Subject scheme and cascading metadata

Table of contents

1 Subject scheme maps and their usage

Subject scheme maps can be used to define controlled values and subject definitions. The controlled values can be bound to attributes, as well as element and attribute pairs. The subject definitions can contain metadata and provide links to more detailed information; they can be used to classify content and provide semantics that can be used in taxonomies and ontologies.

A DITA map can reference a subject scheme map by using a `<mapref>` element. Processors also **MAY** provide parameters by which subject scheme maps are referenced.

1.1 Subject scheme maps

Subject scheme maps use key definitions to define collections of controlled values and subject definitions.

Controlled values are tokens that can be used as values for attributes. For example, the `@audience` attribute can take a value that identifies the users that are associated with a particular product. Typical values for a medical-equipment product might include "therapist", "oncologist", "physicist", and "radiologist". In a subject scheme map, an information architect can define a list of these values for the `@audience` attribute. An authoring tool can then provide a pick list for values for the attribute and generate a warning if an author attempts to specify a value that is not one of the controlled values. Controlled values can also be used to select content for filtering and flagging at build time.

Subject definitions are classifications and sub-classifications that compose a tree. Subject definitions provide semantics that can be used in conjunction with taxonomies and ontologies.

Key references to controlled values are resolved to a key definition using the same precedence rules as apply to any other key. However, once a key is resolved to a controlled value, that key reference does not typically result in links or generated text.

Comment by Kristen J Eberlein on 14 December 2021

Adding content from DITAweb review D:

Comment from Stan Doherty: FWIW -- I do not understand what the second sentence ["However, omce a key is resolved ... "] means.

Comment from Kris Eberlein: Quite simply, that key references resolved within a subjectScheme map do NOT generate variable text or produce links. Within the context of a subjectScheme map, the key references provide bindings or associations with subjects.

Comment from Robert Anderson: I think the root of this problem / this misunderstanding is the poor design choice of using the same keys/keyref attribute for Subject Schemes as we do for normal linking / variable text. We had an item in the 2.0 queue to completely redesign that, but never had anyone with the time / energy to work on it (it would have been a big change).

The problem here is that we have to explain "These don't work like normal keys, and you shouldn't use them in links and expect them to resolve as text or links" -- in a way that is clear, accurate, and short enough that it actually gets read. So, I think we need some work on this paragraph.

1.2 Defining controlled values for attributes

Subject scheme maps can define controlled values for DITA attributes without having to define specializations or constraints. The list of available values can be modified quickly to adapt to new situations.

Each controlled value is defined using a `<subjectdef>` element, which is a specialization of the `<topicref>` element. The `<subjectdef>` element is used to define both a subject category and a list of controlled values. The parent `<subjectdef>` element defines the category, and the children `<subjectdef>` elements define the controlled values.

The subject definitions can include additional information within a `<topicmeta>` element to clarify the meaning of a value:

- A `<navtitle>` (or a `<titlealt>` element with a `@title-role` of `navigation`) can provide a more readable name for the controlled value.
- The `<shortdesc>` element can provide a definition.

In addition, the `<subjectdef>` element can reference a more detailed definition of the subject, for example, another DITA topic or an external resource.

The following behavior is expected of processors in regard to subject scheme maps:

- Authoring tools **SHOULD** use these lists of controlled values to provide lists from which authors can select values when they specify attribute values.
- Authoring tools **MAY** give an organization a list of readable labels, a hierarchy of values to simplify selection, and a shared definition of the value.
- Authoring tools **MAY** support accessing and displaying the content of the subject definition resource in order to provide users with a detailed explanation of the subject.

