Review A: Task elements

Table of contents

1 Topic and map document types

The Technical Content package contains various document types: concept, glossary entry, glossary group, reference, general task, strict task, and troubleshooting. The package also includes the bookmap document type.

1.1 Task

DITA offer

Comment by Stan Doherty
Change "offer" to "offers". Note also that there are two missing spaces between words in the previous "Topic and map document types" topic.

Kris Eberlein, 31 October 2022

Both are now fixed. Thanks for catchng these errors.

Disposition: Completed

two varieties of task topics: general task and strict task.

1.1.1 General task

A general task topic answers the "How do I?" question by providing instructions and other necessary information that enables the user to complete the task successfully. It has a content model that is more relaxed than that of the strict task.

Comment by dstevens73
why is shortdesc in abstract in this and the next topic, but no where else?

Kris Eberlein, 19 October 2022

Strictly an artifact of the editing process. Usually when I revise a short description, I surround it with an abstract element and then add a 2nd shortdesc, so that I can compare the two versions side-by-side.

I've removed the <abstract> element.

Disposition: Completed

Purpose

The purpose of the general task topic is to provide the user with comprehensive instructions for performing a task. In addition, its content model can facilitate the migration of procedural information that does not follow the content model of the strict task topic.

Content model

The general task topic is divided into three parts:

Comment by Zoë Lawson on 30 Oct 2022
I'm not quite sure why general task gets the three parts/sections (Introduction, procedure, post), and the strict task doesn't. I think that concept crosses over between the two.

I think I would highlight that you can use multiple instances of the sub-sections, and the steps-informal. Everything else I want to highlight is probably user guide information, not spec information.

Kris Eberlein, 30 October 2022

I agree that the basic structure of "Before the procedure," "Procedure," and "After the procedure" applies to both general and strict task. However, the strict task does impose a strict ordinality, which I tried to emphasize by using an ordered list to present the structural component of strict task.

The "Content model" section for strict task COULD use the same organizational structure as general task (a three-part definition list), where each <dd> element would contain an ordered list. I could prototype that, and we could which version of the strict task topic TC members liked best.

Kris Eberlein, 01 November 2022

Per TC discussion today, updated the structure of the strict task topic to be parallel with the general task topic.

Disposition: Completed

Introduction

This portion of the topic can contain the following structural sections:

- Prerequisites (`<prerequisites>` (24))

  Comment by Eliot Kimber
  Markup question: why use a separate cross reference when `<xmlelement>` can make a keyref directly?

  Kris Eberlein, 29 October 2022

  Because we are not using DITA 2.0 markup in this topic. DITA-OT and our stylesheets don't provide the necessary support.

  Disposition: Closed

- Contextual information (`<context>` (21))
- Sections (`<sections>`)

These sections are all optional. They can appear in any order and can occur multiple times.

Procedural instructions

This portion of the topic can contain only one of the following structural sections:

- Steps (`<steps>` (29))
- Steps informal (`<steps-informal>` (31))
- Steps unordered (`<steps-unordered>` (31))

Post-instructions

The section of the topic can contain the following structural sections:

Comment by Eliot Kimber
c/The section/This portion/

Kris Eberlein, 29 October 2022

Done.

Disposition: Completed

1. Result (`<result>` (26)
2. Troubleshooting information (`<tasktroubleshooting>` (39)
3. Example (`<example>`)
4. Post-requisites (`<postreq)>` (23)

Comment by Zoë Lawson on 30 Oct 2022
I think Post-requisite is a term coined by DITA and/or programming. It doesn't exist in a dictionary like Result or Example, etc. and could be confusing to non-native speakers. Should it be a phrase like "troubleshooting information" instead of just the name?

I'm struggling with a replacement. I think of it as "What to do next" or maybe "Following steps"

This is an interesting discussion: https://english.stackexchange.com/questions/307436/word-for-opposite-of-prerequisite-something-that-is-possible-because-of-anoth

Its what we've used for 15 years, so we can keep it, but possibly worth discussing.

Kris Eberlein, 30 October 2022

It would be easy to change "Post-requisites" in this topic (and the topic for strict task) to "What to do next" – but I have to wonder how many other places in the spec we use the term "post-requisites"? The answer is five ...

Kris Eberlein, 01 November 2022

Changed to "What to do next" in this and the next topic.

Disposition: Completed

While all of the above structural components are optional, they must occur in the outlined order. Examples and post-requisites can occur multiple times.

Example

Comment by dstevens73
See comments in taskbody. If we identify what is important to illustrate, I would be willing to write an example.

Kris Eberlein, 20 October 2022

Wonderful. I think we might need a few examples, rather than attempting to cover all usages in a single example. For example, in the <taskbody> topic, we might want to show a implementation using a general task topic to store reused task sections, especially <prereq> and <context>,

Disposition: Accepted

Comment by Zoë Lawson
Mildly unrelated, but I was working with DITA 1.3 and trying to make a large container topic with subtopics. FrameMaker and Oxygen both didn't like having a general task inside of a <dita> element.

Was I doing something wrong? or is this a bug somewhere that I'd like to see fixed in DITA 2.0? (And if I need to file this in a different way, let me know.)

Kris Eberlen, 30 October 2022

The composite (ditabase) topic that we ship for people to use out-of-the-box includes strict task. If you want a composite topic that uses general task, you'll need to build your own document-type shell. (FYI, a composite topic cannot contain both general task and strict task.)

Disposition: Closed

1.1.2 Strict task

A strict task topic answers the "How do I?" question by providing precise step-by-step instructions and other necessary information that enables the user to complete the task successfully.

Comment by Eliot Kimber
Add: It has a content model that is more restrictive than general task.

Kris Eberlein, 29 October 2022

Done.

Disposition: Completed

Purpose

The purpose of the strict task topic is to provide the reader with the following information:

- A concise statement of why, when, where, and by whom the task should be performed, as well as what the benefits are
- All the information that they need to perform the task successfully

Comment by Dawn Stevens
I don't think the purpose of general and strict tasks are really any different except for the second sentence in general's purpose, so I'm wondering about the emphasis here on the context section (bullet 1)

Kris Eberlein, 20 October 2022

Yeah, the purpose statement is weak here. I should have commented to folks at the last TC call that these topics were very much first drafts and really need work. I hope you can contribute here!

Personally, I'm a big strict task fan; I think it's a work of art. I've used general task only for specializations and for storing reusable components, when a implementation stores multiple reusable elements in a single file. I've kept waiting for a client who needed to conditionally process <preq> and <context>to show up ...

Kris Eberlein, 25 October 2022

We need to improve the purpose section. Perhaps cover the purpose in the container topic, since it does not difference between the two task types.

Eliot Kimber, 29 Oct 2022

Having differently conditional prereq or postreq elements doesn't seem like a compelling use case because the contents within a single prereq or postreq can be made conditional (and usefully grouped using <div>). I suppose the requirement must go back to pre-div days?

Kris Eberlein, 29 October 2022

I don't think that having multiple <prereq> and <context> elements in general task was a requirement because of reuse. But, it certainly is useful for that purpose.

Disposition: Completed

Content model

The strict task topic contains the following structural sections:

1. Prerequisites (`<prereq>` (24)
2. Contextual information (`<context>` (21)
3. Steps (`<steps>` (29) or `<steps-unordered>` (31))
4. Result (`<result>` (26)
5. Troubleshooting information (`<tasktroubleshooting>` (39)
6. Example (`<example>`)
7. Post-requisites (`<postreq)>` (23)

While all of the above structural components are optional, they must occur in the outlined order.

Example

The following code sample contains a strict task topic:

```
<task id="birdhousebuilding">
    <title>Building a bird house</title>
    <shortdesc>Building a birdhouse is a perfect activity
    for adults to share with their children or grandchildren.
    It can be used to teach about birds, as well as the proper use of tools.
    </shortdesc>
 <taskbody>
  <prereq>To build a sound birdhouse, you will need a complete set of tools:
  <ul><li>hand saw</li>
      <li>hammer ... </li>
  </ul></prereq>
  <context>Birdhouses provide safe locations for birds to build nests and raise their young.
      They also provide shelter during cold and rainy spells.</context>
 <steps>
   <step><cmd>Lay out the dimensions for the birdhouse elements.</cmd></step>
   <step><cmd>Cut the elements to size.</cmd></step>
   <step><cmd>Drill a 1 1/2" diameter hole for the bird entrance on the front.</cmd>
         <info>You need to look at the drawing for the correct placement of the
               hole.</info></step>
   <!--...-->
  </steps>
  <result>You now have a beautiful new birdhouse!</result>
  <postreq>Now find a good place to mount it.</postreq>
 </taskbody>
</task>
```

2 Task elements

Task topics answer "How do I?" questions, and have a well-defined structure that describes how to complete a procedure to accomplish a specific goal. Use the task topic to describe the steps of a particular task, or to provide an overview of a higher-level task. The task topic includes sections for describing the context, prerequisites, actual steps, expected results, example, and expected next steps for a task.

Comment by Stan Doherty
Remove the comma between "task," and "or".

Kris Eberlein, 31October 2022

It's been removed in the rewrite of the short description, or I most certanly would have made the change.

Disposition: Closed

Comment by dstevens73
Add tasktroubleshooting to this list, which is otherwise complete.

Kris Eberlein, 19 October 2022

This short description has not been edited. I generated HTML5 output so that I could look at short descriptions for all the parallel container topics in this portion of the spec. They currently are quite inconsistent. Some refer to the purpose of the document type or domain (as does this one), but others focus on the elements (which matches the title of these container topics.)

Before editing this topic, I want to decide on the pattern that we'll use for all such topics in the spec.

Kris Eberlein, 29 October 2022

Changed to read "Task elements provide the fundamental structure for task topics. The task topic includes sections for describing the context, prerequisites, actual steps, expected results, troubleshooting, example, and expected next steps for a task."

Disposition: Completed

Comment by Eliot Kimber
General comment about short descriptions and using "user"

The rewritten short descriptions are often of the form "allows users to …".

I think these could all be rewritten to passive voice and avoid the use of "user" entirely, i.e. from "A step is an action that a user can take to complete a task." to "A step is an action that can be taken to complete a task".

While passive voice is normally distasteful and probably painful for us Technical Writers to type, I think it's the more appropriate construction in this context.

Kris Eberlein, 30 October 2022

Marking this as "Referred. Noting that Eliot made similar comments (which I've marked as "Closed") in <steptroubleshooting>, <task>, and <tasktroubleshooting>.

Kris Eberlein, 07 November 2022

All instances of "users" in short descriptions have been changed to "people".

Disposition: Completed

2.1 <chdesc>

The `<chdesc>` element provides the content of the second cell in a choice table row. This content provides instructions for completing the step.

Comment by dstevens73
I would say the second sentence is not always true. In fact, I believe the default value for this second column is Description, which is not typically action oriented, instructions. I would suggest just deleting the second sentence.

Kris Eberlein, 19 October 2022

I'm glad that you bring this up. A few questions and points:

- I think what you refer to as the default value is not a default, but what is rendered by DITA-OT. @Robert?
- The description here matches the (current) example for `<choicetable>`, which is new for DITA 2.0. And the example might well simply represent the way that **I've** used choice tables!
- I went and looked at the choice table examples in the DITA 1.0, 1.1, 1.2, and 1.3 specs. They all use the same example, which you can see at http://docs.oasis-open.org/dita/dita/v1.3/errata02/os/complete/part3-all-inclusive/langRef/technicalContent/choicetable.html#choicetable

---

Dawn Stevens 19 October 2022

See my comments on choicetable

Kris Eberlein, 20 October 2022

I reworked the short description and reviewed it with Dawn today. A PDF of the updated topic is attached to a TC e-mail; see https://lists.oasis-open.org/archives/dita/202210/msg00042.html

Disposition: Completed

Specialization hierarchy

The `<chdesc>` element is specialized from `<stentry>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59) and the attributes defined below.

Comment by Eliot Kimber

Re the discussion of putting the scope and headers attributes into an attribute group. Looking at this topic now I agree that that would help demphasize these attributes in this (and similar) contexts.

Kris Eberlein, 29 October 2022

Already done – I completed that work after the TC meeting this week. You'll see the results in the next review

Disposition: Closed

@scope
Specifies that the current entry is a header for other table entries. The following values are valid:

col
Indicates that the current entry is a header for all cells in the column.

colgroup
Indicates that the current entry is a header for all cells in the columns that are spanned by this entry.

row
Indicates that the current entry is a header for all cells in the row.

rowgroup
Indicates that the current entry is a header for all cells in the rows that are spanned by this entry.

-dita-use-conref-target
See STUB CONTENT for more information.

@headers
Specifies which entries in the current table provide headers for this cell. The `@headers` attribute contains an unordered set of unique, space-separated tokens, each of which is an ID reference of an entry from the same table.

Example

See `<choicetable>` (14).

2.2 <chdeschd>

The `<choptionhd>` element provides a label for the second column in a choice table.

Comment by Bill Burns
Correct element name

Kris Eberlein, 25 October 2022

Done.

.
Disposition: Completed

Specialization hierarchy

The `<chdeschd>` element is specialized from `<stentry>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59) and the attributes defined below.

@scope
Specifies that the current entry is a header for other table entries. The following values are valid:

col
Indicates that the current entry is a header for all cells in the column.

colgroup
Indicates that the current entry is a header for all cells in the columns that are spanned by this entry.

row
Indicates that the current entry is a header for all cells in the row.

rowgroup
Indicates that the current entry is a header for all cells in the rows that are spanned by this entry.

-dita-use-conref-target
See STUB CONTENT for more information.

@headers
Specifies which entries in the current table provide headers for this cell. The `@headers` attribute contains an unordered set of unique, space-separated tokens, each of which is an ID reference of an entry from the same table.

Example

See `<choicetable>` (14).

2.3 <chhead>

The `<chhead>` element contains optional header-row content for a choice table.

Comment by Bill Burns
For the sake of clarity, would this be better stated header-row *elements*? The term *content* suggests PCDATA, at least to me.

Kris Eberlein, 26 October 2022

Changed to read "The `<chhead>` element contains elements that provide labels for the columns in a choice table."

Disposition: Completed

Rendering expectations

Labels provided by the `<chhead>` element override any default headings for the `<choicetable>` that are provided by stylesheets.

Specialization hierarchy

The `<chhead>` element is specialized from `<sthead>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59).

Example

See `<choicetable>` (14).

2.4 <choice>

A `<choice>` element describes a way that a user can complete the current step.

Comment by Eliot Kimber
I'm not sure the current short description needs to be changed, in that it's accurate as written, but I would have used something like:

The `<choice>` element specifies one option for completing the current step

---

Kris Eberlein, 29 October 2022

Good idea – I've changed the short description to read "A <choice> element specifies an option for completing the current step."

Disposition: Completed

Specialization hierarchy

The `<choice>` element is specialized from `<li>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59).

Example

See `<choices>` (12)

2.5 <choices>

The `<choices>` element contains a list of choices. Each choice represents a way that a user can complete the current step.

Specialization hierarchy

Comment by Eliot Kimber
Comparing this topic to `<choicetable>`, which has a Usage section, I think this topic needs a Usage section that helps distinguish `<choices>` from `<choicetable>`: Both do the same thing: provide a set of options for completing a step, differing only in one has a place to label each choice and the other does not.

