Darwin Information Typing Architecture (DITA) for Technical Content Version 2.0

Working Draft 19

14 May 2024

This version:
N/A

Previous version:
N/A

Latest version:
N/A

Technical Committee:
OASIS Darwin Information Typing Architecture (DITA) TC

Chair:
Kristen James Eberlein (kris@eberleinconsulting.com), Eberlein Consulting LLC

Editors:
Kristen James Eberlein (kris@eberleinconsulting.com), Eberlein Consulting LLC
Robert D. Anderson (robert.dan.anderson@oracle.com), Oracle

Additional artifacts:
This prose specification is one component of a work product that also includes:

- X (DITA source)
- X (Grammar files)

Related work:
...

Abstract:
The Darwin Information Typing Architecture (DITA) for Technical Content Version 2.0 specification ...

Status:
This document was last revised or approved by the OASIS Darwin Information Typing Architecture (DITA) TC on the above date. The level of approval is also listed above. Check the “Latest stage” location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dita#technical.

TC members should send comments on this specification to the TC’s email list. Others should send comments to the TC’s public comment list, after subscribing to it by following the instructions at the “Send

Citation format:
When referencing this specification, the following citation format should be used:

[wp-abbrev]

Darwin Information Typing Architecture (DITA) for Technical Content Version 2.0. Edited by Kristen James Eberlein and Robert D. Anderson. 14 May 2024. Working Draft 19. X. Latest version: X.

Notices

Table of contents

1 Introduction

The techncial content part of DITA models the semantics of information types for technical content.

The technical content part of DITA adds the semantics of technical-content information-types to base DITA. This is done through several specializations:

- Topic specializations:

  Concept
  Glossary entry
  Reference
  Task
  Troubleshooting
- Domain specializations:

  Abbreviated form
  Equation
  Hazard statement
  Highlight
  Indexing
  Markup
  MathML
  Programming
  Release management
  Software
  SVG
  Syntax diagram
  User interface
  Utilities
  XML mention
- Bookmap map specialization

1.1 About DITA for Technical Content

The DITA technical content specification is designed for users who use information typing and document complex applications and devices, such as software, hardware, medical devices, and more.

1.1.1 Written specification

The specification is written for implementers of the DITA standard, including tool developers and XML architects who develop specializations.

The specification contains several parts:

- Introduction
- Architectural specification
- Language reference
- Conformance statement
- Appendices

The specification is available in the following formats:

- DITA source
- PDF
- HTML5 (available from the OASIS Web site, authoritative)
- ZIP of HTML5 (optimized for local use)

1.1.2 XML grammar files

The XML grammar files are available in RELAX NG (RNG), and XML Document-Type Definitions (DTD).

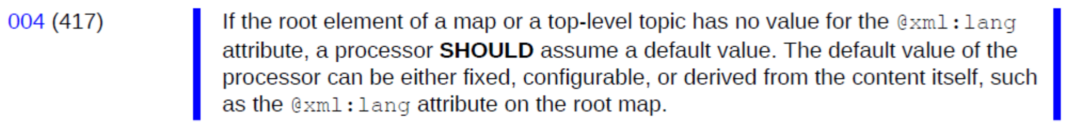
While the files should define the same DITA elements, the RELAX NG grammars are normative if there is a discrepancy.

1.2 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT, "RECOMMEND", "MAY", and "OPTIONAL" in this document are to be interpreted as described in **[RFC-2119]** (7) and **[RFC8174]** when, and only when, they appear in all capitals, as shown here.

The DITA specification uses `<keyword>` elements with the `@outputclass` attribute set to "RFC-2119" for these key words. In general, normative statements that use such key words pertain to what is needed for interoperability.

These key words are rendered with bold formatting. The normative statements are indicated visually in the rendered specification by blue lines at the left and right of the statement:



004 (417) If the root element of a map or a top-level topic has no value for the `@xml:lang` attribute, a processor **SHOULD** assume a default value. The default value of the processor can be either fixed, configurable, or derived from the content itself, such as the `@xml:lang` attribute on the root map.

In addition, a hyperlink is rendered to the left of the statement that contains the normative term. The link is to a generated appendix that groups all the normative statements that appear in the specification.

1.3 References

This section contains the normative and informative references that are used in this document.

While any hyperlinks included in this section were valid at the time of publication, OASIS cannot guarantee their long-term validity.

1.3.1 Normative references

The following documents are referenced in such a way that some or all of their content constitutes requirements of this document.

[RFC-2119]
Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC 3986]
Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC 5646]
Phillips, A., Ed., and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <http://www.rfc-editor.org/info/rfc5646>.

[RFC8174]
Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <http://www.rfc-editor.org/info/rfc8174>.

[XML 1.0]
Extensible Markup Language (XML) 1.0 (Fifth Edition), T Bray, J. Paoli, M. E. Maler, F. Yergeau, Editors, W3C Recommendation, 26 November 2008, http://www.w3.org/TR/2008/REC-xml-20081126/. Latest version available at http://www.w3.org/TR/xml.

[XML 1.1]
Extensible Markup Language (XML) 1.1 (Second Edition), T. Bray, J. Paoli, M. E. Maler, F. Yergeau, J. Cowan, Editors, W3C Recommendation, 16 August 2006, http://www.w3.org/TR/2006/REC-xml11-20060816/. Latest version available at http://www.w3.org/TR/xml11/.

1.3.2 Informative references

The following referenced documents are not required for the application of this document but might assist the reader with regard to a particular subject area.

[ANSI Z535.6]
Product Safety Information in Product Manuals, Instructions And Other Collateral Materials, https://webstore.ansi.org/Standards/NEMA/ansiz5352011r2017-1668876.

[HTML5]
HTML 5, Living Standard, https://html.spec.whatwg.org/.

[ISO 8601]
ISO/TC 154, *Data elements and interchange formats—Information interchange—Representation of dates and times*, 3rd edition, http://www.iso.org/iso/catalogue_detail?csnumber=40874, 12 December 2004.

[ISO/IEC 19757-3]
ISO/IEC JTC 1/SC 34 Document description and processing languages, *Information technology—Document Schema Definition Languages (DSDL)—Part 3: Rule-based validation—Schematron*, http://www.iso.org/iso/catalogue_detail.htm?csnumber=40833, 1 June 2006.

[Namespaces in XML 1.0]
Namespaces in XML 1.0 (Third Edition), T. Bray, D. Hollander, A. Layman, R. Tobin, H. S. Thompson, Editors, W3C Recommendation, 8 December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/. Latest version available at http://www.w3.org/TR/xml-names.

[Namespaces in XML 1.1]
Namespaces in XML 1.1 (Second Edition), T. Bray, D. Hollander, A. Layman, R. Tobin, Editors, W3C Recommendation, 16 August 2006, http://www.w3.org/TR/2006/REC-xml-names11-20060816/. Latest version available at http://www.w3.org/TR/xml-names11/.

[OASIS Table Model]
XML Exchange Table Model Document Type Definition. Edited by Norman Walsh, 1999. Technical Memorandum TR 9901:1999. https://www.oasis-open.org/specs/tm9901.htm.

[RELAX NG]
J. Clark and M. Murata, editors, *RELAX NG Specification*, http://www.oasis-open.org/committees/relax-ng/spec-20011203.html, OASIS Committee Specification, 3 December 2001.

[RELAX NG Compact Syntax]
J. Clark, editor, *RELAX NG Compact Syntax*, http://www.oasis-open.org/committees/relax-ng/compact-20021121.html, OASIS Committee Specification, 21 November 2002.

[RELAX NG DTD Compatibility]
J. Clark and M. Murata, editors, *RELAX NG DTD Compatibility*, http://www.oasis-open.org/committees/relax-ng/compatibility-20011203.html, OASIS Committee Specification, 3 December 2001.

[SVG 1.1]
Scalable Vector Graphics (SVG) Version 1.1 (Second) Edition), E. Dahlstrom, P. Dengler, A. Grasso, C. Lilley, C. McCormack, D. Schepers, J. Watt, Editors, W3C Recommendation, 16 August 2011, https://www.w3.org/TR/SVG11/.

[Unicode BiDi]
Unicode Bidirectional Algorithm, M. Davis, A. Lanin, A. Glass, Editors, Unicode Technical Report, 27 August 2021, https://www.unicode.org/reports/tr9/.

[WCAG 2.1]
Web Content Accessibility Guidelines (WCAG) Version 2.1, A. Kirkpatrick, J. O Connor, A. Campbell, M. Cooper, Editors, W3C Recommendation, 05 June 2018, https://www.w3.org/TR/WCAG21/.

[XHTML 1.0]
XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition), S. Pemberton, Editor, W3C Recommendation, 1 August 2002, http://www.w3.org/TR/2002/REC-xhtml1-20020801. Latest version available at http://www.w3.org/TR/xhtml1.

[XHTML 1.1]
XHTML™ 1.1 – Module-based XHTML – Second Edition, S. McCarron, M. Ishikawa, Editors, W3C Recommendation, 23 November 2010, http://www.w3.org/TR/2010/REC-xhtml11-20101123. Latest version available at http://www.w3.org/TR/xhtml11/.

[XPointer 1.0]
XML Pointer Language (XPointer), S. J. DeRose, R. Daniel, P. Grosso, E. Maler, J. Marsh, N. Walsh, Editors, W3C Working Draft (work in progress), 16 August 2002, http://www.w3.org/TR/2002/WD-xptr-20020816/. Latest version available at http://www.w3.org/TR/xptr/.

[XML Catalogs 1.1]
OASIS Standard, *XML Catalogs Version 1.1*, 7 October 2005, https://www.oasis-open.org/committees/download.php/14809/xml-catalogs.html.

[xml:tm 1.0]
A. Zydroń, R. Raya, and B. Bogacki, editors, *XML Text Memory (xml:tm) 1.0 Specification*, http://www.gala-global.org/oscarStandards/xml-tm/, The Localization Industry Standards Association (LISA) xml:tm 1.0, 26 February 2007.

[XSL 1.0]
Extensible Stylesheet Language (XSL) Version 1.0, S. Adler, A. Berglund, J. S. Deach, T. Graham, P. Grosso, E. Gutentag, A. Milowski, S. Parnell, J. Richman, S. Zilles, Editors, W3C Recommendation, 15 October 2001, http://www.w3.org/TR/2001/REC-xsl-20011015/. Latest version available at http://www.w3.org/TR/xsl/.

[XSL 1.1]
Extensible Stylesheet Language (XSL) Version 1.1, A. Berglund, Editor, W3C Recommendation, 5 December 2006, http://www.w3.org/TR/2006/REC-xsl11-20061205/. Latest version available at http://www.w3.org/TR/xsl11/.

[XSLT 2.0]
XSL Transformations (XSLT) Version 2.0, M. Kay, Editor, W3C Recommendation, 23 January 2007, http://www.w3.org/TR/2007/REC-xslt20-20070123/. Latest version available at http://www.w3.org/TR/xslt20.

[XSLT 3.0]
XSL Transformations (XSLT) Version 3.0, M. Kay, Editor,W3C Recommendation, 8 June 2017, https://www.w3.org/TR/xslt-30/.

[XTM 1.0]
S. Pepper and G. Moore, editors, *XML Topic Maps (XTM) 1.0*, http://www.topicmaps.org/xtm/index.html, TopicMaps.Org XTM 1.0, 2001.

1.4 Formatting conventions in the HTML5 version of the specification

Given the size and complexity of the specification, it is not generated as a single HTML5 file. Instead, each DITA topic is rendered as a separate HTML5 file.

The HTML5 version of the specification uses certain formatting conventions to aid readers in navigating through the specification and locating material easily: Link previews and navigation links.

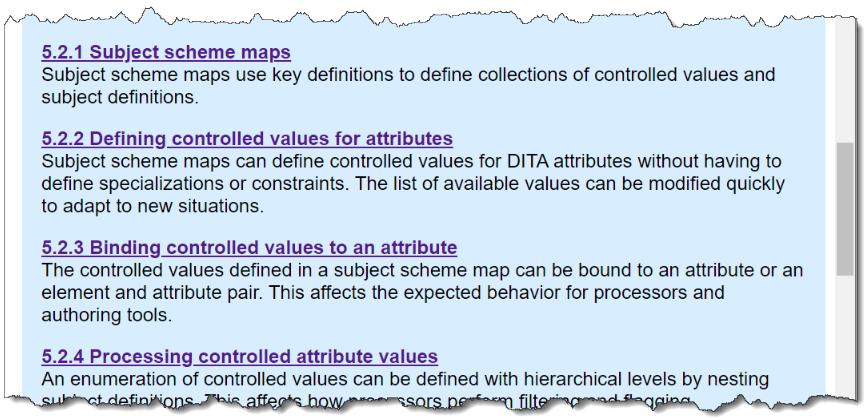
1.4.1 Link previews

The DITA specification uses the content of the DITA `<shortdesc>` element to provide link previews for its readers. These link previews are visually highlighted by a colored background.