Example: Controlled values that provide additional information about the subject

The following code sample illustrates how a subject definition can provide a richer level of information about a controlled value:

```
<subjectdef keys="terminology" href="https://www.oasis-open.org/policies-guidelines/keyword-
guidelines">
  <subjectdef keys="rfc2119" href="rfc-2119.dita">
    <topicmeta>
      <navtitle>RFC-2119 terminology</navtitle>
      <shortdesc>The normative terminology that the DITA TC uses for the DITA specification</
shortdesc>
    </topicmeta>
  </subjectdef>
  <subjectdef keys="iso" href="iso-terminology.dita">
    <topicmeta>
      <navtitle>ISO keywords</navtitle>
      <shortdesc>The normative terminology used by some other OASIS technical committees
      </shortdesc>
    </topicmeta>
  </subjectdef>
</subjectdef>
```

The content of the `<navtitle>` and `<shortdesc>` elements provide additional information that a processor might display to users as they select attribute values or classify content. The resources referenced by the `@href` attributes provide even more detailed information. A processor might render expandable links as part of a user interface that implements a progressive disclosure strategy, or an authoring tool might include the navigation title and short description in a window where the user selects a controlled value.

1.3 Binding controlled values to an attribute

The controlled values defined in a subject scheme map can be bound to an attribute or an element and attribute pair. This affects the expected behavior for processors and authoring tools.

The `<enumerationdef>` element binds the set of controlled values to an attribute. Valid attribute values are those that are defined in the set of controlled values. Invalid attribute values are those that are not defined in the set of controlled values. If an enumeration specifies an empty `<subjectdef>` element that does not reference a set of controlled values, no value is valid for the attribute. An enumeration can also specify an optional default value by using the `<defaultSubject>` element.

If an enumeration is bound, processors **SHOULD** validate attribute values against the controlled values that are defined in the subject scheme map. For authoring tools, this validation prevents users from entering misspelled or undefined values. Recovery from validation errors is implementation specific.

The default attribute values that are specified in a subject scheme map apply only if a value is not otherwise specified in the DITA source or as a default value by the XML grammar.

Example: Binding a list of controlled values to the @audience attribute

The following code sample illustrates the use of the `<subjectdef>` element to define controlled values for types of users. It also binds the controlled values to the `@audience` attribute:

```
<subjectScheme>
  <!-- DEFINE TYPES OF USERS -->
  <subjectdef keys="users">
    <subjectdef keys="therapist"/>
    <subjectdef keys="oncologist"/>
    <subjectdef keys="physicist"/>
    <subjectdef keys="radiologist"/>
  </subjectdef>
  <!-- BIND THE SUBJECT TO THE @AUDIENCE ATTRIBUTE
       This restricts the @audience attribute to the following
       values: therapist, oncologist, physicist, radiologist -->
  <enumerationdef>
    <attributedef name="audience"/>
    <subjectdef keyref="users"/>
  </enumerationdef>
</subjectScheme>
```

When the above subject scheme map is used, the only valid values for the `@audience` attribute are "therapist", "oncologist", "physicist", and "radiologist". Note that "users" is not a valid value for the `@audience` attribute, as it merely identifies the parent or container subject.

Example: Binding an attribute to an empty set

The following code sample specifies that there are no valid values for the `@outputclass` attribute:

```
<subjectScheme>
  <enumerationdef>
    <attributedef name="outputclass"/>
    <subjectdef/>
  </enumerationdef>
</subjectScheme>
```

Authors will not be able to specify the `@outputclass` attribute on an element.

1.4 Processing controlled attribute values

An enumeration of controlled values can be defined with hierarchical levels by nesting subject definitions. This affects how processors perform filtering and flagging.

The following behavior is expected of processors in regard to subject scheme maps:

- Processors **SHOULD** be aware of the hierarchies of attribute values that are defined in subject scheme maps for purposes of filtering, flagging, or other metadata-based categorization.
- Processors **SHOULD** validate that the values of attributes that are bound to controlled values contain only valid values from those sets. This requirement is needed because basic XML parsers do not validate the list of controlled values. If the controlled values are part of a named key scope, the scope name is ignored for the purpose of validating the controlled values.
- Processors **SHOULD** check that all values listed for an attribute in a DITAVAL file are bound to the attribute by the subject scheme before filtering or flagging. If a processor encounters values that are not included in the subject scheme, it **SHOULD** issue a warning.