If I was designing this markup today I would be tempted to have a single `<choices>` element that takes `<choice>` children where each `<choice>` has an optional `<choice-condition>` first child.

Rendering the result as a table or order list or sequence of paragraphs or definition list would then be 100% style.

Note suggesting we make that change (although we could since it's 2.0) but just observing that the current markup design is I think too layout focused and it could be different, and that difference would resolve the potential confusion between current `<choices>` and `<choicetable>`.

Or one could argue that the "table" in `<choicetable>` comes from "finite state table", not "simple table", where each choice is intended to have a unique state indicator as the first cell in a row. Not sure that was the original markup intent though.

Kris Eberlein, 29 October 2022

I suspect that the original design really was slanted towards presentation: <choices> as a list and <choicetable> as a table. Of course, at IBM, typically <choice> was flagged with an image and used for different operating system.

In my experience (and Dawn's) implementations develop business rules for how and when authors should use <choices> versus <choicetable>.

Dawn states here in the review (in the <choicetable> topic that "I typically give the following guidance to clients: use choices when there is more than one way to complete a step that leads to the same result. Use choicetable when there is more than one way to complete a step that leads to different results (to do x, then y; or if this, then that). Otherwise, writers are often inconsistent in which they choose."

I've been less semantic than that. I often suggest the folks use <choices> only when they are using it for options that will be flagged with an image or label during output processing. Or suggest that <choices> be constrained out and only <choicetable> used.

I don't think we can really add a "Usage information" section. What would it say? We don't want to get into formatting as a list, and we really cannot mandate the sort of semantic distinction that Dawn uses with her clients.

Disposition: Closed

The `<choices>` element is specialized from `<ul>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59).

Example

The following code sample shows how a user can be presented with different ways to select a server:

```
<step><cmd>Choose a server:</cmd>
  <choices>
    <choice>If you have a remote server you want to test on, type the IP address
            or hostname of the server.</choice>
    <choice>If you want to do local testing, type localhost.</choice>
  </choices>
</step>
```

Comment by Eliot Kimber
I don't think I ever realized this element even exists. I can see the appeal but if I'm writing a task with the content shown I would just use `<info>` with two paragraphs.

What presentation effect is expected for choice? Other than generating a label "Choices" and rendering the choices as an unordered list, I don't see any compelling motivation for the markup from a rendering standpoint.

This feels like the kind of "Every semantic must be tagged" markup design I was so fond of in my youth (I'm looking at you, IBM ID Doc) but that I have come to understand has diminishing returns and limited ROI in the general case....

Also, the content in the example is clearly a decision tree: IF (remote server) THEN do .. ELSE IF (local) do ...

Which would suggest the use of a choice table or some other type of decision tree, but at the very least the example is not so formal as to require or even benefit from that level of markup precision.

Is there are a more compelling example for `<choices>` as distinct from `<choicetable>`?

Kris Eberlein, 29 October 2022

You raise several points: 1) Utility of the element, 2) Presentation, and 3) Quality of the example.

Re utility: One good thing about having <choices> and <choicetable> is that it gives most implementations what they need OOB without specialization or much (if any stylesheet work).

Re presentation: Yes, some implementations render a label. And often, folks use flagging to generate an icon (or add text) for an operating system, platform, or whatever the criteria is.

Re example: Do you want to volunteer to develop one? We also could "crowd source" on dita-users, and credit the person who comes up with the best example in the spec.

---

08 November 2022

I've added a completely new example.

Disposition: Completed

2.6 <choicetable>

The `<choicetable>` element contains information about a set of options for completing a step.

Usage information

Comment by Eliot Kimber
Seem my comment on `<choices>` about more clearly distinguishing `<choices>` from `<choicetable>`. It seems to be a matter of formality in how the choices are presented.

Kris Eberlein, 31 October 2022

Marking as "Closed," since no changes to the source are required.

Disposition: Closed

A choice table provides information when there is more than one way to complete a step. It is a simple table with two columns.

Comment by Eliot Kimber
I would add "Each row represents a single choice."

I couldn't find any place that the overall content model for choice table as a sequence of row is explicitly stated.

Kris Eberlein, 29 October 2022.

Done.

Disposition: Completed

The first cell in a row provides a name for the option, and the second cell in the row provides instructions for completing the step.

Comment by Eliot Kimber
I don't think "name for the option" can be correct: it has to be more general, like "label or condition indicator for the option". I suppose, technically, if all the first-column values are distinct (which they must be or it wouldn't be a good choice table), they are "names" in the sense of distinct strings associated with a thing" but they aren't "names" in the more general sense of "the way we refer to a thing we know". Rather, they are information about why you would choose one option or another. Compare "If you have a remote server you want to test on," (`<choices>`) and "remote server" (`<choicetable>`) with column heading "Testing environment".

Kris Eberlein, 30 October 2022.

We really cannot describe the content of the <choption> element as a "label for the option," since that potentially will cause misunderstanding in regard to later content about using the header how to "provide labels for the columns".

I altered the sentence so that it read "The first cell in a row names the option".

Disposition: Completed

An optional header row can provide labels for the columns, if an author does not want to use the default labels that are provided by stylesheets.

Comment by dstevens73
I think I am struggling with this review because I feel like the audience has changed. It has been said over and over that the spec is not for the writers, people using DITA, but for implementers. But much of this content seems very directed to how the content should be written. I don't necessarily disagree with the direction, but in this case, I typically give the following guidance to clients: use choices when there is more than one way to complete a step that leads to the same result. Use choicetable when there is more than one way to complete a step that leads to different results (to do x, then y; or if this, then that). Otherwise, writers are often inconsistent in which they choose.

Kris Eberlein, 19 October 2022

Yes, implementers are the primary audience for the spec, especially the base edition, which sets out core architectural principles and normative rules. But we've always known that short descriptions are surfaced in a lot of authoring tools as guidance to authors, and so want to keep authors in mind as we craft short descriptions and usage information.

Another consideration there is that the element reference topics in the Technical Content edition, especially those for task elements, have also had the most guidance/tutorial focus. We've removed some (but not all) of that.

The choice table content for the 1.x versions of the spec was … not very good nor very specific. I might well have erred in writing content that reflected how I've always used – and seen choice tables used. I set up a call with Robert (starting quite soon!) to talk this over.

Kris Eberlein, 20 October 2022

I reworked this topic and reviewed it with Dawn today. I tweaked the "Usage information" and added another example. A PDF of the updated topic is attached to a TC e-mail; see https://lists.oasis-open.org/archives/dita/202210/msg00042.html

Disposition: Completed

Specialization hierarchy

The `<choicetable>` element is specialized from `<simpletable>`. It is defined in the task module.

Comment by Zoë Lawson on 30 Oct 2022
If I am remembering correctly, in DITA 2.0, we added the ability to span rows and columns to simple tables. Since choicetables are specialized from simpletable, are there any interesting consequences of potentially being able to span rows/columns in a choice table?

These questions came to mind when I was reading the @scope attribute description on all the choice table elements.

Kris Eberlein, 30 October 2022

We did not add the @colspan and @rowspan attributes to <choicetable, so no spanning in choice tables.

Disposition: Closed

Attributes

The following attributes are available on this element: universal attributes (59), display attributes (64), and simpletable attributes (64).

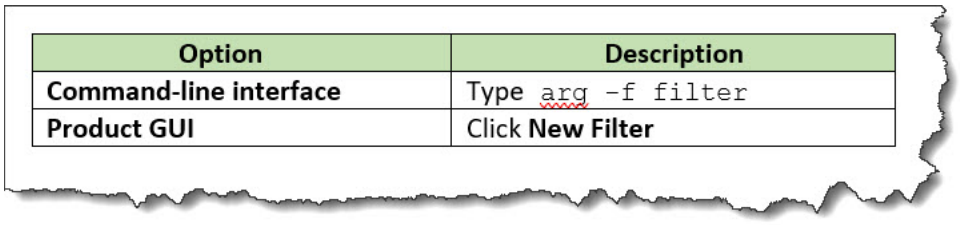
For this element, the `@keycol` attribute has a default value of "1".

Example

The following code sample contains a choice table that contains a header row:

```
<step>
  <cmd>Create a new filter:</cmd>
  <choicetable>
    <chhead>
      <choptionhd>Option</choptionhd>
      <chdeschd>Description</chdeschd>
    </chhead>
    <chrow>
      <choption>Command-line interface</choption>
      <chdesc>Type <codeph>arg -f filter</codeph></chdesc>
    </chrow>
    <chrow>
      <choption>Product GUI</choption>
      <chdesc>Click <uicontrol>New Filter</uicontrol></chdesc>
    </chrow>
  </choicetable>
</step>
```

The choice table might be rendered in the following way:



Option	Description
Command-line interface	Type `arg -f filter`
Product GUI	Click **New Filter**

Comment by Eliot Kimber

Observation that does not require action:

Comparing this example with the new example in the updated version Kris mentions above, it shows that there are at least two semantically distinct ways that `<choicetable>` can be usefully used:

- Alternative ways of accomplishing the same result (CLI or UI)
- Options that have different material results: Attempt cancel vs. force cancel.

Which in turn are distinct from making a choice where the task could fork into two separate sequences of actions (which you'd normally do using substeps or subtasks) (i.e., in the context of a procedure that reflects a flow chart or decision tree rather than a simple linear sequence of steps).

The `<choices>` and `<choicetable>` markup doesn't make a clear distinction between these different use cases. I'm not sure it needs to or even can without more specialized markup, but I can see that this could cause confusion for authors who do not have clear guideance from their local Information Architect about how to use these elements in their tasks.

It also calls into question the general utility of these specializations given you could get the same visual result from their specialization bases with no more information design thought effort.

It feels like either these elements need another level of specialization to reflect the more specific use cases (alternative ways to perform the step vs alternative ways to respond to the state resulting from earlier steps vs action options available at this point in the task.

I think for myself as a technical writer I would tend to try to rewrite the tasks to avoid the need for all but the "alternative ways to perform the step" use (i.e., CLI vs UI).

Disposition: Closed

2.7 <choption>

The `<choption>` element contains the content of the first cell in a choice table row. This content names the option that the user can take to complete the step.

Comment by Eliot Kimber
See my comment on `<choicetable>`: I don't think "names the option" is correct--should be "labels or describes the option" or close to. That is, you would never use the name "Option 1", you would use the label "Local server": both are distinct values (within the choice table), but one is an arbitrary name and one is a descriptive label.

Having said that, I suppose you might use "Option 1" if you write the command or info content such that it references the option name:

Having set the framitiz to "on" you may do Option 1 or Option 2 as shown in the choice table below.

So I guess it depends on whether you are writing the choice table as simple lookup table of options where the option labels are arbitrary or a state table where the option label reflects a specific state ("local server", "remote server").

Kris Eberlein, 31 October 2022

I agree that "names the option" is imprecise, but I think that it's the best that we can do. Shifting to "labels or describes the option" comes into conflict with 1) how we describe the <chdesc> element, and 2) what the <chhead> row provides. We're stuck with the element names – <choption> and <chdesc> – and also do not want to deviate too far from how these elements have been defined in the four versions of the DITA 1.x specifications. (I think the element names were poorly chosen, and our current rigor with short descriptions is highlighting that …)

Nonetheless, I'm setting the comment disposition to "Referred," so that we can discuss it at a TC meeting. Maybe someone else can come up with an acceptable way to describe what the <choption> element does.

Kris Eberlein, 07 November 2022

Changed the shortdesc to read "The <choption> element contains the content of the first cell in a choice table row. This content labels the option that people can take to complete the step."

Disposition: Completed

Specialization hierarchy

The `<choption>` element is specialized from `<stentry>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59) and the attributes defined below.

@scope
Specifies that the current entry is a header for other table entries. The following values are valid:

col
Indicates that the current entry is a header for all cells in the column.

colgroup
Indicates that the current entry is a header for all cells in the columns that are spanned by this entry.

row
Indicates that the current entry is a header for all cells in the row.

rowgroup
Indicates that the current entry is a header for all cells in the rows that are spanned by this entry.

-dita-use-conref-target
See STUB CONTENT for more information.

@headers
Specifies which entries in the current table provide headers for this cell. The `@headers` attribute contains an unordered set of unique, space-separated tokens, each of which is an ID reference of an entry from the same table.

Example

See `<choicetable>` (14).

2.8 <choptionhd>

The `<choptionhd>` element provides a label for the first column in a choice table.

Specialization hierarchy

The `<choptionhd>` element is specialized from `<stentry>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59) and the attributes defined below.

@scope
Specifies that the current entry is a header for other table entries. The following values are valid:

col
Indicates that the current entry is a header for all cells in the column.

colgroup
Indicates that the current entry is a header for all cells in the columns that are spanned by this entry.

row
Indicates that the current entry is a header for all cells in the row.

rowgroup
Indicates that the current entry is a header for all cells in the rows that are spanned by this entry.

-dita-use-conref-target
See STUB CONTENT for more information.

@headers
Specifies which entries in the current table provide headers for this cell. The `@headers` attribute contains an unordered set of unique, space-separated tokens, each of which is an ID reference of an entry from the same table.

Example

See `<choicetable>` (14).

2.9 <chrow>

The `<chrow>` element represents a row in a choice table. It contains a pair of elements: `<choption>` and `<chdesc>`.

Comment by Eliot Kimber
I might say "represents a single choice in a choice table".
Disposition: Closed

Specialization hierarchy

The `<strow>` element is specialized from `<strow>`. It is defined in the task module.

Comment by Bill Burns
Correct element name to `<chrow>`.

Kris Eberlein, 26 October 2022

Done.

Disposition: Completed

Attributes

The following attributes are available on this element: universal attributes (59).

Example

See `<choicetable>` (14).

2.10 <cmd>

The `<cmd>` element contains a command. A command provides instruction on how to complete a step.

Comment by Eliot Kimber
Could collapse the two sentences: "… element provides instruction on how to complete a step."

However, I think the design intent for `<cmd>` is more limited: specifies the direct action to take to complete the step. Of course, that presumes a particular writing style, which may not be appropriate for the spec (for example, at ServiceNow we have Schematrons rule that requires `<xml>` to be exactly one sentence and meet other strict requirements).

The `<step>` element itself provides the "instruction on how to complete a step", because it contains the command and any additional information needed to support the completion of the step (`<info>`, `<stepxmp>`, etc.)

Kris Eberlein, 29 October 2022

The <cmd> shortdesc has been revised; it now uses natural language and reads "A command specifies the action to take to complete a step."

The <step> shortdesc has also been revised; it now reads "A step is an action that a user can take to complete a task. It can contain additional information about the step, such as an example, result, or troubleshooting guidance."

Disposition: Completed

Specialization hierarchy

The `<cmd>` element is specialized from `<ph>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59) and `@keyref` (69).

Example

In the following code sample, the `<cmd>` element provides clear, active-voice instruction for how to complete a step:

```
<step>
  <cmd>Specify the configuration parameters.</cmd>
</step>
```

2.11 <context>

The `<context>` element contains contextual information. Contextual information is background information that helps users understand the purpose of the task and what they will gain by completing it.

Comment by Bill Burns
This could be a single sentence: The `<context>` element contains background information that helps users understand the purpose of the task and what they will gain by completing it.