The link previews serve as enhanced navigation aids, enabling readers to more easily locate content. This usability enhancement is one of the ways in which the specification illustrates the capabilities of DITA and exemplifies DITA best practices.

The following screen capture illustrates how link previews are displayed in the HTML5 version of the specification:

Figure 1: Link previews



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

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

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

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

1.4.2 Navigation links

To ease readers in navigating from one topic to another, each HTML5 file generated by a DITA topic contains navigation links at the bottom.

Parent topic
Takes readers to the parent topic, which is the topic referenced by the closest topic in the containment hierarchy

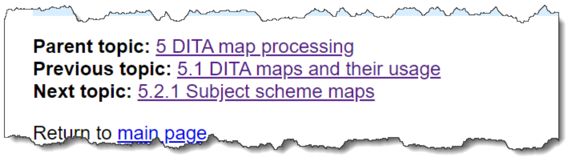
Previous topic
Takes readers to the previous topic in the reading sequence

Next topic
Takes readers to the next topic in the reading sequence

Return to main page
Takes readers to the place in the table of contents for the current topic in the reading sequence

The following screen capture illustrates how navigation links are displayed in the HTML5 version of the specification:

Figure 2: Navigation links


Parent topic: 5 DITA map processing
Previous topic: 5.1 DITA maps and their usage
Next topic: 5.2.1 Subject scheme maps

Return to main page

When readers hover over the navigation links, the short description of the DITA topic is also displayed.

2 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 book map document type.

2.1 Book map

The DITA bookmap specialization represents the key markup requirements for managing DITA content through book-oriented publication processes, including book metadata and book structures for organizing content.

Purpose

Book maps enable authors to produce documents that are structured like traditional print-oriented media. They also provide metadata for recording information about the book, including authors, owners, versions, and production history.

Content model

A book map can contain the following document structures:

- Titles
- Metadata
- DITAVAL references for filtering
- Map resources
- Front matter
- Parts
- Chapters
- Appendixes
- Back matter
- Relationship tables

Other components of a book map enable authors to specify that artifacts such as a table of contents (TOC) or an index should be generated.

The bookmap module has a dependency on two domains from the base specification. The `<mapresources>` element comes from the mapgroup-domain module, and can be used to group resources such as key definitions near the start of the map. The `<ditavalref>` element comes from the DITAVAL-reference domain module, and can be used to support branch filtering for the entire book.

Example

The following code sample contains some common markup for a book map:

```
<bookmap id="taskbook">
  <booktitle>
    <mainbooktitle>Product tasks</mainbooktitle>
    <booktitlealt>Tasks and what they do</booktitlealt>
  </booktitle>
  <bookmeta>
    <author>John Doe</author>
    <bookrights>
      <copyrfirst>
```

```
        <year>2020</year>
      </copyrfirst>
    </bookrights>
  </bookmeta>
  <mapresources>
    <!-- Key definitions used for the map -->
  </mapresources>
  <frontmatter>
    <preface href="task-preface.dita"/>
  </frontmatter>
  <chapter format="ditamap" href="installing.ditamap"/>
  <chapter href="configuring.dita"/>
  <chapter href="maintaining.dita">
    <topicref href="maintain-storage.dita"/>
    <topicref href="maintain-server.dita"/>
    <topicref href="maintain-database.dita"/>
  </chapter>
  <appendix href="task-appendix.dita"/>
</bookmap>
```

2.2 Concept

Concept topics are designed to provide conceptual or descriptive information.

Purpose

Concept topics serve a variety of purposes:

- Provide background information that helps readers understand essential facts about a product, process, or task
- Provide an extended definition of a major abstraction, such as a process or function
- Explain the nature and components of a product and describe how it fits into a category of products
- Help readers map their knowledge and understanding to the tasks that they need to perform

Content model

The body of a concept topic can contain the following document structures:

- Basic block elements: divisions, paragraphs, lists, tables, figures, etc.
- Concept body divisions: `<conbodydiv>` (66)
- Examples: `<example>`
- Sections: `<section>`

However, after a section, example, or concept body division is introduced into the topic structure, it can be followed only by a section, example, or concept body division.

This design supports the following best practices and use cases:

- Ensures that there are clear boundaries between sections
- Enables creation of short concept topics that only include a few basic block elements, for example, a paragraph, list, and image

Example

The following code sample contains a simple concept topic:

```
<concept id="color_triads">
  <title>Color triads</title>
  <shortdesc>A basic concept in color theory is the use of triads, colors that are three
             steps apart on the color wheel.</shortdesc>
  <conbody>
    <p>If you have difficulty picking colors when designing, you can use a color triad. Use
       one color as the main color and then the other two as accent colors.</p>
    <example>
      <p>The most simple color wheel contains 12 colors: red, red-orange, orange,
         yellow-orange, yellow, yellow-green, green, blue-green, blue, blue-violet,
         violet (purple), and red-violet.</p>
      <fig>
        <title>Basic color wheel</title>
        <image href="colorwheel.jpg" placement="break">
          <alt>Circle divided into twelve parts, each part a different color.</alt>
        </image>
      </fig>
      <p>Start with the first color green. Skip the next three colors, and the second color
         of the triad is violet. Skip the next three colors, and the third color of the triad
         is orange. This gives you the triad of green, violet, and orange.</p>
    </example>
  </conbody>
</concept>
```

2.3 Glossary entry

A glossary entry topic defines a single meaning of a term. It also can provide information such as acronyms and synonyms.

Purpose

Glossary entry topics serve the following purposes:

- They ensure that a team of writers can use the same terminology.
- They can be used to create glossaries that provide readers with definitions of terms and acronyms.
- They can be used, in conjunction with the `<abbreviated-form>` element, to enable processors to specify an acronym on second and later uses of a term.

Content model

Each glossary entry topic contains the following structures:

- Term: `<glossterm>` (73)
- Definition of term: `<glossdef>` (69)
- Glossary body: `<glossBody>` (68)

Examples

This section contains examples of glossary entry topics.

Figure 3: Simple glossary entry topic

The following code sample contains a simple glossary entry topic:

```
<glossentry id="ddl">
  <glossterm>data definition language</glossterm>
  <glossdef>A language used for defining database schemas</glossdef>
</glossentry>
```

Figure 4: Glossary entry topic used for acronym and acronym expansion

The following code sample contains a glossary entry topic that is used, in conjunction with an `<abbreviated-form>` element, to render the expanded form on first usage and the acronym on later usages.

```
<glossentry id="glossary-aids">
  <glossterm>acquired immunodeficiency syndrome</glossterm>
    <glossBody>
      <glossSurfaceForm>acquired immunodeficiency syndrome (AIDS)</glossSurfaceForm>
      <glossAlt>
        <glossAcronym>AIDS</glossAcronym>
      </glossAlt>
  </glossBody>
</hecsGlossentry>
```

Assume that a key-definition map specifies a value of "aids" for this glossary entry topic. On the first usage of `<abbreviated-form keyref="aids"/>`, the processor renders the content of the `<glossSurfaceForm>` element. On the second and later usages, the processor renders the content of the `<glossAcronum>` element.

2.4 Glossary group

Glossary group topics enable the authoring of glossary entries in a single topic document, rather than working with many individual glossary-entry topic documents.

Purpose

The glossary group topic serves as an authoring convenience. It enables people to author and manage multiple glossary-entry topics in a single DITA document.

Content model

A glossary group topic can contain multiple glossary-entry topics.

Example

The following code sample shows a glossary group topic with multiple nested glossary groups, one for each English letter group.

```
<glossgroup id="glossgroup" xml:lang="en-US">
  <title>Glossary</title>
  <glossgroup id="glossgroup-a">
    <title>A</title>
    <glossentry id="apple-fruit">
      <glossterm>apple</glossterm>
      <glossdef>A round, edible fruit produced by an apple tree (Malus domestica).</glossdef>
    </glossentry>
    <glossentry id="apple-corp">
      <glossterm>Apple Inc.</glossterm>
      <glossdef>An American multinational technology company headquartered in Cupertino, 
California.</glossdef>
    </glossentry>
  </glossgroup>
  ... (groups B to Y here ) ...
  <glossgroup id="glossgroup-z">
    <title>Z</title>
    <glossentry id="ziziphus-fruit">
      <glossterm>ziziphus</glossterm>
      <glossdef>The edible drupe of ziziphus shrubs (Ziziphus jujuba).</glossdef>
      <glossBody>
        <glossAlt>
          <glossSynonym>jujube</glossSynonym>
        </glossAlt>
      </glossBody>
    </glossentry>
  </glossgroup>
</glossgroup>
```

2.5 Reference

Reference topics contain reference information that users might need to consult occasionally, for example, product specifications, part lists, API calls, and programming language commands.

Purpose and usage

Reference topics serve the following purposes:

- Provide data that supports users as they perform a task
- Provide quick access to fact-based information
- Contain detailed information that users look up infrequently

Reference topics are used for the following types of information and more:

- API documentation
- Bibliographies
- Configuration file options
- Catalogs
- Element references
- Lists of equipment, ingredients, parts, and tools
- Specifications
- Syntax diagrams and explanations

Content model

The body of a reference topic can contain the following document structures:

- Examples: `<example>`
- Property lists: `<properties>` (76)
- Reference body divisions: `<refbodydiv>` (80)
- Sections: `<section>`
- Syntax sections: `<refsyn>` (82)
- Tables: `<simpletable>` and `<table>`

These structures can appear in any order or combination. However, basic document structure such as paragraphs, lists, and figures cannot be placed directly in the body of the reference topic. They must be contained within one of the structures listed above.

Example

The following code sample contains a simple reference topic:

Comment by Kristen J Eberlein on 26 October 2022

While the following code sample is a perfectly reasonable example of a properties list, I think we could provide a better example of a reference topic.

```
<reference id="oil-types">
  <title>Oil types</title>
  <shortdesc>The tables provide the recommended oil types.</shortdesc>
  <refbody>
    <properties>
      <prophead>
        <proptypehd>Oil type</proptypehd>
        <propvaluehd>Oil brand</propvaluehd>
        <propdeschd>Appropriate use</propdeschd>
      </prophead>
      <property>
        <proptype>Primary oil</proptype>
        <propvalue>A1X</propvalue>
        <propdesc>One-cylinder engines</propdesc>
      </property>
      <property>
        <proptype>Secondary oil</proptype>
        <propvalue>B2Z</propvalue>
        <propdesc>Two-cylinder engines</propdesc>
      </property>
    </properties>
  </refbody>
</reference>
```

2.6 Task

DITA offer two varieties of task topics: general task and strict task. Both task topics serve the same purpose: to provide users with comprehensive instructions for performing a task. Their content models vary, however.

2.6.1 General task

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

Content model

The general task topic is divided into three parts:

Introduction

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

- Prerequisites: `<prerequisites>` (90)
- Contextual information: `<context>` (88)
- Sections: `<section>`

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>` (92)
- Steps informal: `<steps-informal>` (92)
- Steps unordered: `<steps-unordered>` (95)

Post-instructions

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

1. Result: `<result>` (90)
2. Troubleshooting information: `<tasktroubleshooting>` (98)
3. Example: `<example>`
4. What to do next: `<postreq)>` (89)

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

The following code sample illustrates the relaxed content model of the general task topic. Note that there are multiple `<prereq>` elements and that they are preceded by a `<context>` element. The stylesheets used to generate output produce different labels for each of the `<prereq>` sections, triggered by the value of the `@outputclass` attribute.

```
<task id="changing_a_tire">
  <title>Changing a tire</title>
  <taskbody>
    <context><p>A flat tire typically shows up unexpectedly and catapults itself onto the
        top of your priority list. A flat tire can happen to anyone at any time. It doesn’t
        matter if you were already running late or if you’re wearing a cocktail dress.
        Regardless of your situation, the basic mechanics of changing a tire are the same
        whether you’re working with a car, truck, van, or SUV. </p>
    </context>
    <prereq outputclass="safety">
      <ul>
        <li>Find a safe place to pull over. The ground should be solid and
          level to keep your car from rolling. If you’re on the side of the
          road, pull over as far possible. Avoid stopping near any bends in
          the road, as this reduces visibility for both you and other
          drivers. If you’re in a dark or unsafe area, carefully drive to a
          better spot.</li>
        <li>Use your hazard lights and parking brake to keep yourself and
          your vehicle safe by increasing your visibility and decreasing
          the vehicle’s ability to roll.</li>
      </ul>
    </prereq>
    <prereq outputclass="tools"><p>At the bare minimum, you’ll need a jack, wrench, and
        a spare tire. These three items should always be in your vehicle just in case a
        flat tire occurs. Additional items that can make tire changing a little easier
        include:</p>
      <ul>
        <li>Flashlight</li>
        <li>Gloves</li>
        <li>Mat for kneeling</li>
        <li>Rain poncho</li>
        <li>Tire gauge</li>
        <li>Your vehicle’s owner’s manual</li>
      </ul>
    </prereq>
    <steps>
      <!-- ... -->
    </steps>
    <postreq>
      <!-- ... -->
    </postreq>
  </taskbody>
</task>
```

2.6.2 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 users to complete the task successfully.