Processors **SHOULD** apply the following algorithm when they apply filtering and flagging rules to attribute values that are defined as a hierarchy of controlled values and bound to an enumeration:

1. If an attribute specifies a value in the taxonomy, and a DITAVAL or other categorization tool is configured with that value, the rule matches.
2. Otherwise, if the parent value in the taxonomy has a rule, that matches.
3. Otherwise, continue up the chain in the taxonomy until a matching rule is found.

Example: A hierarchy of controlled values and conditional processing

The following code sample shows a set of controlled values that contains a hierarchy.

```
<subjectScheme>
  <subjectdef keys="users">
    <subjectdef keys="therapist">
      <subjectdef keys="novice-therapist"/>
      <subjectdef keys="expert-therapist"/>
    </subjectdef>
    <subjectdef keys="oncologist"/>
    <subjectdef keys="physicist"/>
    <subjectdef keys="radiologist"/>
  </subjectdef>
  <enumerationdef>
    <attributedef name="audience"/>
    <subjectdef keyref="users"/>
  </enumerationdef>
</subjectScheme>
```

Processors that are aware of the hierarchy that is defined in the subject scheme map will handle filtering and flagging in the following ways:

- If "therapist" is excluded, both "novice-therapist" and "expert-therapist" are by default excluded.
- If "therapist" is flagged and "novice-therapist" is not explicitly flagged, processors automatically flag "novice-therapist" since it is a type of therapist.

1.5 The @subjectrefs attribute

The `@subjectrefs` attribute specifies one or more keys that are defined by a subject definition in a subject scheme map. Multiple values are separated by white space.

The `@subjectrefs` attribute cascades. It can be used on a `<topicref>` element to associate the referenced topic with a subject defined in a subject scheme map.

The `@subjectrefs` attribute has no meaning if it is specified on a key definition that does not reference a resource.

1.6 Examples of subject scheme maps

This section contains examples and scenarios that illustrate the use of subject scheme maps.

1.6.1 Example: A subject scheme map used to define taxonomic subjects

A subject scheme map can be used to define taxonomic subjects. Once defined, the subjects can be referenced by applying a `@subjectrefs` attribute to a `<topicref>` element.

The following subject scheme map defines a set of subjects that are used to classify content:

```
<subjectScheme>
  <subjectdef keys="content-types">
    <subjectdef keys="conceptual-material"/>
    <subjectdef keys="reference"/>
    <subjectdef keys="tutorial"/>
  </subjectdef>
  <subjectdef keys="operating-systems">
    <subjectdef keys="linux"/>
    <subjectdef keys="macosx"/>
    <subjectdef keys="windows"/>
  </subjectdef>
  <subjectdef keys="user-tasks">
    <subjectdef keys="administering"/>
    <subjectdef keys="developing"/>
    <subjectdef keys="installing"/>
    <subjectdef keys="troubleshooting"/>
  </subjectdef>
</subjectScheme>
```

The keys assigned to the subject definitions can be referenced by specifying the `@subjectrefs` attribute on topic references in a navigation map:

```
<map>
<title>User assistance for the Acme Widget</title>
<!-- ... -->
<topicref keyref="install-overview" subjectrefs="installing">
  <topicref keyref="install-linux"/>
  <topicref keyref="install-macosx"/>
  <topicref keyref="install-windows"/>
  <topicref keyref="install-troubleshooting" subjectrefs="troubleshooting"/>
</topicref>
<!-- ... -->
</map>
```

Because the `@subjectrefs` attribute cascades, the effective value of the above markup is the same as the following markup:

```
<map>
<title>User assistance for the Acme Widget</title>
<!-- ... -->
<topicref keyref="install-overview" subjectrefs="installing">
  <topicref keyref="install-linux" subjectrefs="installing"/>
  <topicref keyref="install-macosx" subjectrefs="installing"/>
  <topicref keyref="install-windows" subjectrefs="installing"/>
  <topicref keyref="install-troubleshooting" subjectrefs="installing troubleshooting"/>
</topicref>
<!-- ... -->
</map>
```

1.6.2 Example: How hierarchies defined in a subject scheme map affect filtering

This scenario demonstrates how a processor evaluates attribute values when it performs conditional processing for an attribute that is bound to a set of controlled values.