Kris Eberlein, 26 October 2022

Even simpler – It's already been rewritten as "Contextual information is background information that helps users understand the purpose of the task and what they will gain by completing it." Part of our commitment to using natural language in short descriptions when appropriate.

Disposition: Closed

Specialization hierarchy

The `<context>` element is specialized from `<section>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59).

Example

An author uses the following markup to provide users with more contextual information than is appropriate for a short description. Style sheets might generate a label, for example, “About this procedure”, to indicate clearly that the information provided is background information.

Comment by dstevens73
postreq and prereq put this information about the possibility of labels in a usage section.

Kris Eberlein, 20 October 2022

I've added a "Rendering expectations" section here that states "Implementations might want to consider having their stylesheets render a label for this element.", per our TC call this week.

I wonder if I should delete the 2nd sentence here – or whether we want to show a possible rendering since we mention that style sheets might render a label. Any thoughts?

Kris Eberlein, 21 October 2022

I've deleted the info here in the example about style sheets generating an example. Robert and I think that, at most, we should have one place where we mentioned task labels and show a screen capture. I think that should be not here in the element reference topics, but in the overview topic.

WEK: 29 Oct 2022

I agree with removing the mention of label rendering in this example.

Disposition: Completed

```
<task id="Generating-stub-files" xml:lang="en-us">
  <title>Generating stub files</title>
  <shortdesc>You can use Task Modeler to generate stub files. Stub files are
             DITA files that contain only a title.</shortdesc>
  <taskbody>
    <context>As you perform this procedure, you can select the conventions that
            you want to use for file names.
    </context>
   <!-- ... -->
  </taskbody>
</task>
```

Comment by dstevens73
most of my clients are moving away from gerund titles largely for localization reasons. Most now use imperative or infinitive. Do we care? If so, we will need to look at all the examples in this section.

Kris Eberlein, 19 October 2022

I don't know if it matters. Our examples need to be exemplars for markup, not writing style. (Although I do cringe when I see poorly-written stuff in our spec examples.)

Out of curiosity, how does using the imperative or infinitive form of a verb aid localization? I'm most steeped in "IBM Style Guide" and "The Global English Style Guide". "IBM Style Guide" uses gerund titles in all their examples … "The Global English Style Guide" does not warn against gerund verb forms ...

Kris Eberlein, 21 October 2022

Marking this one as closed. I think that writing styles for titles is a writing practice that will differ across implementations.

Eliot Kimber, 29 Oct 2022

For what it's worth, I think I would write "Generate stub files". I might have a separate concept topic "About generating stub files" or "Understanding stub file generation".

But looking at current ServiceNow documentation, I find a mix: https://docs.servicenow.com/bundle/tokyo-employee-service-management/page/product/employee-center/concept/setup-emp-center.html. The top-level task is "Setting up Employee Center" but the individual tasks do not use gerunds. Hmm.

Disposition: Closed

2.12 <info>

The `<info>` element contains additional information about the step.

Specialization hierarchy

The `<info>` element is specialized from `<div>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59).

Example

In the following code sample, the `<info>` element provides additional information about the ways that the step can be performed:

```
<step>
  <cmd>Specify the configuration parameters.</cmd>
  <info>The configuration parameters can be specified from either the command line or
        the product GUI.
  </info>
</step>
```

Comment by dstevens73
I realize I'm probably being way too down in the weeds, but the example is passive in what should be an active presentation. Better: Use either the command line or the product GUI.

Done.

Disposition: Completed

2.13 <postreq>

The `<postreq>` element contains post-requisites. Post-requisites are steps or tasks that the users need to perform after completing the current task.

Comment by Bill Burns
Suggest "The `<postreq>` element contains post-requisites—steps or tasks that the users need to perform after completing the current task."

Kris Eberlein, 26 October 2022

Already changed to "Post-requisites are steps or tasks that the users need to perform after completing the current task."

Eliot Kimber, 29 Oct 2022

Could we avoid "users" by making it passive: Post-requisites are steps or tasks that need to be performed after completing the current task."?

Kris Eberlein, 29 October 2022

Users versus a passive construction – what a lovely set of choices! This is a duplicate comment.

Disposition: Closed

Usage information

Implementations might want to consider having their stylesheets render a label for this section.

Specialization hierarchy

The `<postreq>` element is specialized from `<section>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59).

Example

The following code sample shows how a user might be directed to notify a test proctor after completing a test.

```
<steps>
<!-- ... -->
  <step>
    <cmd>Click <uicontrol>Done</uicontrol> to complete the test.</cmd>
  </step>
</steps>
<postreq>Notify the proctor upon completing this self-test.</postreq>
```

Comment by dstevens73
context and postreq have a comment in the example about how a label might be generated.

Yes, that was in the review draft. I've removed that; I think the one place to really mention this is in the overview topics about task. There we can explain that these elements do not allow titles, and do task labels might be a good idea.

Disposition: Closed

2.14 <prereq>

The `<prereq>` contains prerequisites. Prerequisites are things that users need to know or preliminary tasks that users need to perform before starting the current task.

Comment by Bill Burns
Suggested change: "The `<prereq>` contains prerequisites—things that users need to know or preliminary tasks that users need to perform before starting the current task."

Kris Eberlein, 26 October 2022

Already changed to "Prerequisites are things that users need to know or preliminary tasks that users need to perform before starting the current task."

Disposition: Closed

Usage information

Implementations might want to consider having their stylesheets render a label for this section.

Rendering expectations

Processors might render prerequisite links from the related-links section together with the content of the `<prereq>` element.

Comment by Eliot Kimber
I'm not sure I would ever recommend this as there's no way, a-priori, to know if a given related link is for a prereq or postreq step or just a related task for whatever reason.

Kris Eberlein, 29 October 2022

I've never recommended or done this, either. This is verbiage (modified slightly over time) that has been present since early, pre-OASIS days. You can, of course, use @role and @otherrole to indicate that a link is a prerequisite link.

DITA-OT implements some support for this, if my memory is correct. Let's ask the TC whether we should keep or remove this statement.

Kris Eberlein, 07 November 2022

Per TC dscussion, removed the statement.

Disposition: Completed

Specialization hierarchy

The `<prereq>` element is specialized from `<section>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59).

Example

The following code sample is from a topic that explains how to create an SQLJ file. A prerequisite is to log into the SQLJ server.

```
<task id="sqlj">
 <title>Creating an SQLJ file</title>
 <taskbody>
    <prereq>Before creating a new SQLJ file, you must
  log in to the SQLJ server.
    </prereq>
    <!-- ... -->
  </taskbody>
</task>
```

Style sheets might generate a label, for example, “Before you begin”, to indicate clearly that the prerequisite task needs to be performed before embarking on the procedure.

Comment by Eliot Kimber
Would it be appropriate to make the "log in to the SQLJ server" an explicit hyperlink? I.e.:

```
<xref keyref="task.login-sqlj">log in to the SQLJ server</xref>
```

Kris Eberlein, 29 October 2022

Whether or not that would be appropriate would depend on the guidelines for any particular implementation. Some implementation might allow (or maybe encourage that), others might not permit that (especially if the topic for "Logging into the SQLJ server" was nearby – or such a simple task that it does not need documentation). Still other implementation would want to use reltable links.

In any case, I don't think that additional markup improves the example.

Disposition: Closed

2.15 <result>

The `<result>` element describes the expected outcome for the task as a whole.

Specialization hierarchy

The `<result>` element is specialized from `<section>`. It is defined in the task module.

Comment by dstevens73
Should also comment on generating a default title as done in context, postreq, prereq.

Kris Eberlein, 20 October 2022

I have added a "rendering expectations" section, as we discussed at the TC call this week.

Disposition: Completed

Attributes

The following attributes are available on this element: universal attributes (59).

Example

In the following code sample, the author clearly communicates the expected result of successfully completing the task:

```
<task id="sqlj">
  <title>Creating an SQLJ file</title>
  <taskbody>
    <!-- ... -->
    <result>The "File Created" window is displayed, and the SQLJ
            file is successfully created.
    </result>
  </taskbody>
</task>
```

Comment by Eliot Kimber
Super nit picky, but I would reverse the order of the two results: SQLJ file is created and "File created" window is displayed.

Kris Eberlein, 29 October 2022

Done.

Disposition: Completed

Comment by Zoë Lawson on 30 Oct 2022
Silly nit. Should "File Created" be wrapped in <wintitle> instead of using quotes? Or is that outside of the Technical Content?

Kris Eberlein, 30 October 2022

Not a nit at all. Some of these legacy examples tended to have minimal markup. Done.

Disposition: Completed

2.16 <step>

The `<step>` element contains a step. A step represents an action that a user can take to complete a task. It can also contain additional information about the step, such as an example, result, or troubleshooting guidance.

Comment by Bill Burns
Suggested change: "The `<step>` element represents an action that a user can take to complete a task."

Kris Eberlein, 26 October 2022

Already changed to "A step is an action that a user can take to complete a task. It can also contain additional information about the step, such as an example, result, or troubleshooting guidance."

Eliot Kimber, 29 Oct 2022

I would remove " also" as the step contains everything. The command is the thing for which "can also" would be appropriate.

Kris Eberlein, 29 October 2022

Done.

Disposition: Completed

Rendering expectations

Comment by Kristen J Eberlein on 07 October 2022

I think we need to say something about what processors should do with the effective value of the `@importance` attribute. I've seen processors not render anything -- and it causes problems.

I have add the following content:

When the @importance attribute is specified on the element, it indicates whether the step is optional or required. Implementations might want to consider having their stylesheets render a applicable label when @importance is specified on <step>,

Disposition: Completed

Comment by Zoë Lawson on 30 Oct 2022
I think this is just me having a duh moment, but I'm curious about some implementation bits for all the various components of elements that can be in the step. In the dark ages of DITA 1.0 or 1.1, I'm not sure what the expectation was for rendering <info> etc. Once upon a time, I had to specifically put a <p> into <info> or <stepresult> to have separate blocks. I think that's now happening by default? Do we need to specify that this is expected to be a block or inline? Are we assuming this is inferred from the specialization hierarchy?

Kris Eberlein, 30 October 2022

This really depends on the implementation and its stylesheets. IBM stylesheets rendered the content of <info> inline following the content of <cmd>. By default, the DITA-OT renders <info> as a block. This sort of stuff is why so many of us wanted to remove the "Block/inline (presentation)" column from the "Elements-by-elements" topic.

Disposition: Closed

Specialization hierarchy

The `<step>` element is specialized from `<li>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59).

For this element, the `@importance` attribute is limited to the values "optional", "required", or -dita-use-conref-target.

Example

The following code sample shows almost all the elements that the `<step>` element can contain. It does not contain the `<stepxmp>` element.

Comment by dstevens73
nor tutorialinfo nor choicetable. Nor note or hazardstatement before the cmd. Question the "almost all". Perhaps a different introduction to the example?

Kris Eberlein, 20 October 2022

Changed the intro paragraph to read: "The following code sample shows many of the elements that the element can contain:

Disposition: Completed

```
<step>
  <cmd>Specify the configuration parameters.</cmd>
  <info>The configuration parameters can be specified from either the command line or
        the product GUI.</info>
  <choices>
    <choice>From a command prompt, type config -l parameter</choice>
    <choice>Click New Configuration Parameters.</choice>
  </choices>
  <stepresult>You receive a 'Configuration successful message".</stepresult>
  <steptroubleshooting>If you do not receive a 'Configuration successful message,"
                       retry the configuration operation.</steptroubleshooting>
</step>
```

2.17 <stepresult>

The `<stepresult>` element provides information about the expected outcome of a step.

Specialization hierarchy

The `<stepresult>` element is specialized from `<div>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59).

Example

In the following example, the content of the `<stepresult>` element enables the user to ascertain whether they have completed the step correctly:

```
<step>
  <cmd>Specify the configuration parameters.</cmd>
  <info>The configuration parameters can be specified from either the command line or
        the product GUI.</info>
```

Comment by
dstevens73
if we change from my previous comment, should change here too

Kris Eberlein, 20 October 2022

Done.

Disposition: Completed

```
  <choices>
    <choice>From a command prompt, type config -l parameter</choice>
```

Comment by
dstevens73
I would probably put <userinput>config -1 parameter</userinput>. Regardless, missing the period at the end.

Kris Eberlein, 19 October 2022

But then does the reader risks thinking that they should type the period as part of the command? Regarding the markup, the question is whether adding additional inline elements (albeit certainly best semantic practice), makes it harder to read the code sample?

Added extra tagging: <codeph> and <uicontrol>.

Disposition: Completed

```
    <choice>Click New Configuration Parameters.</choice>
  </choices>
  <stepresult>You receive a 'Configuration successful message".
```

Comment by
dstevens73
I would punctuate 'Configuration successful' message. regardless of the position of the quote, the second should match the first. I would actually probably do <msgph> or <systemoutput> around Configuration successful.

Kris Eberlein, 19 October 2022

I corrected the type and placement of the quotation marks. Regarding the markup, the question is whether adding additional inline elements (albeit certainly best semantic practice), makes it harder to read the code sample?

Added extra tagging: <systemoutput>

Disposition: Completed

```
  </stepresult>
</step>
```

2.18 <steps>

The `<steps>` element contains a set of steps. Steps are a series of actions that are conducted in a specific order and manner.

Comment by Bill Burns
Suggestion: "The `<steps>` element contains a series of actions that are conducted in a specific order and manner."

Kris Eberlein, 26 October 2022

Already changed to "Steps are a series of actions that are conducted in a specific order and manner."

Eliot Kimber, 29 Oct 2022

Is "performed" more appropriate than "conducted". I think this content uses "perform" or "performed" throughout. "Conducted" struck me as an odd and unexpected construction in this context.

Kris Eberlein, 29 Oct 2022

Changed to "performed".

Disposition: Completed

Comment by Stan Doherty
Would <steps> ever be rendered as something other than a numbered list in the real world?

Kris Eberlein, 31 Oct 2022

I don't think so; ordinality is critical to the very nature of steps. This morning I added the following content to the "Rendering expectations" section for this topic: "Steps that contain only a single step should be rendered as a paragraph. Steps that contain two or more steps should be rendered as an ordered list." (This content had been incorrectly placed in the "Formatting expectations" appendix.

Disposition: Closed

Specialization hierarchy

The `<steps>` element is specialized from `<ol>`. It is defined in the task module.

Comment by dstevens73
Should also have Usage section to comment on generating a default title as done in context, postreq, prereq.

Kris Eberlein, 20 October 2022

Added a "Rendering expectations" section, as discussed at this week's TC meeting.

Disposition: Completed

Attributes

The following attributes are available on this element: universal attributes (59).

Example

The following code sample shows a simple task topic with two steps:

```
<task id="sqlj">
<title>Creating an SQLJ file</title>
 <taskbody>
 <context>Once you have set up SQLJ, you can create a new SQLJ file.</context>
  <steps>
   <step>
    <cmd>In a text editor, create a new file.</cmd>
   </step>
   <step>
    <cmd>Enter the first query statement.</cmd>
   </step>
  </steps>
 </taskbody>
</task>
```

2.19 <steps-informal>

The `<steps-informal>` element contains a set of informal steps. Informal steps do not follow a strict content model; a paragraph might describe more than one step, or a paragraph might combine procedural information along with other information.