Content model

The strict task topic is divided into three parts:

Introduction

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

1. Prerequisites: `<prerequisites>` (90)
2. Contextual information: `<context>` (88)

These sections are all optional, but they must occur in the outlined order.

Procedural instructions

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

- Steps: `<steps>` (92)
- Steps unordered: `<steps-unordered>` (95)

Post-instructions

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

1. Result: `<result>` (90)
2. Troubleshooting information: `<tasktroubleshooting>` (98)
3. Example: `<example>`
4. What to do next: `<postreq)>` (89)

These sections are all optional, but 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>
```

2.7 Troubleshooting

Troubleshooting topics are designed to document problems that people might encounter. They provide a topic structure that enables content authors to describe a condition, provide diagnostic information, discuss causes, and outline possible solutions.

Purpose

Troubleshooting topics serve the following purposes:

- Describe the problem condition, which usually is a state in a system, product, or service that a reader wants to correct
- Provide information that helps the reader diagnose the cause of the problem, if it is known
- Explain the cause of the problem and how to fix it

Content model

The troubleshooting topic is structured in three parts:

Condition

This section of the topic provides information about the problem condition, and it is specified by the `<condition>` element. This element is optional, as often the problematic condition can be adequately described in the title and short description.

Diagnostics

This optional section of the topic provides information about how to determine possible causes of the problem. It is specified by the `<diagnostics>` element, which must contain one or both of the following structural elements:

- General diagnostic information (`<diagnostics-general>` (102))
- Procedural diagnostic information (`<diagnostics-steps>` (104))

Trouble solution

This optional section of the topic provides information about possible causes and remedies for the problem. It is specified by `<troubleSolution>` elements.

Cause and remedy might occur in combinations other than pairs. It is possible to have:

- Multiple causes with the same remedy
- A single cause with more than one remedy
- A remedy with no known cause
- A cause with no known remedy

Examples

This section of the topics contains examples of troubleshooting topics.

Figure 5: Simple troubleshooting topic

The following code sample shows a simple troubleshooting topic. The title and short description describe the problem, and the single `<troubleSolution>` element explains the cause of the problem and how to remedy it.

```
<troubleshooting id="oasis-spec-not-rendered-correctly">
  <title>Specification PDF is not rendered correctly</title>
  <shortdesc>The specification URIs and notices appear in the TOC; they also appear
             twice in the body of the document.</shortdesc>
  <troublebody>
    <troubleSolution>
```

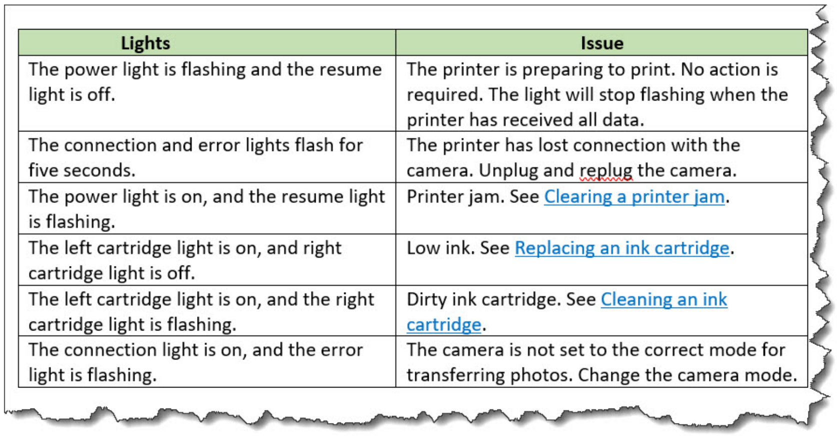
```
      <cause>
        <p>This problem occurs when the <xmlelement>notices</xmlelement> element
           for external publishing is not excluded.</p>
      </cause>
      <remedy>
        <steps>
          <step>
            <cmd>Use a DITAVAL file that excludes the <xmlelement>notices
                 platform="external-publishing-engine"</xmlelement> element
                 when you generate the PDF.</cmd>
          </step>
        </steps>
      </remedy>
    </troubleSolution>
  </troublebody>
</troubleshooting>
```

Figure 6: Complex troubleshooting topic

The following code sample shows a complex troubleshooting topic about "Blinking printer lights." It contains a `<diagnostics-general>` element that contains a table that outlines printer light conditions and possible remedies. It also includes several `<troubleSolution>` elements that reuse steps from other DITA topics.

```
troubleshooting id="blinking-lights">
  <title>Blinking printer lights</title>
  <troublebody>
  <condition><p>The indicator lights are blinking and you are unable to print.</p></condition>
  <diagnostics>
    <diagnostics-general><p>Use the following table to diagnose the problem.</p>
      <simpletable frame="all" id="light-diagnostics" relcolwidth="1* 2*">
        <sthead>
          <stentry>Lights</stentry>
          <stentry>Issue</stentry>
        </sthead>
        <strow>
          <stentry>The power light is flashing and the resume light is off./stentry>
          <stentry>The printer is preparing to print. No action is required. The light
                   will stop flashing when the printer has received all data.</stentry>
        </strow>
        <strow>
          <stentry>The connection and error lights flash for five seconds.</stentry>
          <stentry>The printer has lost connection with the camera. Unplug and replug the
                   camera.</stentry>
        </strow>
        <strow>
          <stentry>The power light is on, and the resume light is flashing.</stentry>
          <stentry>Printer jam. See <xref href="#./jam"/>.</stentry>
        </strow>
        <strow>
          <stentry>The left cartridge light is on, and right cartridge light is off.</stentry>
          <stentry>Low ink. See <xref href="#./ink"/>.</stentry>
        </strow>
        <strow>
          <stentry>The left cartridge light is on, and the right cartridge light is flashing.
          </stentry>
          <stentry>Dirty ink cartridge. See <xref href="#./clean"/>.</stentry>
        </strow>
        <strow>
          <stentry>The connection light is on, and the error light is flashing.</stentry>
          <stentry>The camera is not set to the correct mode for transferring photos. Change
                   the camera mode.</stentry>
        </strow>
      </simpletable>
    </diagnostics-general>
  </diagnostics>
  <troubleSolution>
    <remedy id="jam">
      <title>Clearing a paper jam </title>
      <steps conkeyref="clear-jam/steps">
        <step><cmd/></step>
      </steps>
    </remedy>
  </troubleSolution>
  <troubleSolution>
    <remedy id="clean">
      <title>Cleaning ink cartridges</title>
      <steps conkeyref="clean-cartridge/steps">
        <step><cmd/></step>
      </steps>
    </remedy>
  </troubleSolution>
  <troubleSolution>
    <remedy id="ink">
      <title>Replacing ink cartridges</title>
      <steps conkeyref="replace-ink/steps">
        <step><cmd/></step>
      </steps>
    </remedy>
    </troubleSolution>
  </troublebody>
</troubleshooting>
```

The table in the `<diagnostics-general>` element might be rendered in the following way. The hyperlinks in the "Issue" column resolve to the `<remedy>` elements in the topic.



Lights	Issue
The power light is flashing and the resume light is off.	The printer is preparing to print. No action is required. The light will stop flashing when the printer has received all data.
The connection and error lights flash for five seconds.	The printer has lost connection with the camera. Unplug and replug the camera.
The power light is on, and the resume light is flashing.	Printer jam. See Clearing a printer jam.
The left cartridge light is on, and right cartridge light is off.	Low ink. See Replacing an ink cartridge.
The left cartridge light is on, and the right cartridge light is flashing.	Dirty ink cartridge. See Cleaning an ink cartridge.
The connection light is on, and the error light is flashing.	The camera is not set to the correct mode for transferring photos. Change the camera mode.

Related reference

Troubleshooting elements (100)

Troubleshooting elements provide the fundamental structure for troubleshooting topics. Troubleshooting topics describe problems and provide information about how to fix them.

3 Usage of glossary entry topics

Glossary entry topics are designed for several purposes. In addition to providing the basis for glossaries, they also can be used in conjunction with the abbreviated-form domain to provide different renderings of terms on first and later usage.

3.1 Rendering of <abbreviated-form> elements

The `<abbreviated-form>` element is designed to work with glossary entry topics in specific ways.

The design intent is that processors use the following logic:

When a processor encounters an `<abbreviated-form>` element that references a DITA topic, if the referenced topic is not a `<glossentry>` topic or a specialization of `<glossentry>`, the title of the topic **SHOULD** be rendered.

When a processor encounters an `<abbreviated-form>` element that references a DITA topic, if the referenced topic is a `<glossentry>` topic or a specialization of `<glossentry>`, processors **SHOULD** render the `<abbreviated-form>` element in the following ways:

First usage
: Render the contents of the `<glossSurfaceForm>` elements, if it is not empty. If the `<glossSurfaceForm>` is empty or does not exist, render the contents of the `<glossterm>` element.

Second and later usage
: Render the contents of the `<glossAcronym>` elements, if it is not empty. If the `<glossAcronym>` is empty or does not exist, render the contents of the `<glossterm>` element.

See the examples (109) in the element-reference topic for the `<abbreviated-form>` element.

3.2 Localization and the<abbreviated-form> element

Implementations that choose to use the `<abbreviated-form>` need to consider the effect on translation.

Design of the specialization

The `<glossAcronym>` and `<glossSurfaceForm>` elements were designed to accommodate the following realities:

Acronyms do not exist in all languages

An acronym in one language might not have an equivalent in another language. In addition, languages have varying conventions for how an expanded form of a term is displayed. When acronyms are first displayed, some languages will display the expanded form followed by the acronym in parenthesis, while other languages do the reverse. For some acronyms, a translation might need to render both the original and the translated version of the acronym. The `<glossSurfaceForm>` enables authors and translators to present a locale-appropriate expanded form to the reader.

If a language does not have a acronym for a term, the translation of a glossary entry topic might result in an empty `<glossAcronym>` element.

Synonyms

Synonyms are specific to languages, so translation of a glossary entry topic might result in empty `<glossSynonym>` elements.

Translation quality issues

The use of `<abbreviated-form>` elements might have a negative effect on the quality and cost of the translation.

- Most translation tools do not resolve key references. Accordingly, translators might need to reference supplementary materials in order to understand the content that they are working with.
- Because many non-English languages vary articles based on gender and case, a simple programmatic insertion of an expanded form or acronym might yield translations that are ungrammatical and awkward.

4 Element reference

This section contains topics for each element defined in the technical content specializations. These elements include the original concept, task, and reference specializations, as well as specializations added in later releases . It also includes domains designed primarily for technical content.

4.1 Elements, A to Z

This section provides an alphabetized list of links to all elements in the specification.

textentity (153)
toc (44)
trademarklist (44)
troublebody (106)
troubleshooting (107)
troubleSolution (108)
tutorialinfo (99)
uicontrol (151)
userinput (134)
var (148)
varname (134)
volume (64)
wintitle (151)
xmlatt (154)
xmlelement (154)
xmlnsname (155)
xmlpi (156)
year (64)

4.2 Topic and map specializations

Content TBD

4.2.1 Book map elements

Elements in the book map section are used to organize DITA content into book form. They include elements for dividing up content, such as chapter and appendix, as well as metadata specific to publishing.

4.2.1.1 Book map content elements

The bookmap specialization of `<map>` supports standard book production for collections of DITA topics.

4.2.1.1.1 <abbrevlist>

The `<abbrevlist>` element references a list of abbreviations.

Processing expectations

When a list element in the bookmap module uses `@href` or `@keyref` to reference a topic or map, the referenced content provides the list content. When the list element does not reference a topic or map, a processor might generate a an appropriate list at the specified location in the map. For the `<abbrevlist>` element, a processor could generate a list of abbreviations used in the book.

Specialization hierarchy

The `<abbrevlist>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0).

For this element, the `@href` or `@keyref` attribute references a manual listing for the current element. If the element does not reference a topic or map, processors can choose to generate an appropriate listing for this element.

Example

The following code sample shows how to reference a list of abbreviations as part of a publication's back matter:

```
<backmatter>
  <booklists>
    <abbrevlist href="abbrev.dita"/>
    <indexlist/>
  </booklists>
</backmatter>
```

4.2.1.1.2 <amendments>

The `<amendments>` element either references a list of amendments or changes to the book, or it indicates to a processor that a list of changes should be generated at this location in the book.

Processing expectations

When a list element in the bookmap module uses `@href` or `@keyref` to reference a topic or map, the referenced content provides the list content. When the list element does not reference a topic or map, a processor might generate a an appropriate list at the specified location in the map. For the `<amendments>` element, a processor could generate a list of amendments or updates made to the book.

Specialization hierarchy

The `<amendments>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0).

For this element, the `@href` or `@keyref` attribute references a manual listing for the current element. If the element does not reference a topic or map, processors can choose to generate an appropriate listing for this element.

Example

The following code sample specifies that a change history list is generated in the publication front matter. The content of the change history list is contained in the DITA resource referenced by `@href`.