A company defines a subject category for "Operating system", with a key set to "os". There are sub-categories for Linux, Windows, and z/OS, as well as specific Linux variants: Red Hat Linux and SuSE Linux. The company then binds the values that are enumerated in the "Operating system" category to the `@platform` attribute:

```
<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>
```

The enumeration limits valid values for the `@platform` attribute to the following: "linux", "redhat", "suse", "windows", and "zos". If any other values are encountered, processors validating against the scheme will issue a warning.

The following table illustrates how filtering and flagging operate when the above map is processed by a processor. The first two columns provide the values specified in the DITAVAL file. The third and fourth columns indicate the results of the filtering or flagging operation.

`att="platform" val="linux"`	`att="platform" val="redhat"`	**How `platform="redhat"` is evaluated**	**How `platform="linux"` is evaluated**
action="exclude"	action="exclude"	Excluded.	Excluded.
	action="include" or action="flag"	Excluded. This is an error condition, because if all "linux" content is excluded, "redhat" also is excluded. Applications can recover by generating an error message.	Excluded.
	Unspecified	Excluded, because "redhat" is a kind of "linux", and "linux" is excluded.	Excluded.
action="include"	action="exclude"	Excluded, because all "redhat" content is excluded.	Included.
	action="include"	Included.	Included.
	action="flag"	Included and flagged with the "redhat" flag.	Included.
	Unspecified	Included, because all "linux" content is included.	Included.
action="flag"	action="exclude"	Excluded, because all "redhat" content is excluded.	Included and flagged with the "linux" flag.
	action="include"	Included and flagged with the "linux" flag, because "linux" is flagged and "redhat" is a type of "linux".	Included and flagged with the "linux" flag.
	action="flag"	Included and flagged with the "redhat" flag, because a flag is available that is specifically for "redhat".	Included and flagged with the "linux" flag.
	Unspecified	Included and flagged with the "linux" flag, because "linux" is flagged and "redhat" is a type of "linux"	Included and flagged with the "linux" flag.

att="platform" val="linux"	att="platform" val="redhat"	How platform="redhat" is evaluated	How platform="linux" is evaluated
Unspecified	action="exclude"	Excluded, because all "redhat" content is excluded	If the default value for @platform set in the DITAVAL is "include", this is included. If the default value for @platform set in the DITAVAL is "exclude", this is excluded.
	action="include"	Included.	Included, because all "redhat" content is included, and general Linux content also applies to RedHat
	action="flag"	Included and flagged with the "redhat" flag.	Included, because all "redhat" content is included, and general Linux content also applies to RedHat
	Unspecified	If the default value for @platform set in the DITAVAL is "include", this is included. If the default value for @platform set in the DITAVAL is "exclude", this is excluded.	If the default value for @platform set in the DITAVAL is "include", this is included. If the default value for @platform set in the DITAVAL is "exclude", this is excluded.

1.6.3 Example: Defining values for @deliveryTarget

You can use a subject scheme map to define the values for the @deliveryTarget attribute. This filtering attribute is intended for use with a set of hierarchical, controlled values.

In this scenario, one department produces electronic publications (EPUB, EPUB2, EPUB3, Kindle, etc.) while another department produces traditional, print-focused output. Each department needs to exclude a certain category of content when they build documentation deliverables.

The following subject scheme map provides a set of values for the @deliveryTarget attribute that accommodates the needs of both departments.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE subjectScheme PUBLIC "-//OASIS//DTD DITA Subject Scheme Map//EN"
"subjectScheme.dtd">
<subjectScheme>
  <subjectHead>
    <subjectHeadMeta>
      <navtitle>Example of values for the @deliveryTarget attribute</navtitle>
      <shortdesc>Provides a set of values for use with the
        @deliveryTarget conditional-processing attribute. This set of values is
        illustrative only; you can use any values with the @deliveryTarget
        attribute.</shortdesc>
    </subjectHeadMeta>
  </subjectHead>
  <subjectdef keys="deliveryTargetValues">
    <topicmeta><navtitle>Values for @deliveryTarget attributes</navtitle></topicmeta>
    <!-- A tree of related values -->
    <subjectdef keys="print">
      <topicmeta><navtitle>Print-primary deliverables</navtitle></topicmeta>
      <subjectdef keys="pdf">
```