Specialization hierarchy

The `<steps-informal>` element is specialized from `<section>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59).

Example

The following code sample shows how an author provided informal information about how to grow a flower from seed:

```
<task id="growing-flower>
  <title>Growing a flower from seed</title>
  <taskbody>
    <steps-informal>
      <p>Put the soil in the container any old way. It doesn't really
         matter how you do it as long as it is at least 12 cm deep. Once
         the soil is in place, plant the seeds, water appropriately and
         wait.</p>
    </steps-informal>
  </taskbody>
</task>
```

2.20 <steps-unordered>

The `<steps-unordered>` element contains a set of unordered steps. Unordered steps are steps in which the order of the steps to be performed might vary from one situation to another.

Comment by Bill Burns
Suggeston: "The `<steps-unordered>` element contains a set of steps in which the order of the steps to be performed might vary from one situation to another."

Kris Eberlein, 26 October 2022

Already changed to "Unordered steps are steps in which the order of the steps to be performed might vary from one situation to another."

Disposition: Closed

Specialization hierarchy

The `<steps-unordered>` element is specialized from `<ul>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59).

Example

The following code sample shows how an author provided information about the tasks that need to be performed before leaving on a vacation. While each step involves a single item, the steps can be performed in any order.

```
<task id="prep-for-trip">
  <title>Preparing for a trip</title>
  <taskbody>
    <steps-unordered>
      <step>
        <cmd>Arrange for a pet sitter</cmd>
      </step>
      <step>
        <cmd>Do laundry</cmd>
      </step>
      <step>
        <cmd>Buy a plane ticket</cmd>
      </step>
    </steps-unordered>
  </taskbody>
</task>
```

2.21 <stepsection>

The `<stepsection>` element contains expository text that might be rendered before a step.

Comment by Zoë Lawson on 30 Oct 2022
Before a step, or between steps?

I didn't know that this element existed, and I've been confused by it. I think of it as sort of a catch-all, if you have to put text between steps that doesn't fit into any of the <step> options. It also feels like an element outside of structured authoring practices, but it could be useful when migrating not overly structured stuff, similar to comments you made earlier on General task.

I agree with Dawn's emphasis about not changing the numbering. Do we care about alignment or is that getting too far into the weeds/implementation choices?

Kris Eberlein, 30 October 2022

Re your first point: I use this occasionally. The revision example for this topic for a solid use case, so please check it out when you get the revised PDF.

Re alignment – That is strictly formatting that is entirely up to the individual implementation.

Disposition: Closed

Rendering expectations

001 (44) Processors which render the content of `<stepsection>` elements among the `<step>` elements **MUST NOT** number the `<stepsection>` elements.

Comment by dstevens73
Is it a rendering expectation that the numbering of the step elements is not interrupted – ie doesn't start over? This element is a source of confusion in my clients often who see it as a way to provide more than one set of steps in a single topic; ie when they have multiple ways to complete a task. they are surprised when the steps don't renumber after a stepsection. The example indicates the numbering expectation, but I wonder if it would help to provide more detail here.

Kris Eberlein, 19 October 2022

@Dawn, what additional detail would you want here?

Kris Eberlein, 20 October 2022

Just off a call with Dawn. What she would find helpful is if the spec stated that <stepsection> element do not affect the numbering of steps. That is, that the numbering of the steps is contiguous; it is not interrupted/restarted by the presence of a <stepsection> element.

Kris Eberlein, 21 October 2022

Added a "Usage information" section with the following content: "The <stepsection> element can be used to break up lengthy procedures by providing labels for groups of steps. Note that introducing <stepsection> elements will not affect the contiguous numbering of the steps." @Robert?

Disposition: Completed

Specialization hierarchy

The `<stepsection>` element is specialized from `<li>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59).

Example

The following code sample shows how a `<stepsection>` element might be used:

```
<steps>
  <step>
    <cmd>Get out a bowl.</cmd>
  </step>
  <stepsection>The next two steps are very important!</stepsection>
  <step>
    <cmd>Put on safety gloves.</cmd>
  </step>
  <step>
    <cmd>Put on goggles.</cmd>
  </step>
  <step>
```

```
     <cmd>Pour milk and cereal into the bowl.</cmd>
   </step>
</steps>
```

The code sample above typically would be rendered with "Get out a bowl" as step number one, "Put on safety gloves" as step number two, and "The next two steps are very important!" as an unnumbered item in between the first two items.

Comment by dstevens73
probably the most common use of stepsection I see is for the (unnecessary) "To do task, complete the following steps" sentence. I'm wondering if that practice should be part of an example?

Kris Eberlein, 19 October 2022

@Dawn, are you suggesting that we include "To do task, complete the following steps" in a <stepsection>? I don't think we want to show BAD tagging practices ...

Talked with Dawn, she doesn't advocate including "To do task, complete the following steps" in a <stepsection>. But I'm wondering if an different example AND an example of possible rendering might be useful here.

Kris Eberlein, 21 October 2022

Added a new example, which includes a possible rendering which clearly shows that step numbering is not affected by the <stepsection> elements.

Disposition: Completed

2.22 <steptroubleshooting>

The `<steptroubleshooting>` element contains information that is designed to help remedy the situation when a step does not complete as expected.

Comment by Eliot Kimber
Remove "that is designed ". Or perhaps completely replace with

contains information for responding when a step does not complete as expected.

"remedy the situation" presumes that remediation is within the reader's power, which it may not be. For example, the troubleshooting may consist entirely of gather logs and messages and reporting it to support.

Kris Eberlen, 30 October 2022

I changed this to "Step troubleshoooting is information that is intended to help users respond to the situation if a step does not complete as expected."

Disposition: Completed

Usage information

Tip Do not use `<note type="trouble">` inside of the `<steptroubleshooting>` element.

Comment by dstevens73
Seems like an explanation of why not would be useful.

Kris Eberlein, 19 October 2022

This is content that JoAnn and Bob Thomas insisted on adding in DITA 1.3. In 13, the full note read as follows:

"Do not use <note type="trouble"> inside of <steptroubleshooting> because its meaning there would be ambiguous."

Robert questioned what "would be ambiguous" meant, so I removed that. Can you suggest different wording? I think the original intent was that having such a note in the <steptroubleshooting> element was inherently ambiguous – that it would be unclear whether the note applied to the step or the actual <steptroubleshooting> element.

Kris Eberlein, 20 October 2022

I talked with Dawn today. We both agree that this note should be removed. It's appropriate for guidelines for an implementation, but not the spec. And I removed it from this topic and the <tasktroubleshooting> topic.

Disposition: Completed

Specialization hierarchy

The `<steptroubleshooting>` element is specialized from `<div>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59).

Example

The following code sample shows how the `<steptroubleshooting>` element specifies the troubleshooting actions that a user can take if the step does not complete as they expected:

```
<step>
  <cmd>Log in to the system</cmd>
  <stepresult>
    <p>The <wintitle>Welcome</wintitle> screen appears.</p>
  </stepresult>
  <steptroubleshooting>
    <p>If the <wintitle>Welcome</wintitle> screen does not
      appear, try one or more of the following actions:</p>
      <ul>
        <li>Verify that the user name was entered correctly</li>
        <li>Verify that the password was entered correctly</li>
        <li>Confirm that the maintenance contract is still
          active</li>
      </ul>
  </steptroubleshooting>
</step>
```

2.23 <stepxmp>

The `<stepxmp>` element contains an example that illustrates how a step is completed. The example can be a few words, an image, or a paragraph.

Comment by dstevens73

or a code block. Not sure the last sentence is necessary.

Kris Eberlein, 19 2022

A code block for tasks that are performed from the command line? (I often see people using <info> for that purpose …) FWIW, this short description has been in place since DITA 1.0.

Kris Eberlein, 20 October 2022

Talked to Dawn today. We discussed the short description – not sure if the best solution here is to code block to the second sentence or remove the second sentence. FWIW, we both agree this element is inconsistently used, and that each implementation needs to nail down how they will use it.

Kris Eberlein, 21 October 2022

Changed to read "The example might be text-based, an image, a code sample, a link to a video, or some other representation.

Eliot Kimber, 29 Oct 2022

I often use code blocks in `<stepxmp>` when documenting tasks that involve command-line actions.

Disposition: Completed

Specialization hierarchy

The `<stepxmp>` element is specialized from `<div>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59).

Example

Comment by Kristen James Eberlein

Kris Eberlein, 20 October 2022

We need a different example here, one that better illustrates ways that the element can be used.

Kris Eberlein, 21 October 2022

I've added a real life example.

Disposition: Completed

In the following code sample, the `<stepxmp>` element contains an example of what a user might type as a name for a widget:

```
<step>
 <cmd>Type a name for the widget.</cmd>
 <stepxmp>For example, <userinput>mywidget</userinput></stepxmp>
</step>
```

2.24 <task>

The `<task>` element is the top-level element for a task topic. Task topics provide instructions that enable a user to perform a task.

Comment by Eliot Kimber
In addition to using passive voice per my general comment on short descriptions, I would say "Task topics provide instructions that guide the performance of a task.".

The task, by itself, is not an enabler, in that it does not grant the authority, ability, or power to perform the task--it only defines the procedure by which the task should be performed.

Kris Eberlein, 30 October 2022

Changed to read "Task topics provide the instructions that guide a user to perform a task."

Disposition: Completed

Usage information

The OASIS DITA Technical Committee distributes two document-type shells for task topics: general task and strict task.

General task
Has a more relaxed content model. It allows `<section>` and `<steps-informal>` inside of the task body; it also allows multiple instances and varying order for the elements that make up the task body.

(Strict) task
Maintains a strict order and cardinality for elements within the `<taskbody>` content model. The strict task is implemented with a constraint module.

Specialization hierarchy

The `<task>` element is specialized from `<topic>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59) and architectural attributes (63).

For this element, the `@id` attribute is required.

Example

The following code sample shows that `<task>` is the topic-level element for a task topic:

```
<task id="learn-dita">
  <title>Learning DITA</title>
  <!-- ... -->
</task>
```

2.25 <taskbody>

The `<taskbody>` element contains the body of a task topic. The task body can include prerequisites, contextual information, steps, results, an example, troubleshooting information, and post-requisites. General task topics can also contain generic sections.

Comment by Bill Burns
Suggestion: "The `<taskbody>` element contains the content of a task topic."

Kris Eberlein, 26 October 2022

We want to keep this parallel with how we describes the <body> element in the base. Also, the content model of <taskbody> is quite significant.

Disposition: Rejected

Usage information

The content model for the task topic varies depending on whether the strict task or general task document-type shell is used.

Specialization hierarchy

The `<taskbody>` element is specialized from `<body>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59).

Examples

This section contains examples of the `<taskbody>` element in both (strict) task and general task topics.

Figure 1: Strict task topic

The following code sample shows how the `<taskbody>`element contains the main building blocks of a strict task topic:

```
<task id="Generating-stub-files" xml:lang="en-us">
  <title>Generating stub files</title>
  <shortdesc>You can use Task Modeler to generate stub files. Stub files are DITA files
             that contain only a title.</shortdesc>
  <taskbody>
    <prereq>You must have created a DITA map in Task Modeler.</prereq>
    <context>As you perform this procedure, you can select the conventions that you want to
             use for file names.</context>
    <steps>
      <!-- ... -->
    </steps>
    <result>In the File Manager view, you can see the file names and paths of the DITA
            topics.</result>
    <tasktroubleshooting>If you cannot see the file name and paths of the DITA topics, refresh
            the view.</tasktroubleshooting>
    <example> <! -- ... --> </example>
    <postreq>You now can create a relationship table to define links between the topics in
             your DITA map.</postreq>
  </taskbody>
</task>
```

Comment by Eliot Kimber
The example has awkward line breaks that could be resolves to make the example code easier to read.

Kris Eberlein, 29 October 2022

@Eliot, this looks fine in the review PDF and in Jarno's online version: https://dita-lang.org/dita-techcomm/langRef/technicalContent/taskbody.html. I think Content Fusion's rendering is the problem here.

Disposition: Closed

In a strict task topic, while the child elements of `<taskbody>` are all optional, they can only occur once and must appear in a specific order.

Figure 2: General task topic

The following code sample shows ...

Comment by dstevens73
what are the important things to include in the general task topic example? steps-informal? different order of prereq and context (one of the most common reasons my clients choose general)? section? (I don't think I have any clients adding sections)

Kris Eberlein, 19 October

I think the important things is to show **something** that is different from strict task. I have not had any clients using general task, although I've certainly used it as a specialization base for specialized task topics, such as the one for Applied Materials.

Can you contribute an example?

Eliot Kimber, 29 Oct 2022

One easy example might be a task that has a "Safety Considerations" section before `<prereq>`, which seems like an obvious and compelling reason to not use strict task.

Disposition: Accepted

2.26 <tasktroubleshooting>

The `<tasktroubleshooting>` element contains troubleshooting information that is designed to help users remedy the situation when a task does not complete as expected.

Comment by Bill Burns
Remove "troubleshooting" before information.

Kris Eberlein, 26 October 2022

Changed to "Task troubleshooting information is information that is designed to help users remedy the situation when a task does not complete as expected."

Eliot Kimber, 29 Oct 2022

Same comment as for `<steptroubleshooting>`:

Remove "that is designed ". Or perhaps completely replace with "contains information for responding when a task does not complete as expected."

"remedy the situation" presumes that remediation is within the reader's power, which it may not be. For example, the troubleshooting may consist entirely of gather logs and messages and reporting it to support.

Kris Eberlein, 07 November 2022

Changed to "Task troubleshooting information is information that is intended to help people respond to the situation if a task does not complete as expected."

Disposition: Completed

Comment by Stan Doherty
The most annoying processes that require troubleshooting are those that complete successfully but return inaccurate data or no data.

Kris Eberlein, 31 October 2022

For sure. Unfortunately, I don't think the spec or TC can do much about that ...

Disposition: Closed

Usage information

In particular, the `<tasktroubleshooting>` element can be used to explain how users can recover when the results of a task do not match those listed in the `<result>` element. The troubleshooting remedy typically contains one or more actions for solving a problem. For complex remedies, link to another task.

Comment by Eliot Kimber
Remove "In particular, ". Doesn't add anything and there's no obvious antecedent.

Kris Eberlein, 29 October 2022

Done.

Disposition: Completed

Tip Do not use `<note type="trouble">` inside of the `<tasktroubleshooting>` element.

Comment by dstevens73
again, would be helpful to say why

Kris Eberlein, 19 October 2022

Duplicate comment to that in <steptroubleshooting>.

Kris Eberlein, 20 October 2022

I talked with Dawn today. We both agree that this note should be removed. It's appropriate for guidelines for an implementation, but not the spec. And I removed it from this topic and the <tasktroubleshooting> topic.

Disposition: Completed

Specialization hierarchy

The `<tasktroubleshooting>` element is specialized from `<section>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59).