```
<frontmatter>
  <booklists>
    <amendments href="change-history.dita"/>
  </booklists>
</frontmatter>
```

4.2.1.1.3 <appendices>

The `<appendices>` element is an optional container for `<appendix>` elements.

Specialization hierarchy

The `<appendices>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0).

Example

The following code sample shows how the `<appendices>` element functions as a container to hold several appendix topics for an HTML publication:

```
<bookmap>
  <booktitle>
    <mainbooktitle>About operating the widget</mainbooktitle>
  </booktitle>
  <frontmatter>
    <!-- TOC, preface, and other front matter -->
  </frontmatter>
  <part href="introduction.dita">
    <!-- Part with introductory material about the widget -->
  </part>
  <part href="using.dita">
    <!-- Part about using the widget -->
  </part>
  <part href="troubleshooting.dita">
    <!-- Troubleshooting information -->
  </part>
  <appendices toc="yes">
    <topicmeta>
      <navtitle>Appendices</navtitle>
    </topicmeta>
    <appendix href="return-codes.dita"/>
    <appendix href="messages.dita"/>
    <appendix href="extra-info.dita"/>
  </appendices>
  <backmatter>
    <booklists><indexlist/></booklists>
  </backmatter>
</bookmap>
```

4.2.1.1.4 <appendix>

An `<appendix>` element references a topic or map as an appendix within a book.

Specialization hierarchy

The `<appendix>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0).

Example

The following code sample shows how two `<appendix>` elements are used to create two appendices for a publication:

```
<bookmap>
  <booktitle>
    <mainbooktitle>Developers guide to the widget</mainbooktitle>
  </booktitle>
  <frontmatter>
    <!-- TOC, preface, and other front matter -->
  </frontmatter>
  <chapter href="introduction.dita">
    <!-- Introductory material about the widget -->
  </chapter>
  <chapter href="extending.dita">
    <!-- How to extend the widget widget -->
  </chapter>
  <appendix href="intro.dita">
    <topicref href="caring.dita"/>
    <topicref href="feeding.dita"/>
  </appendix>
  <appendix href="setup.dita">
    <topicref href="prereq.dita"/>
    <topicref href="download.dita"/>
  </appendix>
</bookmap>
```

4.2.1.1.5 <backmatter>

The `<backmatter>` element is a container for material that follows the main body of a document and any appendices. Back matter might include items such as a colophon, legal notices, and book lists such as a glossary or an index.

Specialization hierarchy

The `<backmatter>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), universal attributes (173), `@format` (0), `@keyref` (0), `@scope` (0), and `@type` (0).

Example

See the example in bookmap (36).

4.2.1.1.6 <bibliolist>

The `<bibliolist>` element references a topic that contains a list of bibliographic entries for the publication.

Processing expectations

When a list element in the bookmap module uses `@href` or `@keyref` to reference a topic or map, the referenced content provides the list content. When the list element does not reference a topic or map, a processor might generate a an appropriate list at the specified location in the map. For the `<bibliolist>` element, a processor could generate a list of bibliographic entries used in the book.

Specialization hierarchy

The `<bibliolist>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0).

For this element, the `@href` or `@keyref` attribute references a manual listing for the current element. If the element does not reference a topic or map, processors can choose to generate an appropriate listing for this element.

Example

The following code sample shows how a `<bibliolist>` element is used to add a bibliography topic to the publication's back matter:

```
<bookmap>
  <!-- ... -->
  <backmatter>
    <amendments href="updatesToTheBook.dita"/>
    <booklists>
      <trademarklist href="listoftrademarks.dita"/>
      <bibliolist href="bibliography.dita"/>
      <indexlist/>
    </booklists>
  </backmatter>
</bookmap>
```

4.2.1.1.7 <bookabstract>

The `<bookabstract>` element references a topic that includes a brief summary of the book content.

Specialization hierarchy

The `<bookabstract>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0).

Example

The following code sample shows how a `<bookabstract>` element can be added to the front matter to provide a book summary:

```
<bookmap xml:lang="en-us">
  <booktitle>
    <booklibrary>Book library</booklibrary>
    <mainbooktitle>Book title</mainbooktitle>
  </booktitle>
  <!-- ... Book metadata here ... -->
  <frontmatter>
    <booklists>
      <toc/>
      <figurelist/>
      <tablelist/>
    </booklists>
    <bookabstract href="book-summary.dita"/>
    <preface href="preface.dita"></preface>
  </frontmatter>
  <!-- ... Book content followed by back matter ... -->
</bookmap>
```

4.2.1.1.8 <booklibrary>

The `<booklibrary>` element contains information about the library, series, or collection of documents to which a book belongs.

Specialization hierarchy

The `<booklibrary>` element is specialized from `<ph>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how to use `<booklibrary>` to include this book as part of a larger library:

```
<bookmap>
  <booktitle>
    <booklibrary>Construction Equipment</booklibrary>
    <mainbooktitle>Bulldozer Service Manual</mainbooktitle>
    <booktitlealt>Models 1234-5678</booktitlealt>
  </booktitle>
  <!-- ... metadata, front matter, and book content ... -->
</bookmap>
```

4.2.1.1.9 <booklist>

The `<booklist>` element either references a topic that contains a list of items in the book, or it indicates to a processor that a list of items should be generated at this location in the book. This is a general purpose element designed for use in specializations.

Usage information

The `<booklist>` element is a general purpose element that references a topic or map containing a list of items within the book. For example, it could be used to reference a topic that contains a list of authors for the book. When a more specific element is already available, such as `<tablelist>` for a list of tables, use that element instead. If the element does not reference a topic or map, a processor might be able to generate a list of items based on an `@outputclass` or based on specialization ancestry.

Processing expectations

When a list element in the bookmap module uses `@href` or `@keyref` to reference a topic or map, the referenced content provides the list content. When the list element does not reference a topic or map, a processor might generate a an appropriate list at the specified location in the map. Because `<booklist>` is a general purpose element, the type of list would need to be specified for the processor; for example, it could be provided using `@outputclass` or by further specializing the element.

Specialization hierarchy

The `<booklist>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0).

For this element, the `@href` or `@keyref` attribute references a manual listing for the current element. If the element does not reference a topic or map, processors can choose to generate an appropriate listing for this element.

Example

The following code sample shows how to use `<booklist>` to reference a topic that contains a list of authors of topics in this document:

```
<booklists>
  <toc/>
  <tablelist/>
  <booklist href="authors.dita"/>
</booklists>
```

4.2.1.1.10 <booklists>

The `<booklists>` element is a container for lists of various kinds within the book.

Processing expectations

The `<booklists>` element indicates to processors that lists are to be rendered or generated at that location in the front or back matter.

Specialization hierarchy

The `<booklists>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), universal attributes (173), `@format` (0), `@keyref` (0), `@scope` (0), and `@type` (0).

Example

The following code sample indicates that lists are generated in the front and back matter of the publication.

```
<bookmap>
  <booktitle>
    <mainbooktitle>Sample publication</mainbooktitle>
  </booktitle>
  <frontmatter>
    <booklists>
      <toc>
      <figurelist/>
      <tablelist/>
      <amendments/>
    </booklists>
  </frontmatter>
  ...
  <backmatter>
    <booklists>
      <abbrevlist/>
      <glossarylist/>
      <indexlist/>
    </booklists>
  </backmatter>
</bookmap>
```

4.2.1.1.11 <bookmap>

The `<bookmap>` element is a map specialization that is used to assemble DITA topics as a traditional book.

Usage information

Book maps consist of references to topics organized as book content. The topic references therefore are labeled according to the book components they point to, such as book title, front matter, chapter, and appendix.

Specialization hierarchy

The `<bookmap>` element is specialized from `<map>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: architectural attributes (178), common map attributes (178), universal attributes (173), `@format` (0), `@scope` (0), and `@type` (0).

Example

The following code sample shows how a `<bookmap>` can be used to organize content into a common book structure:

```
<bookmap xml:lang="en-us">
  <booktitle>
    <booklibrary>Book library</booklibrary>
    <mainbooktitle>Book title</mainbooktitle>
  </booktitle>
  <bookmeta>
    <bookrights>
      <copyrfirst><year>2019</year></copyrfirst>
      <copyrlast><year>2023</year></copyrlast>
      <bookowner>
        <organization>OASIS</organization>
      </bookowner>
    </bookrights>
  </bookmeta>
  <frontmatter>
    <booklists>
      <toc/>
      <figurelist/>
      <tablelist/>
    </booklists>
    <bookabstract href="MyBookAbstract.dita"/>
    <preface href="preface.dita"></preface>
  </frontmatter>
  <chapter href="chapter1.dita">
    <topicref href="subchap1.dita"></topicref>
  </chapter>
  <chapter href="chapter2.dita">
    <topicref href="subchap2.dita"></topicref>
  </chapter>
  <appendix href="app1.dita">
    <topicref href="insideApp1.dita"></topicref>
  </appendix>
  <appendix href="app2.dita">
    <topicref href="insideApp2.dita"></topicref>
  </appendix>
  <backmatter>
```

```
    <amendments href="RevisionHistoryTable.dita"/>
    <booklists>
      <glossarylist href="listofterms.dita"/>
      <trademarklist href="listoftrademarks.dita"/>
      <indexlist/>
    </booklists>
  </backmatter>
</bookmap>
```

4.2.1.1.12 <booktitle>

The `<booktitle>` element contains the title information for a book, including the library title, main title, and any other alternative titles.

Specialization hierarchy

The `<booktitle>` element is specialized from `<title>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: ID and conref attributes (174), localization attributes (174), `@base` (0), `@class` (0), `@outputclass` (0), and `@rev` (0).

Example

See the example in bookmap (36).

4.2.1.1.13 <booktitlealt>

The `<booktitlealt>` element contains an alternative title, subtitle, or short title for a book.

Specialization hierarchy

The `<booktitlealt>` element is specialized from `<ph>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows a `<booktitlealt>` element is used to provide a subtitle for the main book title:

```
<bookmap>
  <booktitle>
    <mainbooktitle>Product Z Installation, Operation, and Maintenance Manual</mainbooktitle>
    <booktitlealt>Getting to know Product Z</booktitlealt>
  </booktitle>
  <!-- ... -->
</bookmap>
```

4.2.1.1.14 <chapter>

The `<chapter>` element references a topic or map as a chapter within a book.

Specialization hierarchy

The `<chapter>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0).

Example

The following code sample shows how two `<chapter>` elements are used to create two chapters within a publication:

```
<bookmap>
  <booktitle>
    <mainbooktitle>A book about one thing</mainbooktitle>
  </booktitle>
  <!-- ... metadata and front matter ... -->
  <chapter href="introduction.dita">
    <topicref href="getting-the-one-thing.dita"/>
    <topicref href="preparing-to-use-the-thing.dita"/>
  </chapter>
  <chapter href="setup.dita">
    <topicref href="before-you-use-the-thing.dita"/>
    <topicref href="steps-to-start-using-the-thing.dita"/>
  </chapter>
  <!-- ... more chapters and back matter ... -->
</bookmap>
```

4.2.1.1.15 <colophon>

The `<colophon>` element references a topic that describes how the document was created.

Specialization hierarchy

The `<colophon>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0).

Example

The following code sample shows how the `<colophon>` element is used to add details about the book's production into the back matter:

```
<bookmap>
  <title>Sample book</title>
  <!-- ... -->
  <backmatter>
    <!-- ... other back matter ... -->
    <colophon href="ProductionNotes.dita" />
  </backmatter>
</bookmap>
```

4.2.1.1.16 <dedication>

The `<dedication>` element references a topic that contains a dedication for the book, such as to a person or group.

Specialization hierarchy

The `<dedication>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0   ).

Example

The following code sample shows how a `<dedication>` element is used to supply content for the dedication within the book's frontmatter.

```
<frontmatter>
  <dedication href="dedication-to-mother.dita"/>
  <!-- ... -->
</frontmatter>
```

4.2.1.1.17 <draftintro>

The `<draftintro>` element references a topic used as an introduction to the current draft of a book.

Specialization hierarchy

The `<draftintro>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0   ).

Example

The following code sample uses `<draftintro>` to provide an introductory draft section within the frontmatter of the book.

