a <chapter> element. Nested <chapter> elements might not be understandable by processors; applications can recover as described above.

**<chapter> element references a single <topicref> element rather than a map**
- The referenced <topicref> element is processed as if it were a <chapter> element.
# A Aggregated RFC-2119 statements

This appendix contains all the normative statements from the DITA 2.0 specification. They are aggregated here for convenience in this non-normative appendix.

<table>
<thead>
<tr>
<th>Item</th>
<th>Conformance statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>001 (7)</td>
<td>Implementers <strong>MAY</strong> define their own custom, implementation-specific tokens for the <code>@merge</code> attribute. To avoid name conflicts between implementations or with future additions to the standard, implementation-specific tokens <strong>SHOULD</strong> consist of a prefix that gives the name or an abbreviation for the implementation followed by a colon followed by the token or method name.</td>
</tr>
<tr>
<td>002 (7)</td>
<td>The predefined values for the <code>@cascade</code> attribute <strong>MUST</strong> precede any implementation-specific tokens, for example, <code>cascade=&quot;merge appToken:audience&quot;</code>.</td>
</tr>
<tr>
<td>003 (7)</td>
<td>When determining the value of an attribute, processors <strong>MUST</strong> evaluate each attribute on each individual element in a specific order; this order is specified in the following list. Applications <strong>MUST</strong> continue through the list until a value is established or until the end of the list is reached (at which point no value is established for the attribute). In essence, the list provides instructions on how processors can construct a map where all attribute values are set and all cascading is complete.</td>
</tr>
</tbody>
</table>
| 004 (7) | For attributes within a map, the following processing order **MUST** occur:  
1. The `@conref` and `@keyref` attributes are evaluated.  
2. The explicit values specified in the document instance are evaluated. For example, a `<topicref>` element with the `@toc` attribute set to "no" will use that value.  
3. The default or fixed attribute values are evaluated. For example, the `@toc` attribute on the `<reltable>` element has a default value of "no".  
4. The default values that are supplied by a controlled values file are evaluated.  
5. The attributes cascade.  
6. The processing-supplied default values are applied.  
7. After the attributes are resolved within the map, any values that do not come from *processing-supplied defaults* will cascade to referenced maps.  

For example, most processors will supply a default value of `toc="yes"` when no `@toc` attribute is specified. However, a processor-supplied default of `toc="yes"` does not override a value of `toc="no"` that is set on a referenced map. If the `toc="yes"` value is explicitly specified, is given as a default through a DTD, RNG, or controlled values file, or cascades from a containing element in the map, it will override a `toc="no"` setting on the referenced map. See 2.3 Map-to-map cascading behaviors (10) for more details.  
8. Repeat steps 1 to 4 for each referenced map.  
9. The attributes cascade within each referenced map.  
10. The processing-supplied default values are applied within each referenced map.  
11. Repeat the process for maps referenced within the referenced maps.  
| 005 (12) | Because the `<mapref>` element is a convenience element, the top-level `<topicref>` elements in the map referenced by a `<mapref>` element **MUST NOT** be processed as if they are `<mapref>` elements. The `@class` attribute from the `<mapref>` element (`"+ map/topicref mapgroup-d/mapref ") does not cascade to the referenced map.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 006 (12) | In some cases, preserving the role of the referencing element might result in out-of-context content. For example, a `<chapter>` element that references a bookmark might pull in `<part>` elements that contain nested `<chapter>` elements. **T**reating the `<part>` element as a `<chapter>` will result in a chapter that nests other chapters, which is not valid in bookmark and might not be understandable by
<table>
<thead>
<tr>
<th>Item</th>
<th>Conformance statement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>processors. The result is implementation specific; processors <strong>MAY</strong> choose to treat this as an error, issue a warning, or simply assign new roles to the problematic elements.</td>
</tr>
</tbody>
</table>
Index

C

cascading
definition 6
map-to-map
attributes 10
exceptions 11
metadata elements 11
cascading metadata
example
cascade attribute 14
cascading attributes between maps 13
cascading elements 12
cascading elements between maps 13
cascading topicref roles 15
elements 12

D
definitions
cascading 6

E
examples
cascading metadata
cascade attribute 14
cascading attributes between maps 13
cascading elements 12
cascading elements between maps 13
cascading topicref roles 15
overview 12

M
map-to-map cascading
attributes 10
exceptions 11
metadata elements 11
metadata
cascading 3
elements 3

T
terminology
cascading 6