Example

In the following code sample, the `<tasktroubleshooting>` element contains brief information that explains the steps that the user can take when the results of a task are not as expected. For a complex remedy, the author could provide a link to another task topic.

```
<task id="add-new-categories>
  <title>Adding new user categories</title>
  <taskbody>
    <steps>
    <!-- ... -->
    </steps>
    <result>
      <p>The User Type menu displays the new types you added.</p>
    </result>
    <tasktroubleshooting>
      <p>If the User Type menu does not display the additions, try
      one or more of the following:
        <ul>
          <li>Refresh the page</li>
          <li>Verify that Add Types window is not still open; if so,
           go to it and press <uicontrol>OK</uicontrol>.</li>
        </ul>
      </p>
    </tasktroubleshooting>
  </taskbody>
```

Comment by Eliot Kimber
I know it's simply the history of how DITA evolved but this example really highlights the lack of any kind of general decision tree element that could go here.

Kris Eberlein, 29 October 2022

If my memory is correct, this is one of the examples for the DITA 1.3 proposal that Bob Thomas and Joann championed.

Disposition: Closed

2.27 <tutorialinfo>

The `<tutorialinfo>` element contains additional information that is useful when the task topic is part of a tutorial.

Comment by Zoë Lawson on 30 Oct 2022
I know that we're not listing what elements go where because the dtd/rng files should be validating that, and this is the spec and not a user guide, but I had zero idea this element existed.

My brain is also bending trying to think about creating different processing so that elements are inherently conditional, as opposed to using attributes, and how that may (or may not) be different.

Kris Eberlein, 30 October 2022

I don't think there are ever "inherently conditional" attributes. The use case for <tutorialinfo> is that you can use a single task topic to generate both task procedures AND learning exercises. (This element was part of DITA 1.0 and predates the learning and training specialization.) The idea was that you would use the <tutorialinfo> element to, for example, contain the values that a user would need when working through a tutorial, apply a conditional processing attribute to the <tutorialinfo> element, and exclude the <tutorialinfo> element when building the standard task topic.

Disposition: Closed

Specialization hierarchy

The `<tutorialinfo>` element is specialized from `<div>`. It is defined in the task module.

Attributes

The following attributes are available on this element: universal attributes (59).

Example

The following code sample shows how the `<tutorialinfo>` element might be used in a task topic that is part of a tutorial.

```
<steps>
  <step>
    <cmd>Do this</cmd>
    <tutorialinfo>In your editor, open the first element and click on
      the dialog.
    </tutorialinfo>
  </step>
  <step>
    <cmd>Do that</cmd>
    <tutorialinfo>Move the framulator into the foobar box.</tutorialinfo>
  </step>
</steps>
```

Comment by dstevens73
Clearly we need a better example. I see this most often used in situations where a task topic might be reused in a tutorial. so the cmd itself is generic to work in the user guide, and the tutorialinfo, which renders only in the tutorial, provides detailed information about a scenario, which is often presented in the <context> section. So: <cmd>Complete the dialog box.</cmd><tutorialinfo>For the example scenario, use the following values:<dl>...</dl></tutorialinfo>

Kris Eberlein, 19 October 2022

I agree that a new example is needed. Can you draft one? FWIW, this example appears as-is since DITA 1.0. In DITA 1.0, the topic had the following short description:

"The tutorial info (<tutorialinfo>) element contains information that is included in a step when a task is part of a tutorial. The <tutorialinfo> element allows you to turn a task into a learning exercise by including explanatory content about methods for completing the current step. The information should be excluded when a task is processed on its own."

Eliot Kimber, 29 Oct 2022

The second and third sentences seem like they would be appropriate in Usage and Rendering Expectations respectively.

Kris Eberlein, 29 Oct 2022

I agree that some of the DITA 1.0 content would be useful to have in a "Usage information" section. However, I don't think that the info about excluding content should go in a "Rendering expectations" section – it's just about basic conditional processing. (We tend to use "Rendering expectations" for information about what we expect processors to do vis-a-vis rendering the content, for example, that short descriptions should be rendered as the initial paragraphs.) It would be good if the example clearly had a value applied for a filtering attribute, however.

I added a "Usage information" section with the following content: "The <tutorialinfo> element enables task topics to be rendered as learning exercises. The element can contain explanatory information about a scenario, such as the data to use when completing a particular step in the exercise."

Disposition: Accepted

A Aggregated RFC-2119 statements

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

Item	Conformance statement
001 (33)	Processors which render the content of `<stepsection>` elements among the `<step>` elements **MUST NOT** number the `<stepsection>` elements.

B Attributes

This section contains definitions for commonly-used attributes. If an attribute is defined differently on a specific element, that information is covered in the topic for the specific element.

Comment by Kristen J Eberlein on 29 December 2021

Add a brief overview of the fact that some specific attributes are overloaded – and have different meanings depending on what element they are specified upon.

Disposition: Unassigned

B.1 Attribute groups

Many of the attributes used on DITA elements are defined in attribute groups. These attribute groups are used both in the grammar files and the specification,

Architectural attributes

This group includes a set of attributes that are defined for document-level elements such as `<topic>` and `<map>`:

@DITAArchVersion (architectural attributes)
: Specifies the version of the DITA architecture that is in use. The default value increases with each release of DITA. This attribute is in the namespace `http://dita.oasis-open.org/architecture/2005/`. This attribute is defined with the XML data type CDATA, but it uses a default value of the current version of DITA. The current default is "2.0".

Comment by Kristen J Eberlein on 28 September 2022

Is the second sentence really necessary? And do we want to specify that this attribute is set in the grammar files, specifically, all element-definition module files?

Disposition: Unassigned

@specializations (architectural attributes)
: Specifies the attribute-domain specializations that are included in the document-type shell. This attribute is set as a default within the document-type shell. The value varies depending on what domains are integrated into the document-type shell. For example, a grammar file that includes the specialized attributes `@audience`, `@deliveryTarget`, and `@newBaseAtt` would set the value to `@props/audience @props/deliveryTarget @base/newBaseAtt`.

@xmlns:ditaarch (architectural attributes)
: Declares the default DITA namespace. Although this is a namespace rather than an attribute, it is specified as an attribute in the DTD-based grammar files that are distributed by OASIS. The value is fixed to "http://dita.oasis-open.org/architecture/2005/".

Comment by Kristen J Eberlein on 28 September 2022

Do we want to be precise and change grammar files to "element-definition module files?

Disposition: Unassigned

Common map attributes

This group includes attributes that are frequently used on map elements:

Comment by Kristen J Eberlein on 28 September 2022

I've added draft comments to the attribute definitions in this section that explain how the attribute is defined in the "DITA map attributes" topic.

Disposition: Unassigned

@cascade (common map attributes)

Specifies how metadata attributes cascade within a map. The specification defines the following values:

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.

Processors can also define custom, implementation-specific tokens for this attribute.

See Cascading of metadata attributes in a DITA map for more information about how this attribute interacts with metadata attributes.

Comment by Kristen J Eberlein on 28 September 2022

Here is the content from the "DITA map attributes" topic that is different (the attribute value definitions are reused):

- Introductory paragraph

  Specifies whether the default rules for the cascading of metadata attributes in a DITA map apply. The following values are specified:
- See X for more information paragraph

  For more information, see Example: How the cascade attribute functions.

Disposition: Unassigned

@chunk (common map attributes)

Specifies how a processor should render a map or branch of a map. For example, it can be used to specify that individual topic documents should be rendered as a single document, or that a single document with multiple topics should be rendered as multiple documents.

The following values are valid:

combine
: Instructs a processor to combine the referenced source documents for rendering purposes. This is intended for cases where a publishing process normally results in a single output artifact for each source XML document.

split
: Instructs a processor to split each topic from the referenced source document into its own document for rendering purposes. This is intended for cases where a publishing process

normally results in a single output artifact for each source XML document, regardless of how many DITA topics exist within each source document.

Application-defined token
Other tokens can be defined by applications, but support for those tokens will vary.

For a detailed description of the `@chunk` attribute and its usage, see Chunking.

Comment by Kristen J Eberlein on 28 September 2022

Here is the content from the "DITA map attributes" topic"

@chunk
Specifies that the processor generates an interim set of DITA topics that are used as the input for the final processing. This can produce the following output results:

- Multi-topic files are transformed into smaller files, for example, individual HTML files for each DITA topic.
- Individual DITA topics are combined into a single file.

Specifying a value for the `@chunk` attribute on a `<map>` element establishes chunking behavior that applies to the entire map, unless overridden by `@chunk` attributes that are set on more specific elements in the DITA map. For a detailed description of the `@chunk` attribute and its usage, see Chunking.

Disposition: Unassigned

@collection-type (common map attributes)
Specifies how topics or links relate to each other. The processing default is "unordered", although no default is specified in the OASIS-provided grammar files. The following values are valid:

unordered
Indicates that the order of the child topics is not significant.

sequence
Indicates that the order of the child topics is significant. Output processors will typically link between them in order.

choice
Indicates that one of the children should be selected.

family
Indicates a tight grouping in which each of the referenced topics not only relates to the current topic but also relate to each other.

Comment by Kristen J Eberlein on 28 September 2022

Here is the content from the "DITA map attributes" topic:

@collection-type
The `@collection-type` attribute specifies how the children of a `<topicref>` element relate to their parent and to each other. This attribute, which is set on the parent element, typically is used by processors to determine how to generate navigation links in the rendered topics. For example, a `@collection-type` value of "sequence" indicates that children of the specifying `<topicref>` element represent an ordered sequence of topics; processors might add numbers to the list of child topics or generate next/previous links for online presentation. This attribute is available in topics on the `<linklist>` and `<linkpool>` elements, where it has the same behavior. Where the `@collection-`

`type` attribute is available on elements that cannot directly contain elements, the behavior of the attribute is undefined.

Disposition: Unassigned

Comment by Kristen J Eberlein on 28 September 2022

In the definitions of the supported values, do we want to refer to "resources" instead of "topics"? Since we specify that `@collection-type` specifies "how topics **or links** relate to each other" ...

Disposition: Unassigned

@keyscope (common map attributes)

Specifies that the element marks the boundaries of a key scope.

See STUB CONTENT for information on using this attribute.

Comment by Kristen J Eberlein on 28 September 2022

Here is the content from the "DITA map attributes" topic:

@keyscope

Defines a new scope for key definition and resolution, and gives the scope one or more names. For more information about key scopes, see Indirect key-based addressing.

Disposition: Unassigned

@linking (common map attributes)

Specifies linking characteristics of a topic specific to the location of this reference in a map. If the value is not specified locally, the value might cascade from another element in the map (for cascade rules, see Cascading of metadata attributes in a DITA map).

Comment by robander on Dec 28 2021
The text below matches 1.3 spec text but I'm nervous about "cannot link" type definition. It's describing how to generate links based on the current context in the map - it's not describing what the topic itself is allowed to link to, which is how I interpret "can".
Disposition: Unassigned

The following values are valid:

targetonly

A topic can only be linked to and cannot link to other topics.

sourceonly

A topic cannot be linked to but can link to other topics.

normal

A topic can be linked to and can link to other topics. Use this to override the linking value of a parent topic.

none

A topic cannot be linked to or link to other topics.

-dita-use-conref-target

See STUB CONTENT for more information.

Comment by Kristen J Eberlein on 28 September 2022

Here is the content from the "DITA map attributes" topic:

@linking

By default, the relationships between the topics that are referenced in a map are reciprocal:

- Child topics link to parent topics and vice versa.
- Next and previous topics in a sequence link to each other.
- Topics in a family link to their sibling topics.
- Topics referenced in the table cells of the same row in a relationship table link to each other. A topic referenced within a table cell does not (by default) link to other topics referenced in the same table cell.

This behavior can be modified by using the `@linking` attribute, which enables an author or information architect to specify how a topic participates in a relationship. The following values are valid:

linking="none"
Specifies that the topic does not exist in the map for the purposes of calculating links.

linking="sourceonly"
Specifies that the topic will link to its related topics but not vice versa.

linking="targetonly"
Specifies that the related topics will link to it but not vice versa.

linking="normal"
Default value. It specifies that linking will be reciprocal (the topic will link to related topics, and they will link back to it).

Authors also can create links directly in a topic by using the `<xref>` or `<link>` elements, but in most cases map-based linking is preferable, because links in topics create dependencies between topics that can hinder reuse.

Note that while the relationships between the topics that are referenced in a map are reciprocal, the relationships merely *imply* reciprocal links in generated output that includes links. The rendered navigation links are a function of the presentation style that is determined by the processor.

Disposition: Unassigned

@processing-role (common map attributes)
Describes the role that the referenced resource plays during processing. The following values are valid:

normal
Indicates that the resource is a readable part of the information. This is the processing default.

resource-only
Indicates that the resource should be used only for processing purposes. This topic should not be included in a rendered table of contents, and the topic should not be rendered on its own.

-dita-use-conref-target
See STUB CONTENT for more information.

If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value cascades from the closest containing element.

Comment by Kristen J Eberlein on 28 September 2022

Here is the content from the "DITA map attributes topic:

@processing-role
Specifies whether the topic or map referenced is processed normally or treated as a resource that is only included in order to resolve key or content references.

processing-role="normal"
The topic is a readable part of the information set. It is included in navigation and search results. This is the default value for the `<topicref>` element.

processing-role="resource-only"
The topic is used only as a resource for processing. It is not included in navigation or search results, nor is it rendered as a topic. This is the default value for the `<keydef>` element.

If the `@processing-role` attribute is not specified locally, the value cascades from the closest element in the containment hierarchy.

Disposition: Unassigned

@search (common map attributes)
Specifies whether the target is available for searching. If the value is not specified locally, the value might cascade from another element in the map (for cascade rules, see Cascading of metadata attributes in a DITA map). The following values are valid: "yes", "no", and "-dita-use-conref-target".

Comment by Kristen J Eberlein on 28 September 2022

Here is the content from the "DITA map attributes" topic:

@search
Specifies whether the topic is included in search indexes.

Disposition: Unassigned

@subjectrefs (common map attributes)
Specifies one or more keys that are each defined by a subject definition in a subject scheme map. Multiple values are separated by white space.

@toc (common map attributes)
Specifies whether a topic appears in the table of contents (TOC) based on the current map context. If the value is not specified locally, the value might cascade from another element in the map (for cascade rules, see Cascading of metadata attributes in a DITA map). The following values are valid:

yes
The topic appears in a generated TOC.

no
The topic does not appear in a generated TOC.

-dita-use-conref-target
See STUB CONTENT for more information.

Comment by Kristen J Eberlein on 28 September 2022

Here is the content from the "DITA map attributes" topic:

@toc
Specifies whether topics are excluded from navigation output, such as a Web site map or an online table of contents. By default, `<topicref>` hierarchies are included in navigation output; relationship tables are excluded.

Disposition: Unassigned

Complex table attributes

This group includes attributes that are defined on complex table elements. Unless other noted, these attributes are part of the OASIS Exchange Table Model. Complex table elements typically use only a subset of the attributes that are defined in this group.

@align (complex table attributes)
Specifies the horizontal alignment of text in table entries. The following values are valid:

left
Indicates left alignment of the text.

right
Indicates right alignment of the text.

center
Indicates center alignment of the text.

justify
Justifies the contents to both the left and the right.

char
Indicates character alignment. The text is aligned with the first occurrence of the character specified by the `@char` attribute.

-dita-use-conref-target
See STUB CONTENT for more information.