```
<frontmatter>
  <draftintro href="introducing-this-draft.dita"/>
  <!-- ... -->
</frontmatter>
```

4.2.1.1.18 <figurelist>

The `<figurelist>` element either references a topic that contains a list of figures in the book, or it indicates to a processor that a list of figures should be generated at this location in the book.

Processing expectations

When a list element in the bookmap module uses `@href` or `@keyref` to reference a topic or map, the referenced content provides the list content. When the list element does not reference a topic or map, a processor might generate a an appropriate list at the specified location in the map. For the `<figurelist>` element, a processor typically generates a list of figures used in the book.

Specialization hierarchy

The `<figurelist>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0).

For this element, the `@href` or `@keyref` attribute references a manual listing for the current element. If the element does not reference a topic or map, processors can choose to generate an appropriate listing for this element.

Example

See the example in bookmap (36).

4.2.1.1.19 <frontmatter>

The `<frontmatter>` element is a container for the material that precedes the main body of a document. Front matter might include items such as an abstract, a preface, and book lists such as a figure list or table of contents.

Specialization hierarchy

The `<frontmatter>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), universal attributes (173), `@format` (0), `@keyref` (0), `@scope` (0), and `@type` (0).

Example

See the example in bookmap (36).

4.2.1.1.20 <glossarylist>

The `<glossarylist>` element either references a topic or map that contains a list of glossary entries in the book, or it indicates to a processor that a list of glossary entries should be generated at this location in the book.

Processing expectations

When a list element in the bookmap module uses `@href` or `@keyref` to reference a topic or map, the referenced content provides the list content. When the list element does not reference a topic or map, a processor might generate a an appropriate list at the specified location in the map. For the `<glossarylist>` element, a processor could generate a list of glossary entries used in the book.

Specialization hierarchy

The `<glossarylist>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0).

For this element, the `@href` or `@keyref` attribute references a manual listing for the current element. If the element does not reference a topic or map, processors can choose to generate an appropriate listing for this element.

Example

See the example in bookmap (36).

4.2.1.1.21 <indexlist>

The `<indexlist>` element either references a topic that contains an index for the book, or it indicates to a processor that an index should be generated at this location in the book.

Processing expectations

When a list element in the bookmap module uses `@href` or `@keyref` to reference a topic or map, the referenced content provides the list content. When the list element does not reference a topic or map, a processor might generate a an appropriate list at the specified location in the map. For the `<indexlist>` element, a processor typically generates an index based on index terms used in the book.

Specialization hierarchy

The `<indexlist>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0).

For this element, the `@href` or `@keyref` attribute references a manual listing for the current element. If the element does not reference a topic or map, processors can choose to generate an appropriate listing for this element.

Example

See the example in bookmap (36).

4.2.1.1.22 <mainbooktitle>

The `<mainbooktitle>` element contains the primary title for a book.

Specialization hierarchy

The `<mainbooktitle>` element is specialized from `<ph>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how to use `<mainbooktitle>` to provide the main title of the book:

```
<bookmap>
  <booktitle>
    <booklibrary>Construction Equipment</booklibrary>
    <mainbooktitle>Bulldozer Service Manual</mainbooktitle>
    <booktitlealt>Models 1234-5678</booktitlealt>
  </booktitle>
  <!-- ... metadata, front matter, and book content ... -->
</bookmap>
```

4.2.1.1.23 <notices>

The `<notices>` element references a topic that contains special notice information, such as legal notices about supplementary copyrights and trademarks associated with the book.

Specialization hierarchy

The `<notices>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0).

Example

The following code sample shows how to use a `<notices>` element to reference legal content:

```
<backmatter>
  <notices href="legal-notices.dita"/>
  <!-- ... Other back matter ... -->
</backmatter>
```

4.2.1.1.24 <part>

The `<part>` element references a topic or a map that acts as a part within a book.

Usage information

Use `<part>` to divide a document's chapters into logical groupings. For example, in a document that contains both guide and reference information, you can define two parts, one containing the guide information and the other containing the reference information.

Specialization hierarchy

The `<part>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0).

Example

The following code sample shows how `<part>` elements are used to group chapters in order to divide a book into two major sections for task and reference material:

```
<bookmap>
  <title>Using and maintaining Product Zed</title>
  <!-- ... metadata and front matter ... ->
  <part href="taskguide.dita">
    <chapter href="intro.dita">
      <topicref href="caring.dita"/>
      <topicref href="feeding.dita"/>
    </chapter>
    <chapter href="setup.dita">
      <topicref href="prereq.dita"/>
      <topicref href="download.dita"/>
    </chapter>
  </part>
  <part href="reference.dita">
    <chapter href="commands.dita">
      <topicref href="care.dita"/>
      <topicref href="feed.dita"/>
    </chapter>
    <chapter href="apis.dita">
      <topicref href="acare.dita"/>
      <topicref href="afeed.dita"/>
    </chapter>
  </part>
</bookmap>
```

4.2.1.1.25 <preface>

The `<preface>` element references a topic or map that contains introductory information about a book, such as the purpose and structure of the document.

Specialization hierarchy

The `<preface>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0).

Example

See the example in bookmap (36).

4.2.1.1.26 <tablelist>

The `<tablelist>` element either references a topic that contains a list of tables in the book, or it indicates to a processor that a list of tables should be generated at this location in the book.

Processing expectations

When a list element in the bookmap module uses `@href` or `@keyref` to reference a topic or map, the referenced content provides the list content. When the list element does not reference a topic or map, a processor might generate a an appropriate list at the specified location in the map. For the `<tablelist>` element, a processor typically generates a list of tables used in the book.

When the `@href` attribute is not specified, a processor might generate a list of tables at the specified location in the map.

Specialization hierarchy

The `<tablelist>` element is specialized from `<topicref>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0).

For this element, the @href or @keyref attribute references a manual listing for the current element. If the element does not reference a topic or map, processors can choose to generate an appropriate listing for this element.

Example

See the example in bookmap (36).

4.2.1.1.27 <toc>

The <toc> element either references a topic that contains a table of contents for the book, or it indicates to a processor that a table of contents should be generated at this location in the book.

Processing expectations

When a list element in the bookmap module uses @href or @keyref to reference a topic or map, the referenced content provides the list content. When the list element does not reference a topic or map, a processor might generate a an appropriate list at the specified location in the map. For the <toc> element, a processor typically generates a table of contents based on topics used in the book.

Specialization hierarchy

The <toc> element is specialized from <topicref>. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and @keyref (0).

For this element, the @href or @keyref attribute references a manual listing for the current element. If the element does not reference a topic or map, processors can choose to generate an appropriate listing for this element.

Example

See the example in bookmap (36).

4.2.1.1.28 <trademarklist>

The <trademarklist> element either references a topic that contains a list of trademarks in the book, or it indicates to a processor that a list of trademarks should be generated at this location in the book.

Processing expectations

When a list element in the bookmap module uses @href or @keyref to reference a topic or map, the referenced content provides the list content. When the list element does not reference a topic or map, a processor might generate a an appropriate list at the specified location in the map. For the <trademarklist> element, a processor could generate a list of trademarks used in the book.

Specialization hierarchy

The <trademarklist> element is specialized from <topicref>. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: common map attributes (178), link-relationship attributes (179), universal attributes (173), and `@keyref` (0   ).

For this element, the `@href` or `@keyref` attribute references a manual listing for the current element. If the element does not reference a topic or map, processors can choose to generate an appropriate listing for this element.

Example

See the example in bookmap (36).

4.2.1.2 Book map metadata elements

The book map specialization supports standard book production. This section contains the metadata elements used by bookmap to store book-related metadata.

4.2.1.2.1 <approved>

The `<approved>` element contains detailed information about a book approval, such as the revision that was approved, who approved the book, and when the approval occurred.

Usage information

Information within `<approval>` can apply to different aspects of the map, such as approval of the overall content or of a specific deliverable created from the map.

Specialization hierarchy

The `<approved>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following code sample shows how the `<approved>` element can be used to specify who approved the book and when it was approved:

```
<bookmeta>
  <bookchangehistory>
    <reviewed>
      <revisionid>2</revisionid>
      <started><year>2019</year><month>10</month></started>
      <completed><year>2020</year><month>01</month></completed>
    </reviewed>
    <approved>
      <organization>OASIS</organization>
      <completed><year>2020</year><month>05</month></completed>
    </approved>
  </bookchangehistory>
</bookmeta>
```

4.2.1.2.2 <bookchangehistory>

The `<bookchangehistory>` element contains publishing life-cycle information about the book.

Specialization hierarchy

The `<bookchangehistory>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following code sample shows how the `<bookchangehistory>` element can be used to specify details about when the content was reviewed, edited, tested, approved, and indexed:

```
<bookmeta>
  <bookchangehistory>
    <reviewed>
      <revisionid>2</revisionid>
      <started><year>2019</year><month>10</month></started>
      <completed><year>2020</year><month>01</month></completed>
    </reviewed>
    <edited>
      <revisionid>1</revisionid>
      <person>Joe T. Editor</person>
      <completed><year>2019</year><month>03</month><day>15</day></completed>
      <summary>Corrected grammatical errors.</summary>
    </edited>
    <edited>
      <revisionid>3</revisionid>
      <person>Joe T. Editor</person>
      <completed><year>2022</year><month>06</month><day>30</day></completed>
    </edited>
    <tested>
      <organization>OASIS</organization>
      <completed><year>2023</year><month>04</month></completed>
    </tested>
    <approved>
      <organization>OASIS</organization>
      <completed><year>2023</year><month>08</month></completed>
    </approved>
    <bookevent>
      <bookeventtype name="indexed"/>
      <completed><year>2023</year><month>01</month></completed>
    </bookevent>
  </bookchangehistory>
</bookmeta>
```

4.2.1.2.3 <bookevent>

The `<bookevent>` element contains detailed information about a custom book event, such as the type of event, which book revision was part of the event, who was responsible for the event, and when the event occurred.

Usage information

This element is appropriate for specialization if the existing `<reviewed>`, `<edited>`, or `<approved>` event type elements do not meet the needs of the organization.

Specialization hierarchy

The `<bookevent>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following code sample shows how the `<bookevent>` element can be used to specify that the book was indexed and when indexing was completed:

```
<bookmeta>
   <bookchangehistory>
      <bookevent>
         <bookeventtype name="indexed"/>
         <completed><month>09</month><year>2022</year></completed>
      </bookevent>
   </bookchangehistory>
</bookmeta>
```

4.2.1.2.4 <bookeventtype>

The `<bookeventtype>` element indicates a custom publication event, for example, updated, indexed, or deprecated.

Usage information

The `@value` attribute is used to indicate the event type.

Specialization hierarchy

The `<bookeventtype>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

For this element, the `@value` attribute specifies the type of book event.

Example

The following code sample shows how the `<bookeventtype>` element can be used to specify a custom publication event:

```
<bookmeta>
   <bookchangehistory>
      <bookevent>
         <bookeventtype name="indexed"/>
          <completed><month>09</month><year>2022</year></completed>
      </bookevent>
   </bookchangehistory>
</bookmeta>
```

4.2.1.2.5 <bookid>

The `<bookid>` element contains publishing information used to identify the book, such as part number, edition number, or ISBN number.

Specialization hierarchy

The `<bookid>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following code sample shows how the `<bookid>` element can be used to define detailed information that identifies the book:

```
<bookmeta>
  <bookid>
    <bookpartno>99F1234</bookpartno>
    <edition>Second</edition>
    <isbn>123-0-456-12345-1</isbn>
    <booknumber>SC21-1234-00</booknumber>
    <volume>2</volume>
    <maintainer>
      <person>John Smith</person>
    </maintainer>
  </bookid>
</bookmeta>
```

4.2.1.2.6 <bookmeta>

The `<bookmeta>` element contains metadata about the book, such as information related to publishing, copyright details, and book identification.

Specialization hierarchy

The `<bookmeta>` element is specialized from `<topicmeta>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

The following code sample shows how the `<bookmeta>` element can be used to specify publishing details, book identification details, and copyright details.