```
        <topicmeta><navtitle>PDF</navtitle></topicmeta>
      </subjectdef>
      <subjectdef keys="css-print">
        <topicmeta><navtitle>CSS for print</navtitle></topicmeta>
      </subjectdef>
      <subjectdef keys="xsl-fo">
        <topicmeta><navtitle>XSL-FO</navtitle></topicmeta>
      </subjectdef>
      <subjectdef keys="afp">
        <topicmeta><navtitle>Advanced Function Printing</navtitle></topicmeta>
      </subjectdef>
      <subjectdef keys="ms-word">
        <topicmeta><navtitle>Microsoft Word</navtitle></topicmeta>
      </subjectdef>
      <subjectdef keys="indesign">
        <topicmeta><navtitle>Adobe InDesign</navtitle></topicmeta>
      </subjectdef>
      <subjectdef keys="open-office">
        <topicmeta><navtitle>Open Office</navtitle></topicmeta>
      </subjectdef>
    </subjectdef>
    <subjectdef keys="online">
      <topicmeta><navtitle>Online deliverables</navtitle></topicmeta>
      <subjectdef keys="html-based">
        <topicmeta><navtitle>HTML-based deliverables</navtitle></topicmeta>
        <subjectdef keys="html">
          <topicmeta><navtitle>HTML</navtitle></topicmeta>
          <subjectdef keys="html5">
            <topicmeta><navtitle>HTML5</navtitle></topicmeta>
          </subjectdef>
        </subjectdef>
        <subjectdef keys="help">
          <topicmeta><navtitle>Contextual help</navtitle></topicmeta>
          <subjectdef keys="htmlhelp">
            <topicmeta><navtitle>HTML Help</navtitle></topicmeta>
          </subjectdef>
          <subjectdef keys="webhelp">
            <topicmeta><navtitle>Web help</navtitle></topicmeta>
          </subjectdef>
          <subjectdef keys="javahelp">
            <topicmeta><navtitle>Java Help</navtitle></topicmeta>
          </subjectdef>
          <subjectdef keys="eclipseinfocenter">
            <topicmeta><navtitle>Eclipse InfoCenter</navtitle></topicmeta>
          </subjectdef>
        </subjectdef>
        <subjectdef keys="epub">
          <topicmeta><navtitle>EPUB</navtitle></topicmeta>
          <subjectdef keys="epub2">
            <topicmeta><navtitle>EPUB2</navtitle></topicmeta>
          </subjectdef>
          <subjectdef keys="epub3">
            <topicmeta><navtitle>EPUB3</navtitle></topicmeta>
          </subjectdef>
          <subjectdef keys="ibooks">
            <topicmeta><navtitle>iBooks</navtitle></topicmeta>
          </subjectdef>
          <subjectdef keys="nook">
            <topicmeta><navtitle>nook</navtitle></topicmeta>
          </subjectdef>
        </subjectdef>
        <subjectdef keys="kindle">
          <topicmeta><navtitle>Amazon Kindle</navtitle></topicmeta>
          <subjectdef keys="kindle8">
            <topicmeta><navtitle>Kindle Version 8</navtitle></topicmeta>
          </subjectdef>
        </subjectdef>
      </subjectdef>
    </subjectdef>
  </subjectdef>
  <enumerationdef>
    <attributedef name="deliveryTarget"/>
    <subjectdef   keyref="deliveryTargetValues"/>
  </enumerationdef>
</subjectScheme>
```

2 Metadata 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 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 specialization 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 the OASIS-provided document-type shells: `@audience`, `@deliveryTarget`, `@platform`, `@product`, `@otherprops`
- `@linking`, `@toc`, `@search`
- `@format`, `@scope`, `@type`
- `@xml:lang`, `@dir`, `@translate`
- `@processing-role`
- `@cascade`
- `@subjectrefs`

Cascading is additive for attributes that accept multiple values, except when `cascade="nomerge"` is specified. For attributes that take a single value, the value that is defined on the closest containing element takes effect.

In a relationship table, metadata can be applied to entire rows or columns, as well as individual cells. The metadata cascade operates differently due to the nature of this tabular structure The cascade is not driven by a strict containment hierarchy because `<relcolspec>` elements do not contain child elements.

The following list illustrates how metadata cascades in a relationship table:

- `<reltable>`
  - `<relcolspec>`
    - `<relrow>`
      - `<relcell>`
        - `<topicref>`

Related reference
topicmeta

2.1.1 Processing cascading attributes in a map

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

Comment by Kristen J Eberlein on 04 June 2022

Robert Anderson has an open action item to rework the ordered list.

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 Map-to-map cascading behaviors (17) for more details.
8. Repeat steps 1 (14) to 4 (14) 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 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 when evaluating the `@toc` attribute.

2.1.2 Merging of cascading attributes

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

merge

Indicates that the metadata attributes cascade, and that the values of the metadata attributes are additive. This is the processing default for the `@cascade` attribute.

nomerge

Indicates that the metadata attributes cascade, but that 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 ancestor elements.

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.

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

2.2 Reconciling topic and map metadata elements

The `<topicmeta>` element in maps can contain numerous metadata elements. These metadata elements can have an effect on the parent `<topicref>` element, any child `<topicref>` elements, and – if a direct child of the `<map>` element – on the .