The `@align` attribute is available on the following table elements: `<colspec>`, `<entry>`, and `<tgroup>`.

@char (complex table attributes)
Specifies the alignment character, which is the character that is used for aligning the text in table entries. This attribute applies when `align="char"`. A value of "" (the null string) means there is no aligning character.

For example, if `align="char"` and `char="."` are specified, then text in the table entry aligns with the first occurrence of the period within the entry. This might be useful if decimal alignment is required.

The `@char` attribute is available on the following table elements: `<colspec>` and `<entry>`.

@charoff (complex table attributes)
Specifies the horizontal offset of the alignment character that is specified by the `@char` attribute. The value is a greater-than-zero number that is less than or equal to 100. It represents the percentage of the current column width by which the text is offset to the left of the alignment character.

For example, if `align="char"`, `char="."`, and `charoff="50"` are all specified, then text in the table entry is aligned 50% of the distance to the left of the first occurrence of the period character within the table entry.

The `@charoff` attribute is available on the following table elements: `<colspec>` and `<entry>`.

@colsep (complex table attributes)
Specifies whether to render column separators between table entries. The following values are valid: "0" (no separators) and "1" (separators).

The `@colsep` attribute is available on the following table elements: `<colspec>`, `<entry>`, `<table>`, and `<tgroup>`.

@rowheader (complex table attributes)

Specifies whether the entries in the respective column are row headers. The following values are valid:

firstcol

Indicates that entries in the first column of the table are row headers. This applies when the `@rowheader` attribute is specified on the `<table>` element.

headers

Indicates that entries of the column that is described using the `<colspec>` element are row headers. This applies when the `@rowheader` attribute is specified on the `<colspec>` element.

norowheader

Indicates that entries in the first column are not row headers. This applies when the `@rowheader` attribute is specified on the `<table>` element.

-dita-use-conref-target

See STUB CONTENT for more information.

Note This attribute is not part of the OASIS Exchange Table Model upon which DITA tables are based. Some processors or output formats might not support all values.

The `@rowheader` attribute is available on the following table elements: `<table>` and `<colspec>`.

@rowsep (complex table attributes)

Specifies whether to render row separators between table entries. The following values are valid: "0" (no separators) and "1" (separators).

The `@rowsep` attribute is available on the following table elements: `<colspec>`, `<entry>`, `<row>`, `<table>`, and `<tgroup>`.

@valign (complex table attributes)

Specifies the vertical alignment of text in table entries. The following values are valid:

bottom

Indicates that text is aligned with the bottom of the table entry.

middle

Indicates that text is aligned with the middle of the table entry.

top

Indicates that text is aligned with the top of the table entry.

-dita-use-conref-target

See STUB CONTENT for more information.

The `@valign` attribute is available on the following table elements: `<entry>`, `<tbody>`, `<thead>`, and `<row>`.

Data-element attributes

This group includes attributes that are defined on the `<data>` element and its specializations:

@datatype (data-element attributes)

Specifies the type of data contained in the `@value` attribute or within the `<data>` element. A typical use of `@datatype` will be the identifying URI for an XML Schema datatype.

@name (data-element attributes)

Defines a unique name for the object.

Comment by robander

Do we need to specify the scope of "unique" here?
Disposition: Unassigned

@value (data-element attributes)
Specifies a value associated with the current property or element.

Date attributes

This group includes attributes that take date values. They are defined on metadata elements that work with date information:

@expiry (date attributes)
Specifies the date when the information should be retired or refreshed. The date is specified using the ISO 8601 format: *YYYY-MM-DD*, where *YYYY* is the year, *MM* is the month (01 to 12), and *DD* is the day (01-31).

@golive (date attributes)
Specifies the publication or general availability (GA) date. The date is specified using the ISO 8601 format: *YYYY-MM-DD*, where *YYYY* is the year, *MM* is the month (01 to 12), and *DD* is the day (01-31).

Display attributes

This group includes attributes that affect the rendering of many elements: `@expanse`, `@frame`, and `@scale`.

@expanse (display attributes)
Specifies the horizontal placement of the element. The following values are valid:

column
Indicates that the element is aligned with the current column margin.

page
Indicates that the element is placed on the left page margin for left-to-right presentation or the right page margin for right-to-left presentation.

spread
Indicates that the object is rendered across a multi-page spread. If the output format does not have anything that corresponds to spreads, then "spread" has the same meaning as "page".

textline
Indicates that the element is aligned with the left (for left-to-right presentation) or right (for right-to-left presentation) margin of the current text line and takes indentation into account.

-dita-use-conref-target
See STUB CONTENT for more information.

For `<table>`, in place of the `@expanse` attribute that is used by other DITA elements, the `@pgwide` attribute is used in order to conform to the OASIS Exchange Table Model.

Some processors or output formats might not support all values.

@frame (display attributes)
Specifies which portion of a border surrounds the element. The following values are valid:

all
Indicates that a line is rendered at the top, bottom, left, and right of the containing element.

bottom
Indicates that a line is rendered at the bottom of the containing element.

none
Indicates that no lines are rendered.

sides
Indicates that a line is rendered at the left and right of the containing element.

top
Indicates that a line is rendered at the top of the containing element.

topbot
Indicates that a line is rendered at the top and bottom of the containing element.

-dita-use-conref-target
See STUB CONTENT for more information.

Some processors or output formats might not support all values.

@scale (display attributes)
Specifies the percentage by which fonts are resized in relation to the normal text size. The value of this attribute is a positive integer. When used on `<table>` or `<simpletable>`, the following values are valid: "50", "60", "70", "80", "90", "100", "110", "120", "140", "160", "180", "200", and -dita-use-conref-target.

This attribute is primarily useful for print-oriented display. Some processors might not support all values.

If the `@scale` attribute is specified on an element that contains an image, the image is not scaled. The image is scaled **only** if a scaling property is explicitly specified for the `<image>` element.

ID and conref attributes

This group includes the attributes that enable the naming and referencing of elements:

@conaction
Specifies how the element content will be pushed into a new location. The following values are valid:

mark
The element acts as a marker when pushing content before or after the target, to help ensure that the push action is valid. The element with `conaction="mark"` also specifies the target of the push action with `@conref`. Content inside of the element with `conaction="mark"` is not pushed to the new location.

pushafter
Content from this element is pushed after the location specified by `@conref` on the element with `conaction="mark"`. The element with `conaction="pushafter"` is the first sibling element after the element with `conaction="mark"`.

pushbefore
Content from this element is pushed before the location specified by `@conref` on the element with `conaction="mark"`. The element with `conaction="pushbefore"` is the first sibling element before the element with `conaction="mark"`.

pushreplace
Content from this element replaces any content from the element referenced by the `@conref` attribute. A second element with `conaction="mark"` is not used when using `conaction="pushreplace"`.

-dita-use-conref-target
See STUB CONTENT for more information.

See STUB CONTENT for examples and details about the syntax.

@conkeyref
: Specifies a key name or a key name with an element ID that acts as an indirect reference to reusable content. The referenced content is used in place of the content of the current element. See STUB CONTENT for more details about the syntax and behaviors.

@conref
: Specifies a URI that references a DITA element. The referenced content is used in place of the content of the current element. See STUB CONTENT for examples and details about the syntax.

@conrefend
: Specifies a URI that references the last element in a sequence of elements, with the first element of the sequence specified by `@conref`. The referenced sequence of elements is used in place of the content of the current element. See STUB CONTENT for examples and details about the syntax.

@id
: Specifies an identifier for the current element. This ID is the target for references by `@href` and `@conref` attributes and for external applications that refer to DITA or LwDITA content. This attribute is defined with the XML data type NMTOKEN, except where noted for specific elements within the language reference.

  See id attribute for more details.

@id
: Specifies an identifier for the current element. This ID is the target for references by `@href` and `@conref` attributes and for external applications that refer to DITA or LwDITA content. This attribute is defined with the XML data type NMTOKEN, except where noted for specific elements within the language reference.

  See id attribute for more details.

Inclusion attributes

This group includes attributes defined on `<include>` and its specializations:

Comment by Kristen J Eberlein on 28 September 2002

What is specialized from `<include>`? Both base (if any) and technical content ...

Disposition: Unassigned

@encoding (inclusion attributes)

Comment by Kristen J Eberlein on 29 April 2019

Can we replace "should" in the following definition?

Disposition: Unassigned

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

When `parse="xml"`, standard XML parsing rules apply for the detection of character encoding. The necessity and uses of `@encoding` for non-standard values of `@parse` are implementation-dependent.

@parse (inclusion attributes)
Specifies the processing expectations for the referenced resource. Processors must support the following values:

text

The contents should be treated as plain text. Reserved XML characters should be displayed, and not interpreted as XML markup.

xml

The contents of the referenced resource should be treated as an XML document, and the referenced element should be inserted at the location of the `<include>` element. If a fragment identifier is included in the address of the content, processors must select the element with the specified ID. If no fragment identifier is included, the root element of the referenced XML document is selected. Any grammar processing should be performed during resolution, such that default attribute values are explicitly populated. Prolog content must be discarded.

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

Processors may support other values for the `@parse` attribute with proprietary processing semantics. Processors should issue warnings and use `<fallback>` when they encounter unsupported `@parse` values. Non-standard `@parse` instructions should be expressed as URIs.

Note Proprietary `@parse` values will likely limit the portability and interoperability of DITA content, so should be used with care.

Link relationship attributes

This group includes attributes whose values can be used for representing navigational relationships:

@format (link-relationship attributes)
Specifies the format of the resource that is referenced. See STUB CONTENT for detailed information on supported values and processing implications.

@href (link-relationship attributes)
Specifies a reference to a resource. See STUB CONTENT for detailed information on supported values and processing implications.

@scope (link-relationship attributes)
Specifies the closeness of the relationship between the current document and the referenced resource. The following values are valid: "local", "peer", "external", and "-dita-use-conref-target".

See STUB CONTENT for detailed information on supported values and processing implications.

@type (link-relationship attributes)
Describes the target of a reference. See STUB CONTENT for detailed information on supported values and processing implications.

Localization attributes

This group includes attributes that are related to translation and localization: `@dir`, `@translate`, and `@xml:lang`.

@dir

Identifies or overrides the text directionality. The following values are valid:

lro
Indicates an override of the Unicode Bidirectional Algorithm, forcing the element into left-to-right mode.

ltr
Indicates left-to-right.

rlo
Indicates an override of the Unicode Bidirectional Algorithm, forcing the element into right-to-left mode.

rtl
Indicates right-to-left.

-dita-use-conref-target
See STUB CONTENT for more information.

See The dir attribute for more information.

@translate

Comment by Kristen J Eberlein on 31 December 2021

Does Element-by-element recommendations for translators really provide suggested processing defaults for each element? I thought it covered whether an element was block or in-line and whether there were considerations that translators needed to be aware of.

Disposition: Unassigned

Specifies whether the content of the element should be translated. The following values are valid: "yes", "no", and "-dita-use-conref-target".

See Element-by-element recommendations for translators for suggested processing defaults for each element.

@xml:lang
Specifies the language of the content contained in an element. The following values are valid: language tokens or the null string. The `@xml:lang` attribute and its values are described in the Extensible Markup Language 1.0 specification, fifth edition.

Metadata attributes

This group includes common metadata attributes. The `@base` and `@props` attributes can be specialized.

@base
Specifies metadata about the current element. It is often used as a base for specialized attributes that have a simple syntax for values but are not filtering or flagging attributes.

The `@base` attribute takes a space-delimited set of values. However, when serving as a container for generalized attributes, the attribute values will be more complex. See Attribute generalization for more details.

@importance
Specifies the importance or priority that is assigned to an element. The following values are valid: "default", "deprecated", "high", "low", "normal", "obsolete", "optional", "recommended", "required", "urgent", and "-dita-use-conref-target". This attribute is not used for DITAVAL-based filtering or flagging, although applications might use the importance value to highlight elements. For example, in

steps of a task, the value of the `@importance` attribute indicates whether a step is optional or required.

@props
Specifies metadata about the current element. New attributes can be specialized from the `@props` attribute. This is an attribute that supports conditional processing for filtering or flagging. If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value cascades from the closest containing element.

The `@props` attribute takes a space-delimited set of values. However, when acting as a container for generalized attributes, the attribute values will be more complex; see Attribute generalization for more details.

@rev
Specifies a revision level of an element that identifies when the element was added or modified. It can be used to flag outputs when it matches a run-time parameter. It cannot be used for filtering nor is it sufficient to be used for version control. If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value cascades from the closest containing element.

@status
Specifies modification status of the current element. The following values are valid: "new", "changed", "deleted", "unchanged", and "-dita-use-conref-target".

Simple table attributes

This group includes attributes that are defined only on the `<simpletable>` element: `@keycol` and `@relcolwidth`. These attributes are listed in a group because the `<simpletable>` element is frequently used as a specialization base.

@keycol (simpletable attributes)
Specifies the column that contains the content that represents the key to the tabular structure. If `@keycol` is present and assigned a numerical value, the specified column is treated as a vertical header.

@relcolwidth (simpletable attributes)
Specifies the width of each column in relationship to the width of the other columns. The value is a space-separated list of relative column widths. Each column width is specified as a positive integer or decimal number followed by an asterisk character.

For example, the value `relcolwidth="1* 2* 3*"` gives a total of 6 units across three columns. The relative widths are 1/6, 2/6, and 3/6 (16.7%, 33.3%, and 50%). Similarly, the value `relcolwidth="90* 150*"` causes relative widths of 90/240 and 150/240 (37.5% and 62.5%).

Universal attributes

This group defines a set of attributes that are available on almost all DITA elements. It includes all elements in the ID, localization, and metadata attribute groups, as well as the following attributes:

@class *(not for use by authors)*
This attribute is not for use by authors. If an editor displays `@class` attribute values, do not edit them. Specifies a default value that defines the specialization ancestry of the element. Its predefined values allow DITA tools to work correctly with specialized elements. In a generalized DITA document the `@class` attribute value in the generalized instance might differ from the default value for the `@class` attribute for the element as given in the DTD or schema. See The class attribute rules and syntax for more information. This attribute is specified on every element except for the `<dita>` container element. It is always specified with a default value, which varies for each element.

@outputclass

Specifies a role that the element is playing. The role must be consistent with the basic semantic and expectations for the element. In particular, the `@outputclass` attribute can be used for styling during output processing; HTML output will typically preserve `@outputclass` for CSS processing.

> **Comment by robander**
> I don't like "The role must be consistent...", that seems like best practice that cannot be normative – and I could easily say outputclass="flashy" which makes my element show up with sparkles, and has nothing to do with "the basic semantic and expectations for the element".
> **Disposition: Unassigned**

B.2 Universal attribute group

The universal attribute group defines a set of common attributes that are available on almost every DITA element. The universal attribute group includes all attributes from the ID, localization, and metadata attribute groups, plus the `@class` and `@outputclass` attributes.

> **Comment by Kristen J Eberlein on 29 December 2021**
>
> This is something wrong with the organizational structure of this topic ... Look at it in outline form, and check that the sections, titles, and content all make logical sense with the topic title of "Universal attribute group".
>
> **Disposition: Unassigned**

Common attribute groups

The following attribute groups are referenced in this specification. They are also used in the grammar files when the element attributes are defined.