```
<bookmeta>
  <publisherinformation>
      <organization>NY Publishing</organization>
      <printlocation>United States of America</printlocation>
      <published>
        <publishtype value="general"/>
        <revisionid>2</revisionid>
        <started><month>01</month><year>2020</year></started>
        <completed><month>02</month><year>2023</year></completed>
      </published>
  </publisherinformation>
```

```
    <bookid>
     <bookpartno>99F1234</bookpartno>
     <edition>Second</edition>
     <isbn>123-0-456-12345-1</isbn>
     <booknumber>SC21-1234-00</booknumber>
     <volume>2</volume>
     <maintainer>
       <person>John Smith</person>
     </maintainer>
   </bookid>
   <bookchangehistory>
     <reviewed>
       <person>Jack</person>
        <revisionid>1</revisionid>
       <completed><day>31</day><month>07</month><year>2022</year></completed>
     </reviewed>
     <edited>
       <organization>XYZ Editing</organization>
       <completed><day>18</day><month>01</month><year>2023</year></completed>
     </edited>
     <approved>
       <organization>OASIS</organization>
     </approved>
     <bookevent>
       <bookeventtype name="indexed"/>
       <completed><month>09</month><year>2022</year></completed>
     </bookevent>
   </bookchangehistory>
   <bookrights>
     <copyrfirst><year>2020</year></copyrfirst>
     <copyrlast><year>2023</year></copyrlast>
     <bookowner><organization>OASIS</organization></bookowner>
     <bookrestriction value="unclassified"/>
   </bookrights>
 </bookmeta>
```

4.2.1.2.7 <booknumber>

The `<booknumber>` element contains the number used to identify a book that is part of a collection of works that belong to the same author.

Comment by tammy
The original description states that <booknumber> is the "book's form number" while the original examples states that that it "is a number that identifies this book among all of the author's works". As noted in the current short description, I opted to go with the latter definition. However, I don't actually know if either is accurate. It's not an element I've ever had the need to use.

Specialization hierarchy

The `<booknumber>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following code sample shows how the `<booknumber>` element can be used to identify this book among the author's works:

```
<bookmeta>
  <bookid>
    <bookpartno>99F1234</bookpartno>
    <edition>Second</edition>
    <isbn>123-0-456-12345-1</isbn>
    <booknumber>SC21-1234-00</booknumber>
    <volume>2</volume>
    <maintainer>
      <person>John Smith</person>
    </maintainer>
  </bookid>
</bookmeta>
```

4.2.1.2.8 <bookowner>

The `<bookowner>` element specifies the person (`<person>`) or business unit (`<organization>`) that owns the copyrights to the book.

Specialization hierarchy

The `<bookowner>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following code sample shows how the `<bookowner>` element can be used to specify the business unit that owns the copyrights to the book:

```
<bookmeta>
  <bookrights>
    <copyrfirst><year>2020</year></copyrfirst>
    <copyrlast><year>2023</year></copyrlast>
    <bookowner>
      <organization>OASIS</organization>
    </bookowner>
    <bookrestriction value="unclassified"/>
  </bookrights>
</bookmeta>
```

4.2.1.2.9 <bookpartno>

The `<bookpartno>` element contains the part number of the book, such as 99F1234.

Comment by tammy
I don't fully understand how this element is to be used. Usage information suggests that it can be used by the publisher. Does it have other uses?

Usage information

A publisher may use the `<bookpartno>` element to identify a book for tracking purposes.

Specialization hierarchy

The `<bookpartno>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following code sample shows how the `<bookpartno>` element can be used to identify the part number of the book:

```
<bookmeta>
  <bookid>
    <bookpartno>99F1234</bookpartno>
    <edition>Second</edition>
    <isbn>123-0-456-12345-1</isbn>
    <booknumber>SC21-1234-00</booknumber>
    <volume>2</volume>
  </bookid>
</bookmeta>
```

4.2.1.2.10 <bookrestriction>

The `<bookrestriction>` element specifies whether the book is classified or restricted in some way.

Usage information

The `@value` attribute is required to specify whether there is a restriction on the book.

Specialization hierarchy

The `<bookrestriction>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

For this element, the `@value` attribute specifies any restrictions on the use of the material, such as declaring the information confidential or for licensed use only.

Comment by tammy
This seems redundant since it's also captured in the Usage information section.

Example

The following code sample shows how the `<bookrestriction>` element can be used to specify the book restriction:

```
<bookrights>
  <copyrfirst><year>2016</year></copyrfirst>
  <copyrlast><year>2020</year></copyrlast>
  <bookowner><organization>Example Corporation</organization></bookowner>
  <bookrestriction value="unclassified"/>
</bookrights>
```

4.2.1.2.11 <bookrights>

The `<bookrights>` element contains information about the legal rights associated with the book, including copyright dates and owners.

Specialization hierarchy

The `<bookrights>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following code sample shows how the `<bookrights>` element can be used to specify copyright information, who owns the book, and whether there are restrictions in place:

```
<bookmeta>
  <bookrights>
    <copyrfirst><year>2020</year></copyrfirst>
    <copyrlast><year>2023</year></copyrlast>
    <bookowner>
      <organization>OASIS</organization>
    </bookowner>
    <bookrestriction value="unclassified"/>
  </bookrights>
</bookmeta>
```

4.2.1.2.12 <completed>

The `<completed>` element indicates when a book event ended.

Specialization hierarchy

The `<completed>` element is specialized from `<ph>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how the `<completed>` element can be used to specify when publishing and editing ended:

```
<bookmeta>
  <publisherinformation>
      <organization>NY Publishing</organization>
      <printlocation>United States of America</printlocation>
      <published>
        <publishtype value="general"/>
        <revisionid>2</revisionid>
        <started><month>11</month><year>2022</year></started>
        <completed>
          <month>02</month><year>2023</year>
        </completed>
      </published>
  </publisherinformation>
  <bookchangehistory>
    <edited>
```

```
      <organization>XYZ Editing</organization>
      <started><month>08</month><year>2022</year></started>
      <completed>
        <month>10</month><year>2022</year>
      </completed>
    </edited>
</bookmeta>
```

4.2.1.2.13 <copyrfirst>

The `<copyrfirst>` element contains the copyright year, or the first copyright year within a multi-year copyright statement.

Specialization hierarchy

The `<copyrfirst>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following code sample shows how the `<copyrfirst>` element can be used to specify the first copyright year:

```
<bookmeta>
  <bookrights>
    <copyrfirst>
      <year>2020</year>
    </copyrfirst>
    <copyrlast><year>2023</year></copyrlast>
    <bookowner>
      <organization>OASIS</organization>
    </bookowner>
    <bookrestriction value="unclassified"/>
  </bookrights>
</bookmeta>
```

4.2.1.2.14 <copyrlast>

The `<copyrlast>` element contains the last copyright year within a multi-year copyright statement.

Specialization hierarchy

The `<copyrlast>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following code sample shows how the `<copyrlast>` element can be used to specify the last copyright year:

```
<bookmeta>
  <bookrights>
    <copyrfirst><year>2020</year></copyrfirst>
    <copyrlast>
      <year>2023</year>
    </copyrlast>
    <bookowner>
      <organization>OASIS</organization>
    </bookowner>
    <bookrestriction value="unclassified"/>
  </bookrights>
</bookmeta>
```

4.2.1.2.15 <day>

The `<day>` element denotes a day of the month.

Specialization hierarchy

The `<day>` element is specialized from `<ph>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

See the example in bookmeta (48).

4.2.1.2.16 <edited>

The `<edited>` element contains detailed information about a book edit, such as the revision that was edited, who completed the edit, and when the edit occurred.

Specialization hierarchy

The `<edited>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following code sample shows how the `<edited>` element can be used to show which revisions of the book were edited, who completed the edits, and when the edits occurred:

```
<bookmeta>
  <bookchangehistory>
    <reviewed>
      <revisionid>2</revisionid>
      <started><year>2019</year><month>10</month></started>
      <completed><year>2020</year><month>01</month></completed>
    </reviewed>
   <edited>
      <revisionid>1</revisionid>
      <person>Joe T. Editor</person>
      <completed><year>2020</year><month>03</month><day>15</day></completed>
    </edited>
    <edited>
      <revisionid>2</revisionid>
      <person>Joe T. Editor</person>
      <completed><year>2023</year><month>06</month><day>30</day></completed>
    </edited>
  </bookchangehistory>
</bookmeta>
```

4.2.1.2.17 <edition>

The `<edition>` element contains information used by a publisher to identify the edition of the book, such as First or Third edition.

Specialization hierarchy

The `<edition>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following example shows how the `<edition>` element can be used to specify the edition of the book:

```
<bookmeta>
  <bookid>
    <bookpartno>99F1234</bookpartno>
    <edition>Second</edition>
    <isbn>123-0-456-12345-1</isbn>
    <booknumber>SC21-1234-00</booknumber>
    <volume>2</volume>
    <maintainer>
      <person>John Smith</person>
    </maintainer>
  </bookid>
</bookmeta>
```

4.2.1.2.18 <isbn>

The `<isbn>` element contains the International Standard Book Number (ISBN) for the book.

Specialization hierarchy

The `<isbn>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following example shows how the `<isbn>` element can be used to show to specify the ISBN for the book:

```
<bookmeta>
  <bookid>
    <bookpartno>99F1234</bookpartno>
    <edition>Second</edition>
    <isbn>123-0-456-12345-1</isbn>
    <booknumber>SC21-1234-00</booknumber>
    <volume>2</volume>
    <maintainer>
```

```
      <person>John Smith</person>
    </maintainer>
  </bookid>
</bookmeta>
```

4.2.1.2.19 <maintainer>

The `<maintainer>` element contains the name of the person (`<person>`) or business unit (`<organization>`) that maintains the book.

Specialization hierarchy

The `<maintainer>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following example shows how the `<maintainer>` element can be used identify the person that maintains the book:

```
<bookmeta>
  <bookid>
    <isbn>123-0-456-12345-1</isbn>
    <booknumber>SC21-1234-00</booknumber>
    <maintainer>
      <person>John Smith</person>
    </maintainer>
  </bookid>
</bookmeta>
```

4.2.1.2.20 <month>

The `<month>` element denotes a month of the year.

Comment by tammy
Are there any expectations to provide the month number vs. the name?

Specialization hierarchy

The `<month>` element is specialized from `<ph>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

See the example in bookmeta (48).

4.2.1.2.21 <organization>

The `<organization>` element contains the name of a business unit.

Specialization hierarchy

The `<organization>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following code sample shows how the `<organization>` element can be used to specify the business units responsible for publishing, editing, and approving the book as well as the business unit that owns the copyrights to the book:

```
<bookmeta>
  <publisherinformation>
      <organization>NY Publishing</organization>
      <printlocation>United States of America</printlocation>
  </publisherinformation>
   <bookid>
    <isbn>123-0-456-12345-1</isbn>
    <maintainer>
      <person>John Smith</person>
    </maintainer>
  </bookid>
  <bookchangehistory>
    <reviewed>
      <person>Jack</person>
      <completed><month>July</month><year>2022</year></completed>
    </reviewed>
    <edited>
      <organization>XYZ Editing</organization>
      <completed><month>January</month><year>2023</year></completed>
    </edited>
    <approved>
      <organization>OASIS</organization>
    </approved>
  </bookchangehistory>
  <bookrights>
    <copyrfirst><year>2020</year></copyrfirst>
    <copyrlast><year>2023</year></copyrlast>
    <bookowner>
      <organization>OASIS</organization>
    </bookowner>
  </bookrights>
</bookmeta>
```

4.2.1.2.22 <person>

The `<person>` element contains name of a person.

Specialization hierarchy

The `<person>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following code sample shows how the `<person>` element can be used to specify who maintains the book and who reviewed the book:

```
<bookmeta>
  <publisherinformation>
      <organization>NY Publishing</organization>
      <printlocation>United States of America</printlocation>
  </publisherinformation>
   <bookid>
    <isbn>123-0-456-12345-1</isbn>
    <maintainer>
      <person>John Smith</person>
    </maintainer>
  </bookid>
  <bookchangehistory>
    <reviewed>
      <person>Jack</person>
      <completed><month>July</month><year>2022</year></completed>
    </reviewed>
    <edited>
      <organization>XYZ Editing</organization>
      <completed><month>January</month><year>2023</year></completed>
    </edited>
    <approved>
      <organization>OASIS</organization>
    </approved>
  </bookchangehistory>
  <bookrights>
    <copyrfirst><year>2020</year></copyrfirst>
    <copyrlast><year>2023</year></copyrlast>
    <bookowner><organization>OASIS</organization></bookowner>
  </bookrights>
</bookmeta>
```

4.2.1.2.23 <printlocation>

The `<printlocation>` element indicates where the book was printed.

Usage information

Typically the print location includes only the country in which the book was printed.

Specialization hierarchy

The `<printlocation>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following code sample shows how the `<printlocation>` element can be used to indicate where the book is printed:

```
<bookmeta>
  <publisherinformation>
      <organization>NY Publishing</organization>
      <printlocation>United States of America</printlocation>
      <published>
        <publishtype value="general"/>
        <completed><year>2023</year></completed>
      </published>
  </publisherinformation>
</bookmeta>
```

4.2.1.2.24 <published>

The `<published>` element contains information about the publication, such as the published revision, the publication type, and when the book was published.

Specialization hierarchy

The `<published>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following code sample shows how the `<published>` element can be used to specify the publication type, the published revision, and when the book was published:

```
<bookmeta>
  <publisherinformation>
      <organization>NY Publishing</organization>
      <printlocation>United States of America</printlocation>
      <published>
        <publishtype value="general"/>
        <revisionid>2</revisionid>
        <completed><month>02</month><year>2023</year></completed>
      </published>
  </publisherinformation>
</bookmeta>
```

4.2.1.2.25 <publisherinformation>

The `<publisherinformation>` element contains information about who published the book, where it was printed, the publication type, which revision was published, and when it was published.

Usage information

Additional information about the publication history is found in the `<bookchangehistory>` element.

Comment by tammy
I think this can be removed. Thoughts?

Specialization hierarchy

The `<publisherinformation>` element is specialized from `<publisher>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following code sample shows how the `<publisherinformation>` element can be used to identity who published the book, where the book was printed, the published revision, the publication dates, and the publication type:

```
<bookmeta>
  <publisherinformation>
      <organization>NY Publishing</organization>
      <printlocation>United States of America</printlocation>
      <published>
        <publishtype value="general"/>
        <revisionid>1</revisionid>
        <started><month>01</month><year>2020</year></started>
        <completed><month>02</month><year>2023</year></completed>
      </published>
  </publisherinformation>
</bookmeta>
```

4.2.1.2.26 <publishtype>

The `<publishtype>` element indicates whether there are any restrictions on the availability of the book.

Usage information

The `@value` attribute is required to specify the type of availability, for example, beta release, limited availability, or general availability.

Specialization hierarchy

The `<publishtype>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

For this element, the `@value` attribute specifies any restrictions on the availability of the publication.

Comment by tammy
This seems redundant since it's also captured in the Usage information section.

Example

The following code sample shows how the `<publishtype>` element can be used to specify any restrictions on the availability of the book. In this example, there are no restrictions.