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 referenced topic. In most cases, the properties are additive. For example, when a topic reference in a map contains `<category>installation</category>`, `<category>installation</category>` is added during processing to any metadata that is specified in the topic prolog.

Does it cascade to other topics in the map?
This column indicates whether the specified metadata element cascades to nested `<topicref>` elements. For example, when a topic reference in a map contains `<author>Jane Doe</author>`, `<author>Jane Doe</author>` is added during processing to the metadata for all child topic references. Some elements do not cascade.

What is the purpose when specified on the <map> element?
The map element permits metadata to be specified for the map. This column describes the effect that an element has when specified at this level.

When set on the <map> element, does it apply to all topics referenced in the map?
When specified on the `<map>` element element, some metadata elements then apply to all the topics that are referenced in the map.

Table 1: <topicmeta> elements and their properties

Element	How does it apply to the topic?	Does it cascade to child `<topicref>` elements?	What is the purpose when set on the `<map>` element?	When set on the `<map>` element, does it apply to all topics referenced in the map?
`<audience>`	Add to the topic	Yes	Specify an audience for the map	Yes
`<author>`	Add to the topic	Yes	Specify an author for the map	Yes
`<category>`	Add to the topic	Yes	Specify a category for the map	Yes

Element	How does it apply to the topic?	Does it cascade to child `<topicref>` elements?	What is the purpose when set on the `<map>` element?	When set on the `<map>` element, does it apply to all topics referenced in the map?
`<copyright>`	Add to the topic	Yes	Specify a copyright for the map	Yes
`<critdates>`	Add to the topic	Yes	Specify critical dates for the map	Yes
`<data>`	Add to the topic	No, unless specialized for a purpose that cascades	No stated purpose	No
`<foreign>`	Add to the topic	No, unless specialized for a purpose that cascades	No stated purpose	No
`<keytext>`	Not added to the topic	No	No stated purpose	No
`<keywords>`	Add to the topic	No	No stated purpose	No
`<metadata>`	Add to the topic	Yes	Specify metadata for the map	Yes
`<othermeta>`	Add to the topic	No	Define metadata for the map	Yes
`<permissions>`	Add to the topic	Yes	Specify permissions for the map	Yes
`<prodinfo>`	Add to the topic	Yes	Specify product info for the map	Yes
`<publisher>`	Add to the topic	Yes	Specify a publisher for the map	No
`<resourceid>`	Add to the topic	No	Specify a resource ID for the map itself	No
`<shortdesc>`	Applies only to links created based on this occurrence in the map	No	Provide a description of the map	No
`<source>`	Add to the topic	No	Specify a source for the map	No
`<titlealt>`	Add to the topic before its `<titlealt>` elements	No	Specify an alternative title for the map	No
`<unknown>`	Add to the topic	No, unless specialized for a purpose that cascades	No stated purpose	No

Element	How does it apply to the topic?	Does it cascade to child `<topicref>` elements?	What is the purpose when set on the `<map>` element?	When set on the `<map>` element, does it apply to all topics referenced in the map?
`<ux-window>`	Not added to the topic	No	Definitions are global, so setting at map level is equivalent to setting anywhere else.	No

Related reference

topicmeta

2.3 Map-to-map cascading behaviors

When a DITA map or map branch is referenced by another DITA map, by default certain rules apply. These rules pertain to the cascading behaviors of attributes, metadata elements, and the roles that are assigned to content , for example, the role of "Chapter" that is 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 attributes 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 the OASIS-provided document-type shells: `@audience`, `@deliveryTarget`, `@platform`, `@product`, `@otherprops`
- `@linking`, `@toc`, `@search`
- `@type`
- `@translate`
- `@processing-role`
- `@cascade`
- `@subjectrefs`

As with values that cascade within a map, the cascading is additive if the attribute permits multiple values, such as `@audience`. For attributes that take a single value, the value that is defined on the closest containing element takes effect.

The following attributes do not cascade from map to map

@format
: The `@format` attribute is set to "ditamap" when a map or map branch is referenced, so it cannot cascade through to the referenced map.

@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.

@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`.

While the `@class` attribute is unique and does not cascade, the value of the attribute is used to determine the processing roles that cascade from map to map. See Cascading of roles from map to map (18) for more information.

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 Reconciling topic and map metadata elements (15).

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 have a mechanism to specify 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 June 2022

Proposal #670, championed by Robert Anderson, will rework this content.

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").

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 withough cascading.

2.4.1 Example: How map-level metadata elements cascade to the referenced topics

In this scenario, elements located in the`<topicmeta>` element for a map cascade to the referenced topics.

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