Universal attributes

Includes `@class` and `@outputclass`, along with every attribute in the ID, localization, and metadata attribute groups.

ID attributes

This group includes the attributes that enable the naming and referencing of elements: `@conaction`, `@conkeyref`, `@conref`, `@conrefend`, and `@id`.

Localization attributes

This group includes attributes that are related to translation and localization: `@dir`, `@translate`, and `@xml:lang`.

Metadata attributes

> **Comment by Kristen J Eberlein on 31 December 2021**
>
> Why do we need to mention that two attributes are available for specialization here? I think it makes the paragraph hard to read.
>
> **Disposition: Unassigned**

This group includes common metadata attributes, two of which are available for specialization: `@base`, `@importance`, `@props`, `@rev`, and `@status`.

The base DITA vocabulary from OASIS includes several specializations of `@props`: `@audience`, `@deliveryTarget`, `@otherprops`, `@platform`, and `@product`. These attributes are defined as attribute-extension domains. By default, they are integrated into all OASIS-provided document-type shells, but they can be made unavailable by implementing custom document-type shells.

Comment by Kristen J Eberlein on 29 December 2021

Why do we provide information about specialization and custom document-type shells here? I think that information could be removed.

Disposition: Unassigned

Universal attribute definitions

The universal attributes for OASIS DITA elements are defined below. Specialized attributes, which are part of the OASIS distribution but are only available when explicitly included in a shell, are noted in the list.

@audience *(specialized attribute)*
: Indicates the intended audience for the element. If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value cascades from the closest containing element.

@base
: Specifies metadata about the current element. It is often used as a base for specialized attributes that have a simple syntax for values but are not filtering or flagging attributes.

  The `@base` attribute takes a space-delimited set of values. However, when serving as a container for generalized attributes, the attribute values will be more complex. See Attribute generalization for more details.

@class *(not for use by authors)*
: *This attribute is not for use by authors. If an editor displays `@class` attribute values, do not edit them.* Specifies a default value that defines the specialization ancestry of the element. Its predefined values allow DITA tools to work correctly with specialized elements. In a generalized DITA document the `@class` attribute value in the generalized instance might differ from the default value for the `@class` attribute for the element as given in the DTD or schema. See The class attribute rules and syntax for more information. This attribute is specified on every element except for the `<dita>` container element. It is always specified with a default value, which varies for each element.

@conaction
: Specifies how the element content will be pushed into a new location. The following values are valid:

  mark
  : The element acts as a marker when pushing content before or after the target, to help ensure that the push action is valid. The element with `conaction="mark"` also specifies the target of the push action with `@conref`. Content inside of the element with `conaction="mark"` is not pushed to the new location.

  pushafter
  : Content from this element is pushed after the location specified by `@conref` on the element with `conaction="mark"`. The element with `conaction="pushafter"` is the first sibling element after the element with `conaction="mark"`.

  pushbefore
  : Content from this element is pushed before the location specified by `@conref` on the element with `conaction="mark"`. The element with `conaction="pushbefore"` is the first sibling element before the element with `conaction="mark"`.

pushreplace
Content from this element replaces any content from the element referenced by the `@conref` attribute. A second element with `conaction="mark"` is not used when using `conaction="pushreplace"`.

-dita-use-conref-target
See STUB CONTENT for more information.

See STUB CONTENT for examples and details about the syntax.

@conkeyref
Specifies a key name or a key name with an element ID that acts as an indirect reference to reusable content. The referenced content is used in place of the content of the current element. See STUB CONTENT for more details about the syntax and behaviors.

@conref
Specifies a URI that references a DITA element. The referenced content is used in place of the content of the current element. See STUB CONTENT for examples and details about the syntax.

@conrefend
Specifies a URI that references the last element in a sequence of elements, with the first element of the sequence specified by `@conref`. The referenced sequence of elements is used in place of the content of the current element. See STUB CONTENT for examples and details about the syntax.

@deliveryTarget *(specialized attribute)*
Specifies the intended delivery target of the content, for example, "html", "pdf", or "epub". If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value cascades from the closest containing element.

@dir

Identifies or overrides the text directionality. The following values are valid:

lro
Indicates an override of the Unicode Bidirectional Algorithm, forcing the element into left-to-right mode.

ltr
Indicates left-to-right.

rlo
Indicates an override of the Unicode Bidirectional Algorithm, forcing the element into right-to-left mode.

rtl
Indicates right-to-left.

-dita-use-conref-target
See STUB CONTENT for more information.

See The dir attribute for more information.

@id
Specifies an identifier for the current element. This ID is the target for references by `@href` and `@conref` attributes and for external applications that refer to DITA or LwDITA content. This attribute is defined with the XML data type NMTOKEN, except where noted for specific elements within the language reference.

See id attribute for more details.

@importance

Specifies the importance or priority that is assigned to an element. The following values are valid: "default", "deprecated", "high", "low", "normal", "obsolete", "optional", "recommended", "required", "urgent", and "-dita-use-conref-target". This attribute is not used for DITAVAL-based filtering or flagging, although applications might use the importance value to highlight elements. For example, in steps of a task, the value of the `@importance` attribute indicates whether a step is optional or required.

@otherprops *(specialized attribute)*

Specifies a property or properties that provide selection criteria for the element. Alternatively, the `@props` attribute can be specialized to provide a new metadata attribute instead of using the general `@otherprops` attribute. If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value cascades from the closest containing element.

@outputclass

Specifies a role that the element is playing. The role must be consistent with the basic semantic and expectations for the element. In particular, the `@outputclass` attribute can be used for styling during output processing; HTML output will typically preserve `@outputclass` for CSS processing.

Comment by robander
I don't like "The role must be consistent...", that seems like best practice that cannot be normative – and I could easily say outputclass="flashy" which makes my element show up with sparkles, and has nothing to do with "the basic semantic and expectations for the element".
Disposition: Unassigned

@platform *(specialized attribute)*

Indicates operating system and hardware. If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value cascades from the closest containing element.

Comment by robander
I think this could specify a platform that is not an operating system or hardware, right? The current definition explicitly limits platform to those two … maybe "Specifies a platform or platforms to which the element applies, such as the operating system or hardware relevant to a task."
Disposition: Unassigned

@product *(specialized attribute)*

Specifies the name of the product to which the element applies. If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value cascades from the closest containing element.

@props

Specifies metadata about the current element. New attributes can be specialized from the `@props` attribute. This is an attribute that supports conditional processing for filtering or flagging. If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value cascades from the closest containing element.

The `@props` attribute takes a space-delimited set of values. However, when acting as a container for generalized attributes, the attribute values will be more complex; see Attribute generalization for more details.

@rev

Specifies a revision level of an element that identifies when the element was added or modified. It can be used to flag outputs when it matches a run-time parameter. It cannot be used for filtering nor is it sufficient to be used for version control. If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value cascades from the closest containing element.

@status
Specifies modification status of the current element. The following values are valid: "new", "changed", "deleted", "unchanged", and "-dita-use-conref-target".

@translate

Comment by Kristen J Eberlein on 31 December 2021

Does Element-by-element recommendations for translators really provide suggested processing defaults for each element? I thought it covered whether an element was block or in-line and whether there were considerations that translators needed to be aware of.

Disposition: Unassigned

Specifies whether the content of the element should be translated. The following values are valid: "yes", "no", and "-dita-use-conref-target".

See Element-by-element recommendations for translators for suggested processing defaults for each element.

@xml:lang
Specifies the language of the content contained in an element. The following values are valid: language tokens or the null string. The `@xml:lang` attribute and its values are described in the Extensible Markup Language 1.0 specification, fifth edition.

B.3 Common attributes

The common attributes topic collects defines most of the attributes that are used on more than one base element.

Common attribute groups

The following groups are referenced in this specification, and they are also used in grammar files when defining attributes for elements.

Architectural attributes

This group includes a set of attributes that are defined for document-level elements such as `<topic>` and `<map>`: `@DITAArchVersion`, `@specializations`, and `@xmlns:ditaarch`.

Common map attributes

This group includes attributes that are frequently used on map elements: `@cascade`, `@chunk`, `@collection-type`, `@keyscope`, `@linking`, `@processing-role`, `@search`, `@toc`, and `@subjectrefs`.

Complex table attributes

This group includes attributes that are defined on table elements but not simple table elements. These attributes are part of the OASIS Exchange Table Model, unless otherwise noted. Table elements generally use only a subset of the attributes that are defined in this group. This group contains the following attributes: `@align`, `@char`, `@charoff`, `@colsep`, `@rowheader`, `@rowsep`, and `@valign`.

Data-element attributes
Includes attributes defined on `<data>` and its many specializations: `@datatype`, `@name`, and `@value`

Date attributes
Includes attributes that take date values, and are defined on metadata elements that work with date information: `@expiry` and `@golive`

Display attributes

This group includes attributes that affect the rendering of many elements: `@expanse`, `@frame`, and `@scale`.

Inclusion attributes
Includes attributes defined on `<include>` and its specializations: `@encoding` and `@parse`.

Link-relationship attributes

This group includes attributes whose values can be used for representing navigational relationships: `@format`, `@href`, `@type`, and `@scope`.

Simple table attributes

Comment by Kristen J Eberlein on 29 December 2021

If I have jumped to this place in a document from the element-reference topic, I want the attributes listed here in the "Simple table group" to be hyperlinked to the actual definition.

Disposition: Unassigned

This group includes attributes that are defined only on the `<simpletable>` element: `@keycol` and `@relcolwidth`. These attributes are listed in a group because the `<simpletable>` element is frequently used as a specialization base.

Other attributes (not in a group)

These are attributes that are used in the same way on more than one base element, but they are not formally grouped together: `@compact`, `@duplicates`, `@otherrole`, `@role`, and `@title-role`.

Common attribute definitions

Common attributes, including those in the groups listed above, are defined as follows.

@align (complex table attributes)
Specifies the horizontal alignment of text in table entries. The following values are valid:

left
Indicates left alignment of the text.

right
Indicates right alignment of the text.

center
Indicates center alignment of the text.

justify
Justifies the contents to both the left and the right.

char
Indicates character alignment. The text is aligned with the first occurrence of the character specified by the `@char` attribute.

-dita-use-conref-target
See STUB CONTENT for more information.

The `@align` attribute is available on the following table elements: `<colspec>`, `<entry>`, and `<tgroup>`.

@cascade (common map attributes)

Specifies how metadata attributes cascade within a map. The specification defines the following values:

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.

Processors can also define custom, implementation-specific tokens for this attribute.

See Cascading of metadata attributes in a DITA map for more information about how this attribute interacts with metadata attributes.

Comment by Kristen J Eberlein on 28 September 2022

Here is the content from the "DITA map attributes" topic that is different (the attribute value definitions are reused):

- Introductory paragraph

  Specifies whether the default rules for the cascading of metadata attributes in a DITA map apply. The following values are specified:
- See X for more information paragraph

  For more information, see Example: How the cascade attribute functions.

Disposition: Unassigned

@char (complex table attributes)

Specifies the alignment character, which is the character that is used for aligning the text in table entries. This attribute applies when `align="char"`. A value of "" (the null string) means there is no aligning character.

For example, if `align="char"` and `char="."` are specified, then text in the table entry aligns with the first occurrence of the period within the entry. This might be useful if decimal alignment is required.

The `@char` attribute is available on the following table elements: `<colspec>` and `<entry>`.

@charoff (complex table attributes)

Specifies the horizontal offset of the alignment character that is specified by the `@char` attribute. The value is a greater-than-zero number that is less than or equal to 100. It represents the percentage of the current column width by which the text is offset to the left of the alignment character.

For example, if `align="char"`, `char="."`, and `charoff="50"` are all specified, then text in the table entry is aligned 50% of the distance to the left of the first occurrence of the period character within the table entry.

The `@charoff` attribute is available on the following table elements: `<colspec>` and `<entry>`.

@chunk (common map attributes)

Specifies how a processor should render a map or branch of a map. For example, it can be used to specify that individual topic documents should be rendered as a single document, or that a single document with multiple topics should be rendered as multiple documents.

The following values are valid:

combine

Instructs a processor to combine the referenced source documents for rendering purposes. This is intended for cases where a publishing process normally results in a single output artifact for each source XML document.

split

Instructs a processor to split each topic from the referenced source document into its own document for rendering purposes. This is intended for cases where a publishing process normally results in a single output artifact for each source XML document, regardless of how many DITA topics exist within each source document.

Application-defined token

Other tokens can be defined by applications, but support for those tokens will vary.

For a detailed description of the `@chunk` attribute and its usage, see Chunking.

Comment by Kristen J Eberlein on 28 September 2022

Here is the content from the "DITA map attributes" topic"

@chunk

Specifies that the processor generates an interim set of DITA topics that are used as the input for the final processing. This can produce the following output results:

- Multi-topic files are transformed into smaller files, for example, individual HTML files for each DITA topic.
- Individual DITA topics are combined into a single file.

Specifying a value for the `@chunk` attribute on a `<map>` element establishes chunking behavior that applies to the entire map, unless overridden by `@chunk` attributes that are set on more specific elements in the DITA map. For a detailed description of the `@chunk` attribute and its usage, see Chunking.

Disposition: Unassigned

@collection-type (common map attributes)

Specifies how topics or links relate to each other. The processing default is "unordered", although no default is specified in the OASIS-provided grammar files. The following values are valid:

unordered

Indicates that the order of the child topics is not significant.

sequence

Indicates that the order of the child topics is significant. Output processors will typically link between them in order.

choice

Indicates that one of the children should be selected.

family

Indicates a tight grouping in which each of the referenced topics not only relates to the current topic but also relate to each other.

Comment by Kristen J Eberlein on 28 September 2022

Here is the content from the "DITA map attributes" topic:

@collection-type

The `@collection-type` attribute specifies how the children of a `<topicref>` element relate to their parent and to each other. This attribute, which is set on the parent element, typically is used by processors to determine how to generate navigation links in the rendered topics. For example, a `@collection-type` value of "sequence" indicates that children of the specifying `<topicref>` element represent an ordered sequence of topics; processors might add numbers to the list of child topics or generate next/previous links for online presentation. This attribute is available in topics on the `<linklist>` and `<linkpool>` elements, where it has the same behavior. Where the `@collection-type` attribute is available on elements that cannot directly contain elements, the behavior of the attribute is undefined.

Disposition: Unassigned

Comment by Kristen J Eberlein on 28 September 2022

In the definitions of the supported values, do we want to refer to "resources" instead of "topics"? Since we specify that `@collection-type` specifies "how topics **or links** relate to each other" ...

Disposition: Unassigned

@colsep (complex table attributes)

Specifies whether to render column separators between table entries. The following values are valid: "0" (no separators) and "1" (separators).

The `@colsep` attribute is available on the following table elements: `<colspec>`, `<entry>`, `<table>`, and `<tgroup>`.

@compact

Specifies whether the vertical spacing between list items is tightened. The following values are valid: "yes", "no", and "-dita-use-conref-target". Some DITA processors or output formats might not support the `@compact` attribute.

@datatype (data-element attributes)

Specifies the type of data contained in the `@value` attribute or within the `<data>` element. A typical use of `@datatype` will be the identifying URI for an XML Schema datatype.