```
<bookmeta>
  <publisherinformation>
      <organization>NY Publishing</organization>
      <printlocation>United States of America</printlocation>
      <published>
        <publishtype value="general"/>
        <revisionid>1</revisionid>
        <completed><month>02</month><year>2023</year></completed>
      </published>
  </publisherinformation>
</bookmeta>
```

4.2.1.2.27 <reviewed>

The `<reviewed>` element contains detailed information about a book review, such as the revision that was reviewed, who reviewed the book, and when the review occurred.

Specialization hierarchy

The `<reviewed>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following code sample shows how the `<reviewed>` element can be used to show which revision of the book was reviewed and when it was reviewed:

```
<bookmeta>
  <bookchangehistory>
    <reviewed>
      <revisionid>2</revisionid>
      <started><year>2019</year><month>10</month></started>
      <completed><year>2020</year><month>01</month></completed>
    </reviewed>
    <approved>
      <organization>OASIS</organization>
      <completed><year>2020</year><month>05</month></completed>
    </approved>
  </bookchangehistory>
</bookmeta>
```

4.2.1.2.28 <revisionid>

The `<revisionid>` element specifies the revision of the book.

Processing expectations

A processor determines how or whether the revision level is displayed. Common methods include using a dash, for example "-01", or a period, such as ".01".

Comment by tammy
I don't understand these processing expectations.

Specialization hierarchy

The `<revisionid>` element is specialized from `<ph>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how the `<revisionid>` element can be used to specify the book revision that was published and the revisions that were edited:

```
<bookmeta>
  <publisherinformation>
    <organization>NY Publishing</organization>
    <printlocation>United States of America</printlocation>
    <published>
      <revisionid>2</revisionid>
      <completed><month>02</month><year>2023</year></completed>
    </published>
  </publisherinformation>
  <bookchangehistory>
    <edited>
      <revisionid>1</revisionid>
      <person>Joe T. Editor</person>
      <completed><year>2020</year><month>03</month><day>15</day></completed>
    </edited>
    <edited>
      <revisionid>2</revisionid>
      <person>Joe T. Editor</person>
      <completed><year>2023</year><month>06</month><day>30</day></completed>
    </edited>
  </bookchangehistory>
</bookmeta>
```

4.2.1.2.29 <started>

The `<started>` element indicates when a book event began.

Specialization hierarchy

The `<started>` element is specialized from `<ph>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how the `<started>` element can be used to specify when publishing and editing began:

```
<bookmeta>
  <publisherinformation>
      <organization>NY Publishing</organization>
      <printlocation>United States of America</printlocation>
      <published>
        <publishtype value="general"/>
        <revisionid>2</revisionid>
        <started>
          <month>11</month><year>2022</year>
        </started>
        <completed><month>02</month><year>2023</year></completed>
      </published>
  </publisherinformation>
  <bookchangehistory>
    <edited>
      <organization>XYZ Editing</organization>
      <started>
        <month>08</month><year>2022</year>
      </started>
      <completed><month>10</month><year>2022</year></completed>
    </edited>
  </bookchangehistory>
</bookmeta>
```

4.2.1.2.30 <summary>

The `<summary>` element contains a brief summary related to a book event or to the copyrights of the book.

Specialization hierarchy

The `<summary>` element is specialized from `<ph>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how the `<summary>` element can be used to provide a brief summary of the edits that were completed on two different revisions of the book:

```
<bookmeta>
  <bookchangehistory>
    <edited>
      <person>Joe T. Editor</person>
      <revisionid>1</revisionid>
      <completed><year>2020</year><month>03</month><day>15</day></completed>
      <summary>Added several new topics</summary>
    </edited>
    <edited>
      <person>Joe T. Editor</person>
      <revisionid>2</revisionid>
      <completed><year>2020</year><month>10</month><day>13</day></completed>
      <summary>Fixed a few typos</summary>
    </edited>
  </bookchangehistory>
</bookmeta>
```

4.2.1.2.31 <tested>

The `<tested>` element contains detailed information about book testing, such as the revision that was tested, who tested the book, and when testing occurred.

Specialization hierarchy

The `<tested>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following code sample shows how the `<tested>` element can be used to indicate who completed testing and when testing occurred:

```
<bookmeta>
  <bookchangehistory>
    <reviewed>
      <revisionid>2</revisionid>
      <started><year>2019</year><month>10</month></started>
      <completed><year>2020</year><month>01</month></completed>
    </reviewed>
    <tested>
      <organization>OASIS</organization>
      <completed><year>2020</year><month>04</month></completed>
    </tested>
    <approved>
      <organization>OASIS</organization>
      <completed><year>2020</year><month>05</month></completed>
    </approved>
  </bookchangehistory>
</bookmeta>
```

4.2.1.2.32 <volume>

The `<volume>` element contains the volume number of the book.

Specialization hierarchy

The `<volume>` element is specialized from `<data>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: data-element attributes (179), link-relationship attributes (179), and universal attributes (173).

Example

The following example shows how the `<volume>` element can be used to indicate the volume number of the book:

```
<bookmeta>
  <bookid>
    <bookpartno>99F1234</bookpartno>
    <edition>Second</edition>
    <isbn>123-0-456-12345-1</isbn>
    <booknumber>SC21-1234-00</booknumber>
  <volume>2</volume>
  </bookid>
</bookmeta>
```

4.2.1.2.33 <year>

The `<year>` element denotes a year.

Specialization hierarchy

The `<year>` element is specialized from `<ph>`. It is defined in the bookmap module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0   ).

Example

See the example in bookmeta (48).

4.2.2 Concept elements

Concept elements provide the fundamental structure for concept topics. Concept topics are useful for introducing the background or overview information for task or reference topics.

4.2.2.1 <concept>

The `<concept>` element is the top-level element for a topic that answers the question “what is?”

Usage information

Concepts provide background information that users must know before they can successfully work with a product or interface. Often, a concept is an extended definition of a major abstraction such as a process or function. It might also contain an example, image, or diagram.

Specialization hierarchy

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

Attributes

The following attributes are available on this element: architectural attributes (178) and universal attributes (173).

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

Example

The following code sample shows a concept topic:

```
<concept id="concept">
  <title>DITA concept topic</title>
  <shortdesc>The concept topic answers the question <q>what is?</q></shortdesc>
    <conbody>
      <p>Concept topics provide background information that users must know
         before they can successfully work with a product or interface. Often,
         a concept is an extended definition of a major abstraction such as a
         process or function. It might also have an example or a graphic.</p>
    </conbody>
</concept>
```

4.2.2.2 <conbody>

The `<conbody>` element contains the main content of a concept topic.

Usage information

The `<conbody>` element allows paragraphs, lists, and other elements as well as sections and examples. However, `<conbody>` element has a restriction that a `<section>` or an `<example>` can be followed only by other sections, examples, or `<conbodydiv>` elements that group sections and examples.

Specialization hierarchy

The `<conbody>` element is specialized from `<body>`. It is defined in the concept module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

See `<concept>` (65).

4.2.2.3 <conbodydiv>

The `<conbodydiv>` element provides an container for content that might be grouped within a concept topic.

Usage information

There are no additional semantics attached to the `<conbodydiv>` element. It is purely a grouping element that is provided to help organize content for reuse.

The content model of the `<conbodydiv>` element only permits `<section>` and `<example>`.

Specialization hierarchy

The `<conbodydiv>` element is specialized from `<bodydiv>`. It is defined in the concept module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

The following code sample shows how a `<conbodydiv>`element can be used to group content for reuse:

```
<conbody>
  <conbodydiv id="concept-purpose-content-model">
    <section id="purpose">
      <title>Purpose</title>
        <p>Concept topics serve a variety of purposes:</p>
        <!-- ... -->
    </section>
    <section id="content-model">
      <title>Content model</title>
        <p>The body of a concept topic can contain the following document
          structures:</p>
        <!-- ... -->
    </section>
  </conbodydiv>
</conbody>
```

4.2.3 Glossary entry elements

The glossary entry specialization contains markup that supports the development and delivery of glossaries. It also can be used in conjunction with the abbreviated-form domain to provide a solution for rendering different forms of a term on first and later usage in a publication.

4.2.3.1 <glossAcronym>

The `<glossAcronym>` element defines an acronym for the term that is specified in the `<glossterm>` element.

Usage information

This element can be used with the `<abbreviated-form>` element to display an expanded version of an acronym the first time that acronym appears in a set of text. See `<abbreviated-form>` (109) for information on how the two elements interact.

Specialization hierarchy

The `<glossAcronym>` element is specialized from `<title>`. It is defined in the glossary entry module.

Attributes

The following attributes are available on this element: ID and conref attributes (174), localization attributes (174), `@base` (0), `@class` (0), `@outputclass` (0), and `@rev` (0).

Example

The following code sample shows how the `<glossAcronym>` element can be used:

```
<glossentry id="united-nations">
    <glossterm>United Nations</glossterm>
    <glossdef>The United Nations, referred to informally as the UN, is an
    intergovernmental organization whose stated purposes are to maintain
    international peace and security, develop friendly relations among
    nations, achieve international cooperation, and serve as a center for
    harmonizing the actions of nations.</glossdef>
  <glossBody>
    <glossSurfaceForm>United Nations (UN)</glossSurfaceForm>
    <glossAlt>
      <glossAcronym>UN</glossAcronym>
    </glossAlt>
  </glossBody>
</glossentry>
```

4.2.3.2 <glossAlt>

The `<glossAlt>` element contains information about a variant for the term that is specified in the `<glossterm>` element. A variant might include an acronym or a synonym.

Usage information

The variant should have the same meaning as the term in the `<glossterm>` element; the variant is simply another way to refer to the same term. There might be many ways to refer to a term; each variant is placed in its own `<glossAlt>` element. The `<glossUsage>` element can be used within `<glossAlt>` to indicate when use of the alternate term is appropriate.

Specialization hierarchy

The `<glossAlt>` element is specialized from `<section>`. It is defined in the glossary entry module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

The following code sample shows how a glossary entry topic might provide alternative forms of the term.

```
<glossentry id="usa">
  <glossterm>United States of America</glossterm>
  <glossdef>A federal republic in the northern Western Hemisphere comprising 48 contiguous
     states, the District of Columbia, Alaska in North America, and Hawaii in the North
     Pacific, and in some contexts considered along with its five inhabited island
     territories (Puerto Rico, U.S. Virgin Islands, Guam, North Mariana Islands, American
     Samoa).</glossdef>
  <glossBody>
    <glossAlt>
      <glossAcronym>US</glossSynonym>
      <glossUsage>Used as an adjective.</glossUsage>
    </glossAlt>
    <glossAlt>
      <glossAcronym>USA</glossSynonym>
      <glossUsage>Used as a noun.</glossUsage>
    </glossAlt>
    <glossAlt>
      <glossSynonym>America</glossSynonym>
      <glossUsage>Do not use; the American continents include other countries.</glossUsage>
    </glossAlt>
  </glossBody>
</glossentry>
```

4.2.3.3 <glossBody>

The `<glossBody>` element contains information about the term that is specified in the `<glossterm>` element, such as a acronym, synonyms, or usage notes.

Specialization hierarchy

The `<glossBody>` element is specialized from `<conbody>`; it is defined in the glossary entry module. The `<conbody>` element is specialized from `<body>`; it is defined in the concept module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

The following code sample shows how the `<glossBody>` element contains a synonym for the term:

```
<glossentry id="sport-drink">
    <glossterm>Sport drink</glossterm>
    <glossdef>A soft drink designed or marketed for consumption in
    conjunction with sporting activity or strenuous exercise, and which
    typically contains electrolytes such as sodium, potassium, and
    chloride, and a high percentage of sugar to restore energy.</glossdef>
  <glossBody>
    <glossAlt>
      <glossSynonym>energy drink</glossSynonym>
    </glossAlt>
  </glossBody>
</glossentry>
```

4.2.3.4 <glossdef>

The `<glossdef>` element defines the meaning of the term that is specified in the `<glossterm>` element.

Usage information

If a term has multiple meanings, create a separate `<glossentry>` topic for each.

Specialization hierarchy

The `<glossdef>` element is specialized from `<abstract>`. It is defined in the glossary entry module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

The following code sample shows the `<glossdef>` element can be used to define the meaning of the term "raster pattern":