```
<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 that are referenced in the DITA map. When the DITA map is processed to HTML5, for example, the author and copyright metadata apply to each generated HTML5 file.

2.4.2 Example: How metadata elements cascade from one map to another

In this scenario, a metadata element that is located in a map reference cascades to the topics that are referenced in a nested map.

Consider the following code examples:

Figure 1: Root map

```
<map>
  <title>Acme User Guide</title>
  <topicref href="acme-defects.ditamap" format="ditamap">
    <topicmeta>
      <shortdesc>This map contains information about Acme defects.</shortdesc>
    </topicmeta>
  </topicref>
  <topicref href="installing.ditamap" format="ditamap">
    <topicmeta>
      <audience type="installer"/>
    </topicmeta>
  </topicref>
  <mapref href="troubleshooting.ditamap"/>
  <mapref href="reference.ditamap"/>
</map>
```

Figure 2: installing.ditamap

```
<map>
  <title>Installation topics</title>
  <topicmeta>
    <audience type="administrator"/>
  </topicmeta>
  <topicref href="install-1.dita"/>
  <topicref href="install-2.dita"/>
</map>
```

When the root map 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 `acme-defects.ditamap`.
- Because the `<audience>` element cascades, the `<audience>` element in the reference to `installing.ditamap` combines with the `<audience>` element that is specified at the top level of `installing.ditamap`. The result is that the `install-1.dita` and `install-2.dita` topics are processed as though they each contained the following child `<topicmeta>` element:

  ```
  <topicmeta>
    <audience type="installer"/>
    <audience type="administrator"/>
  </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`:

```
<map>
  <topicref href="a.ditamap" format="ditamap" toc="no"/>
  <mapref href="b.ditamap" audience="developer"/>
  <mapref href="c.ditamap#branch2" platform="myPlatform"/>
  <mapref> href="d.ditamap" subjectrefs="puzzles"/>
</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.
- The map `d.ditamap` is treated as if `subjectrefs="puzzles"` is set on the root `<map>` element. If the `@subjectrefs` attribute is already set on the root `<map>` element within `d.ditamap`, the value "puzzles" is added to any existing values.