@DITAArchVersion (architectural attributes)

Specifies the version of the DITA architecture that is in use. The default value increases with each release of DITA. This attribute is in the namespace `http://dita.oasis-open.org/architecture/2005/`. This attribute is defined with the XML data type CDATA, but it uses a default value of the current version of DITA. The current default is "2.0".

Comment by Kristen J Eberlein on 28 September 2022

Is the second sentence really necessary? And do we want to specify that this attribute is set in the grammar files, specifically, all element-definition module files?

Disposition: Unassigned

@duplicates

Specifies whether duplicate links are removed from a group of links. Duplicate links are links that address the same resource using the same properties, such as link text and link role. How duplicate links are determined is processor-specific. The following values are valid:

yes
Specifies that duplicate links are retained.

no
Specifies that duplicate links are removed.

-dita-use-conref-target
See STUB CONTENT for more information.

The suggested processing default is "yes" within `<linklist>` elements and "no" for other links.

Comment by robander on Dec 28 2021
"How duplicate links are determined is processor-specific" ==> this should be included in any updates to standardize language around "implementation dependent".
Disposition: Unassigned

@encoding (inclusion attributes)

Comment by Kristen J Eberlein on 29 April 2019

Can we replace "should" in the following definition?

Disposition: Unassigned

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

When `parse="xml"`, standard XML parsing rules apply for the detection of character encoding. The necessity and uses of `@encoding` for non-standard values of `@parse` are implementation-dependent.

@expanse (display attributes)

Specifies the horizontal placement of the element. The following values are valid:

column
Indicates that the element is aligned with the current column margin.

page
Indicates that the element is placed on the left page margin for left-to-right presentation or the right page margin for right-to-left presentation.

spread
Indicates that the object is rendered across a multi-page spread. If the output format does not have anything that corresponds to spreads, then "spread" has the same meaning as "page".

textline
Indicates that the element is aligned with the left (for left-to-right presentation) or right (for right-to-left presentation) margin of the current text line and takes indentation into account.

-dita-use-conref-target
See STUB CONTENT for more information.

For `<table>`, in place of the `@expanse` attribute that is used by other DITA elements, the `@pgwide` attribute is used in order to conform to the OASIS Exchange Table Model.

Some processors or output formats might not support all values.

@expiry (date attributes)
Specifies the date when the information should be retired or refreshed. The date is specified using the ISO 8601 format: *YYYY-MM-DD*, where *YYYY* is the year, *MM* is the month (01 to 12), and *DD* is the day (01-31).

@format (link-relationship attributes)
Specifies the format of the resource that is referenced. See STUB CONTENT for detailed information on supported values and processing implications.

@frame (display attributes)
Specifies which portion of a border surrounds the element. The following values are valid:

all
Indicates that a line is rendered at the top, bottom, left, and right of the containing element.

bottom
Indicates that a line is rendered at the bottom of the containing element.

none
Indicates that no lines are rendered.

sides
Indicates that a line is rendered at the left and right of the containing element.

top
Indicates that a line is rendered at the top of the containing element.

topbot
Indicates that a line is rendered at the top and bottom of the containing element.

-dita-use-conref-target
See STUB CONTENT for more information.

Some processors or output formats might not support all values.

@golive (date attributes)
Specifies the publication or general availability (GA) date. The date is specified using the ISO 8601 format: *YYYY-MM-DD*, where *YYYY* is the year, *MM* is the month (01 to 12), and *DD* is the day (01-31).

@href (link-relationship attributes)
Specifies a reference to a resource. See STUB CONTENT for detailed information on supported values and processing implications.

@keycol (simpletable attributes)
Specifies the column that contains the content that represents the key to the tabular structure. If `@keycol` is present and assigned a numerical value, the specified column is treated as a vertical header.

@keyref
Specifies a key name that acts as a redirectable reference based on a key definition within a map. See STUB CONTENT for information on using this attribute.

Comment by robander
The definiton above for @keyref should be synchronized with the definition in the linked section on keys.

Disposition: Unassigned

@keys
Specifies one or more names for a resource. See STUB CONTENT for information on using this attribute.

@keyscope (common map attributes)
Specifies that the element marks the boundaries of a key scope.

See STUB CONTENT for information on using this attribute.

Comment by Kristen J Eberlein on 28 September 2022

Here is the content from the "DITA map attributes" topic:

@keyscope
Defines a new scope for key definition and resolution, and gives the scope one or more names. For more information about key scopes, see Indirect key-based addressing.

Disposition: Unassigned

@linking (common map attributes)
Specifies linking characteristics of a topic specific to the location of this reference in a map. If the value is not specified locally, the value might cascade from another element in the map (for cascade rules, see Cascading of metadata attributes in a DITA map).

Comment by robander on Dec 28 2021
The text below matches 1.3 spec text but I'm nervous about "cannot link" type definition. It's describing how to generate links based on the current context in the map - it's not describing what the topic itself is allowed to link to, which is how I interpret "can".
Disposition: Unassigned

The following values are valid:

targetonly
A topic can only be linked to and cannot link to other topics.

sourceonly
A topic cannot be linked to but can link to other topics.

normal
A topic can be linked to and can link to other topics. Use this to override the linking value of a parent topic.

none
A topic cannot be linked to or link to other topics.

-dita-use-conref-target
See STUB CONTENT for more information.

Comment by Kristen J Eberlein on 28 September 2022

Here is the content from the "DITA map attributes" topic:

@linking

By default, the relationships between the topics that are referenced in a map are reciprocal:

- Child topics link to parent topics and vice versa.
- Next and previous topics in a sequence link to each other.

- Topics in a family link to their sibling topics.
- Topics referenced in the table cells of the same row in a relationship table link to each other. A topic referenced within a table cell does not (by default) link to other topics referenced in the same table cell.

This behavior can be modified by using the `@linking` attribute, which enables an author or information architect to specify how a topic participates in a relationship. The following values are valid:

linking="none"
Specifies that the topic does not exist in the map for the purposes of calculating links.

linking="sourceonly"
Specifies that the topic will link to its related topics but not vice versa.

linking="targetonly"
Specifies that the related topics will link to it but not vice versa.

linking="normal"
Default value. It specifies that linking will be reciprocal (the topic will link to related topics, and they will link back to it).

Authors also can create links directly in a topic by using the `<xref>` or `<link>` elements, but in most cases map-based linking is preferable, because links in topics create dependencies between topics that can hinder reuse.

Note that while the relationships between the topics that are referenced in a map are reciprocal, the relationships merely *imply* reciprocal links in generated output that includes links. The rendered navigation links are a function of the presentation style that is determined by the processor.

Disposition: Unassigned

@name (data-element attributes)
Defines a unique name for the object.

Comment by robander
Do we need to specify the scope of "unique" here?
Disposition: Unassigned

@otherrole
Specifies an alternate role for a link relationship when the `@role` attribute is set to "other".

@parse (inclusion attributes)
Specifies the processing expectations for the referenced resource. Processors must support the following values:

text

The contents should be treated as plain text. Reserved XML characters should be displayed, and not interpreted as XML markup.

xml

The contents of the referenced resource should be treated as an XML document, and the referenced element should be inserted at the location of the `<include>` element. If a fragment identifier is included in the address of the content, processors must select the element with the specified ID. If no fragment identifier is included, the root element of the referenced XML document is selected. Any grammar processing should be performed during resolution, such that default attribute values are explicitly populated. Prolog content must be discarded.

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

Processors may support other values for the `@parse` attribute with proprietary processing semantics. Processors should issue warnings and use `<fallback>` when they encounter unsupported `@parse` values. Non-standard `@parse` instructions should be expressed as URIs.

Note Proprietary `@parse` values will likely limit the portability and interoperability of DITA content, so should be used with care.

@processing-role (common map attributes)
Describes the role that the referenced resource plays during processing. The following values are valid:

normal
Indicates that the resource is a readable part of the information. This is the processing default.

resource-only
Indicates that the resource should be used only for processing purposes. This topic should not be included in a rendered table of contents, and the topic should not be rendered on its own.

-dita-use-conref-target
See STUB CONTENT for more information.

If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value cascades from the closest containing element.

Comment by Kristen J Eberlein on 28 September 2022

Here is the content from the "DITA map attributes topic:

@processing-role
Specifies whether the topic or map referenced is processed normally or treated as a resource that is only included in order to resolve key or content references.

processing-role="normal"
The topic is a readable part of the information set. It is included in navigation and search results. This is the default value for the `<topicref>` element.

processing-role="resource-only"
The topic is used only as a resource for processing. It is not included in navigation or search results, nor is it rendered as a topic. This is the default value for the `<keydef>` element.

If the `@processing-role` attribute is not specified locally, the value cascades from the closest element in the containment hierarchy.

Disposition: Unassigned

@relcolwidth (simpletable attributes)
Specifies the width of each column in relationship to the width of the other columns. The value is a space-separated list of relative column widths. Each column width is specified as a positive integer or decimal number followed by an asterisk character.

For example, the value `relcolwidth="1* 2* 3*"` gives a total of 6 units across three columns. The relative widths are 1/6, 2/6, and 3/6 (16.7%, 33.3%, and 50%). Similarly, the value `relcolwidth="90* 150*"` causes relative widths of 90/240 and 150/240 (37.5% and 62.5%).

@role
Specifies the role that a linked topic plays in relationship with the current topic.

For example, in a parent/child relationship, the role would be "parent" when the target is the parent of the current topic, and "child" when the target is the child of the current topic. This can be used to sort and classify links when rendering.

The following values are valid:

ancestor
Indicates a link to a topic above the parent topic.

child
Indicates a link to a direct child such as a directly nested or dependent topic.

cousin
Indicates a link to another topic in the same hierarchy that is not a parent, child, sibling, next, or previous.

descendant
Indicates a link to a topic below a child topic.

friend
Indicates a link to a similar topic that is not necessarily part of the same hierarchy.

next
Indicates a link to the next topic in a sequence.

other
Indicates any other kind of relationship or role. The type of role is specified as the value for the `@otherrole` attribute.

parent
Indicates a link to a topic that is a parent of the current topic.

previous
Indicates a link to the previous topic in a sequence.

sibling
Indicates a link between two children of the same parent topic.

-dita-use-conref-target
See STUB CONTENT for more information.

@rowheader (complex table attributes)
Specifies whether the entries in the respective column are row headers. The following values are valid:

firstcol
Indicates that entries in the first column of the table are row headers. This applies when the `@rowheader` attribute is specified on the `<table>` element.

headers
Indicates that entries of the column that is described using the `<colspec>` element are row headers. This applies when the `@rowheader` attribute is specified on the `<colspec>` element.

norowheader
Indicates that entries in the first column are not row headers. This applies when the `@rowheader` attribute is specified on the `<table>` element.

-dita-use-conref-target
See STUB CONTENT for more information.

Note This attribute is not part of the OASIS Exchange Table Model upon which DITA tables are based. Some processors or output formats might not support all values.

The `@rowheader` attribute is available on the following table elements: `<table>` and `<colspec>`.

@rowsep (complex table attributes)
Specifies whether to render row separators between table entries. The following values are valid: "0" (no separators) and "1" (separators).

The `@rowsep` attribute is available on the following table elements: `<colspec>`, `<entry>`, `<row>`, `<table>`, and `<tgroup>`.

@scale (display attributes)
Specifies the percentage by which fonts are resized in relation to the normal text size. The value of this attribute is a positive integer. When used on `<table>` or `<simpletable>`, the following values are valid: "50", "60", "70", "80", "90", "100", "110", "120", "140", "160", "180", "200", and -dita-use-conref-target.

This attribute is primarily useful for print-oriented display. Some processors might not support all values.

If the `@scale` attribute is specified on an element that contains an image, the image is not scaled. The image is scaled **only** if a scaling property is explicitly specified for the `<image>` element.

@scope (link-relationship attributes)
Specifies the closeness of the relationship between the current document and the referenced resource. The following values are valid: "local", "peer", "external", and "-dita-use-conref-target".

See STUB CONTENT for detailed information on supported values and processing implications.

@search (common map attributes)
Specifies whether the target is available for searching. If the value is not specified locally, the value might cascade from another element in the map (for cascade rules, see Cascading of metadata attributes in a DITA map). The following values are valid: "yes", "no", and "-dita-use-conref-target".

Comment by Kristen J Eberlein on 28 September 2022

Here is the content from the "DITA map attributes" topic:

@search
Specifies whether the topic is included in search indexes.

Disposition: Unassigned

@specializations (architectural attributes)
Specifies the attribute-domain specializations that are included in the document-type shell. This attribute is set as a default within the document-type shell. The value varies depending on what domains are integrated into the document-type shell. For example, a grammar file that includes the specialized attributes `@audience`, `@deliveryTarget`, and `@newBaseAtt` would set the value to `@props/audience @props/deliveryTarget @base/newBaseAtt`.

@subjectrefs (common map attributes)
Specifies one or more keys that are each defined by a subject definition in a subject scheme map. Multiple values are separated by white space.

@title-role (REQUIRED)
Specifies the role that the alternative title serves. Multiple roles are separated by white space. The following roles are defined in the specification: "linking", "navigation", "search", "subtitle", and "hint".

Processors can define custom values for the `@title-role` attribute.

@toc (common map attributes)

Specifies whether a topic appears in the table of contents (TOC) based on the current map context. If the value is not specified locally, the value might cascade from another element in the map (for cascade rules, see Cascading of metadata attributes in a DITA map). The following values are valid:

yes

The topic appears in a generated TOC.

no

The topic does not appear in a generated TOC.

-dita-use-conref-target

See STUB CONTENT for more information.

Comment by Kristen J Eberlein on 28 September 2022

Here is the content from the "DITA map attributes" topic:

@toc

Specifies whether topics are excluded from navigation output, such as a Web site map or an online table of contents. By default, `<topicref>` hierarchies are included in navigation output; relationship tables are excluded.

Disposition: Unassigned

@type (link-relationship attributes)

Describes the target of a reference. See STUB CONTENT for detailed information on supported values and processing implications.

@value (data-element attributes)

Specifies a value associated with the current property or element.

@valign (complex table attributes)

Specifies the vertical alignment of text in table entries. The following values are valid:

bottom

Indicates that text is aligned with the bottom of the table entry.

middle

Indicates that text is aligned with the middle of the table entry.

top

Indicates that text is aligned with the top of the table entry.

-dita-use-conref-target

See STUB CONTENT for more information.

The `@valign` attribute is available on the following table elements: `<entry>`, `<tbody>`, `<thead>`, and `<row>`.

@xml:space

Specifies how to handle white space in the current element. This attribute is provided on `<pre>`, `<lines>`, and on elements specialized from those. It ensures that parsers respect white space that is part of the data in those elements, including line-end characters. When defined, it has a fixed value of "preserve", making it a default property of the element that cannot be changed or deleted by authors.

@xmlns:ditaarch (architectural attributes)

Declares the default DITA namespace. Although this is a namespace rather than an attribute, it is specified as an attribute in the DTD-based grammar files that are distributed by OASIS. The value is fixed to "http://dita.oasis-open.org/architecture/2005/".

Comment by Kristen J Eberlein on 28 September 2022

Do we want to be precise and change grammar files to "element-definition module files?

Disposition: Unassigned

Index