2.4.4 Example: How the @cascade attribute 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:

```
<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:

```
<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 of changing the @cascade value within the map

Consider the following code example:

```
<map platform="a" product="x" cascade="merge">
  <topicref href="one.dita" platform="b" product="y">
    <topicref href="two.dita">
      <topicref href="three.dita" cascade="nomerge" product="z">
        <topicref href="four.dita"/>
      </topicref>
    </topicref>
  </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 topic reference to `one.dita`, `cascade="merge"` is specified. This results in an effective `@platform` value of "a b" and an effective `@product` value of "x y".
- The topic reference to `two.dita` does not specify any additional attributes. The effective values for the `@platform` and `@product` attributes are the same as those on the parent topic reference to `one.dita`. The effective value of of the `@platform` attribute is "a b", and the effective value for the `@product` attribute is "x y".
- The topic reference to `three.dita` specifies `cascade="nomerge"`, so attribute values from other elements do not merge with anything specified on the topic 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".
- The topic reference to `four.dita` does not specify any additional attributes. The effective values for the `@platform` and `@product` attributes are the same as those on the parent topic reference

to `three.dita`. The effective value of of the `@platform` attribute is "a b", and the effective value for the `@product` attribute is "z".

2.4.5 Example: How <topicref> roles cascade to referenced maps

In this scenario, a specialized `<topicref>` element references content in another map.

Comment by Kristen J Eberlein on 04 June 2022

Proposal #670, championed by Robert Anderson, will rework this content.

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, since the processing role of the `<chapter>` element cascades.

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, although 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.

Appendix 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.

Rule number	Conformance statement
Rule 1 (3)	A DITA map can reference a subject scheme map by using a `<mapref>` element. Processors also **MAY** provide parameters by which subject scheme maps are referenced.
Rule 2 (4)	The following behavior is expected of processors in regard to subject scheme maps: • Authoring tools **SHOULD** use these lists of controlled values to provide lists from which authors can select values when they specify attribute values. • Authoring tools **MAY** give an organization a list of readable labels, a hierarchy of values to simplify selection, and a shared definition of the value. • Authoring tools **MAY** support accessing and displaying the content of the subject definition resource in order to provide users with a detailed explanation of the subject.
Rule 3 (5)	If an enumeration is bound, processors **SHOULD** validate attribute values against the controlled values that are defined in the subject scheme map. For authoring tools, this validation prevents users from entering misspelled or undefined values. Recovery from validation errors is implementation specific.
Rule 4 (6)	The following behavior is expected of processors in regard to subject scheme maps: • Processors **SHOULD** be aware of the hierarchies of attribute values that are defined in subject scheme maps for purposes of filtering, flagging, or other metadata-based categorization. • Processors **SHOULD** validate that the values of attributes that are bound to controlled values contain only valid values from those sets. This requirement is needed because basic XML parsers do not validate the list of controlled values. If the controlled values are part of a named key scope, the scope name is ignored for the purpose of validating the controlled values. • Processors **SHOULD** check that all values listed for an attribute in a DITAVAL file are bound to the attribute by the subject scheme before filtering or flagging. If a processor encounters values that are not included in the subject scheme, it **SHOULD** issue a warning.
Rule 5 (6)	Processors **SHOULD** apply the following algorithm when they apply filtering and flagging rules to attribute values that are defined as a hierarchy of controlled values and bound to an enumeration: **1.** If an attribute specifies a value in the taxonomy, and a DITAVAL or other categorization tool is configured with that value, the rule matches. **2.** Otherwise, if the parent value in the taxonomy has a rule, that matches. **3.** Otherwise, continue up the chain in the taxonomy until a matching rule is found.
Rule 6 (14)	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.
Rule 7 (14)	For attributes within a map, the following processing order **MUST** occur: **1.** The `@conref` and `@keyref` attributes are evaluated.

Rule number	Conformance statement
	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 Map-to-map cascading behaviors (17) for more details. **8.** Repeat steps 1 (14) to 4 (14) 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.
Rule 8 (15)	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.
Rule 9 (15)	The predefined values for the `@cascade` attribute **MUST** precede any implementation-specific tokens, for example, `cascade="merge appToken:audience"`.
Rule 10 (18)	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.
Rule 11 (19)	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.

Index