```
<glossentry id="rasterpattern">
  <glossterm>raster pattern</glossterm>
  <glossdef>A series of picture elements (pels) arranged in scan lines to form
            an image.</glossdef>
</glossentry>
```

4.2.3.5 <glossentry>

The `<glossentry>` element is the top-level element for a topic that defines a glossary term.

Rendering expectations

Because the glossary entry specialization is designed for multiple purposes, it contains elements that typically are not intended to be rendered when a glossary is generated. In addition, when a collection of glossary entry topics is rendered as authoring guidance, generated text might be required for ease of reading. Specialized style sheets and processing are needed to ensure useful output.

Processing expectations

Processing expectations for glossary entry topics are highly implementation-specific and will depend on the output format.

For HTML-based output formats, one possible implementation of glossary entry topics is to generate hyperlinks for `<term>` element that are associated by key reference with the glossary entry topic. The term definition might be displayed when someone hovers over or clicks on the hyperlink.

For PDF output, one possible implementation of glossary entry topics is to render acronyms or expanded acronyms for `<abbreviated-form>` elements that are associated by key reference with the glossary entry topics. The surface form is rendered on first usage, and the acronym is rendered on second or later usage.

Specialization hierarchy

The `<glossentry>` element is specialized from `<concept>`; it is defined in the glossary entry module. The `<concept>` element is specialized from `<topic>`; it is defined in the concept module.

Attributes

The following attributes are available on this element: architectural attributes (178) and universal attributes (173).

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

Example

The following code samples shows how a glossary entry topic provides information about a term that aids in terminology management:

```
<glossentry id="usbfd">
  <glossterm>USB flash drive</glossterm>
  <glossdef>A small portable drive.</glossdef>
  <glossBody>
    <glossUsage>Do not use this term in upper case (for example,  in "USB Flash Drive")
      because that suggests a trademark.</glossUsage>
    <glossAlt>
      <glossAcronym>UFD</glossAcronym>
    </glossAlt>
    <glossAlt id="memoryStick">
      <glossSynonym>memory stick</glossSynonym>
      <glossUsage>This is a colloquial term.</glossUsage>
    </glossAlt>
  </glossBody>
</glossentry>
```

4.2.3.6 <glossSurfaceForm>

The `<glossSurfaceForm>` element specifies how the term that is specified by the `<glossterm>` element should appear in the text. The surface form is suitable to introduce the term in new contexts or as the first occurrence.

Usage information

The `<glossSurfaceForm>` element is most often used for terms that also specify the `<glossAcronym>` element. In that context, the `<glossSurfaceForm>` element contains the term in a manner that introduces both the term and the acronym, so that later references to the term can be replaced with the acronym alone.

See the `<abbreviated-form>` (109) element for a full description of how the surface form is used together with acronyms.

Specialization hierarchy

The `<glossSurfaceForm>` element is specialized from `<p>`. It is defined in the glossary entry module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

The following glossary entry topic defines the term "Anti-lock Braking System". Within the topic, the `<glossSurfaceForm>` element provides a version of the term that combines both the primary term and the acronym. The content of the `<glossSurfaceForm>` might be rendered in introductory contexts when the glossary entry topic is referenced from an `<abbreviated-form>` element.

```
<glossentry id="abs">
  <glossterm>Anti-lock Braking System</glossterm>
  <glossBody>
    <glossSurfaceForm>Anti-lock Braking System (ABS)</glossSurfaceForm>
    <glossAlt>
      <glossAcronym>ABS</glossAcronym>
    </glossAlt>
  </glossBody>
</glossentry>
```

4.2.3.7 <glossSymbol>

The `<glossSymbol>` element identifies a standard image that is associated with the subject of the term that is specified by `<glossterm>` element.

Specialization hierarchy

The `<glossSymbol>` element is specialized from `<image>`. It is defined in the glossary entry module.

Attributes

The following attributes are available on this element: universal attributes (173), `@format` (0), `@href` (0), `@keyref` (0), `@scope` (0), and the attributes defined below.

@align
Controls the horizontal alignment of an image when `@placement` is specified as "break". Common values include "left", "right", and "center".

@height
Specifies the vertical dimension for the resulting display. The value of this attribute is a real number expressed in decimal notation, optionally followed by a unit of measure. The following units of measurement are supported: cm, em, in, mm, pc, pt, and px (centimeters, ems, inches, millimeters, picas, points, and pixels, respectively). The default unit is px (pixels). Possible values include:"5", "5in", and "10.5cm".

@placement
Indicates whether an image is displayed inline or on a separate line. The default value is inline. Allowable values are "inline", "break", and "-dita-use-conref-target" (190).

@scale
Specifies a percentage as an unsigned integer by which to scale the image in the absence of any specified image height or width; a value of 100 implies that the image should be presented at its intrinsic size. If a value has been specified for the `@height` or `@width` attribute (or both), the `@scale` attribute is ignored.

@scalefit

Specifies whether an image is scaled up or down to fit within available space. The allowable values are "yes", "no", and "-dita-use-conref-target" (190). If `@height`, `@width`, or `@scale` is specified, those attributes determine the graphic size, and the `@scalefit` attribute is ignored. If none of those attributes are specified and `scalefit="yes"`, then the image is scaled by the same factor in both dimensions, so that the graphic will just fit within the available height or width, whichever is more constraining.

The available width would be that of the prevailing column or table cell, that is, the width a paragraph of text would have if the graphic were a paragraph instead od text. The available height is implementation dependent, but if feasible, it is suggested to be the page or table cell height or some other reasonable value.

@width
Specifies the horizontal dimension for the resulting display. The value of this attribute is a real number expressed in decimal notation, optionally followed by a unit of measure. The following units of measurement are supported: cm, em, in, mm, pc, pt, and px (centimeters, ems, inches, millimeters, picas, points, and pixels, respectively). The default unit is px (pixels). Possible values include:"5", "5in", and "10.5cm".

Example

The following code sample shows how the `<glossSymbol>` can be used to associate a regional classification icon with the Atlantic puffin:

```
<glossentry id="atlanticpuffin">
  <glossterm>Atlantic Puffin</glossterm>
  <glossdef>A sea bird that lives in the Atlantic
    <image href="puffinicon.jpg">
      <alt>Image of an atlantic puffin</alt>
    </image>
  </glossdef>
  <glossBody>
    <glossSymbol href="atlantic.jpg" scope="local">
      <alt>Icon denoting the Atlantic region</alt>
    </glossSymbol>
  </glossBody>
</glossentry>
```

4.2.3.8 <glossSynonym>

The `<glossSynonym>` element provides a term that is a synonym of the term that is specified by the `<glossterm>` element.

Usage information

The `<glossSynonym>` element should not be used to specify an acronym; use the `<glossAcronym>` element for that purpose.

Comment by Kristen James Eberlein on 24 March 2024

Eliot Kimber made the following comment in the (1st) Content Fusion review:

"I think this element is the answer to Dawn's question about `<glossAlternateFor>`not applying to the entire glossary definition. This element serves that role and, because it allows both `<term>` and `<keyword>`, can be a link to the glossary entry of the synonym term.

Do we want to provide an example of this?

Specialization hierarchy

The `<glossSynonym>` element is specialized from `<title>`. It is defined in the glossary entry module.

Attributes

The following attributes are available on this element: ID and conref attributes (174), localization attributes (174), `@base` (0), `@class` (0), `@outputclass` (0), and `@rev` (0).

Example

The following code sample shows how the `<glossSynonym>` element can be used to identify a synonym for the word automobile:

```
<glossentry id="automobile">
    <glossterm>Automobile</glossterm>
    <glossdef>A road vehicle, typically with four wheels, powered by an
    internal combustion engine or an electric motor.</glossdef>
  <glossBody>
    <glossAlt>
      <glossSynonym>car</glossSynonym>
    </glossAlt>
  </glossBody>
</glossentry>
```

4.2.3.9 <glossterm>

The `<glossterm>` element specifies the term that is defined by the glossary entry topic.

Specialization hierarchy

The `<glossterm>` element is specialized from `<title>`. It is defined in the glossary entry module.

Attributes

The following attributes are available on this element: ID and conref attributes (174), localization attributes (174), `@base` (0), `@class` (0), `@outputclass` (0), and `@rev` (0).

Example

The following code sample shows how the `<glossterm>` element specifies the term that is defined in the `<glossentry>` topic:

```
<glossentry id="css">
    <glossterm>cascading style sheets</glossterm>
    <glossdef>Cascading Style Sheets is a style sheet language that is used
    for rendering the presentation of a document written in a markup
    language such as HTML or XML</glossdef>
  <glossBody>
    <glossSurfaceForm>cascading style sheets (CSS)</glossSurfaceForm>
    <glossAlt>
      <glossAcronym>CSS</glossAcronym>
    </glossAlt>
    <glossAlt>
      <glossSynonym>web style sheets</glossSynonym>
    </glossAlt>
  </glossBody>
</glossentry>
```

4.2.3.10 <glossUsage>

The `<glossUsage>` element provides information about how to use the term that is specified in the `<glossterm>` element. It also can be used to provide usage information for acronyms or synonyms.

Specialization hierarchy

The `<glossUsage>` element is specialized from `<note>`. It is defined in the glossary entry module.

Attributes

The following attributes are available on this element: universal attributes (173) and the attributes defined below.

@othertype
Specifies an alternate note type. This value is used as the user-provided note label when the `@type` attribute value is set to "other".

@type
Specifies the type of a note. This differs from the `@type` attribute on many other DITA elements. The following are the allowable values:

- "attention"
- "caution"
- "danger"
- "important"
- "note"
- "notice"
- "other"
- "remember"
- "restriction"
- "tip"
- "trouble"
- "warning"
- "-dita-use-conref-target"

Example

The following code sample shows how the `<glossUsage>` element is used to provide additional information about possible variants for the term "soft drink:"

```
<glossentry id="softdrink">
  <glossterm>soft drink</glossterm>
  <glossdef>A nonalcoholic drink, especially one that is carbonated.</glossdef>
  <glossBody>
    <glossAlt>
      <glossSynonym>pop</glossSynonym>
      <glossUsage>Used primarily in the North and Midwest</glossUsage>
    </glossAlt>
    <glossAlt>
      <glossSynonym>soda</glossSynonym>
      <glossUsage>Used primarily in the West and Northeast</glossUsage>
    </glossAlt>
    <glossAlt>
      <glossSynonym>Coke</glossSynonym>
      <glossUsage>Used primarily in the South</glossUsage>
    </glossAlt>
  </glossBody>
</glossentry>
```

4.2.4 <glossgroup>

A glossary group topic organizes related glossary entry topics within a single topic document.

Usage information

Glossary groups are primarily a convenience for authoring when it is useful to author multiple glossary entry topics in a single document.

Specialization hierarchy

The `<glossgroup>` element is specialized from `<concept>`; it is defined in the glossary group module. The `<concept>` element is specialized from `<topic>`; it is defined in the concept module.

Attributes

The following attributes are available on this element: architectural attributes (178) and universal attributes (173).

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

Example

```
<glossgroup id="things" xml:lang="en">
  <title>Some terms</title>
  <glossentry id="bicycle">
    <glossterm>bicycle</glossterm>
    <glossdef>Human powered mode of transport
       with two wheels</glossdef>
  </glossentry>
  <glossentry id="fruitbat">
    <glossterm>Fruit bat</glossterm>
    <glossdef>A bat which likes fruit</glossdef>
  </glossentry>
</glossgroup>
```

4.2.5 Reference elements

Reference elements provide the fundamental structure for reference topics. Reference topics include specialized sections for programming language syntax and property lists, as well as standard elements such as sections, tables, and examples.

4.2.5.1 <propdesc>

The `<propdesc>` element contains content that describes the property type and its values.

Specialization hierarchy

The `<propdesc>` element is specialized from `<stentry>`. It is defined in the reference module.

Attributes

The following attributes are available on this element: table accessibility attributes (179), universal attributes (173), and the attribute defined below.

@rowspan
Specifies the number of rows that a cell is to span inside a simple table.

Example

See `<properties>` (76).

4.2.5.2 <propdeschd>

The `<propdeschd>` element provides a label for the description column in a properties table.

Specialization hierarchy

The `<propdeschd>` element is specialized from `<stentry>`. It is defined in the reference module.

Attributes

The following attributes are available on this element: table accessibility attributes (179) and universal attributes (173)

Example

See `<properties>` (76).

4.2.5.3 <properties>

A properties table describes the properties of a thing, such as an object, part, or category. Each property can include the type, value, and a description.

Usage information

A properties table typically is represented as a simple table with a maximum of three columns. The first column is for the property type, the second column can contain a value or values for the property, and the third column can contain a description.

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

Rendering expectations

If a properties table does not contain a header row, processors typically auto-generate labels for the columns in the properties table. The text for the labels is specified in stylesheets.

Specialization hierarchy

The `<properties>` element is specialized from `<simpletable>`. It is defined in the reference module.

Attributes

The following attributes are available on this element: universal attributes (173), display attributes (179), and simpletable attributes (179).

Examples

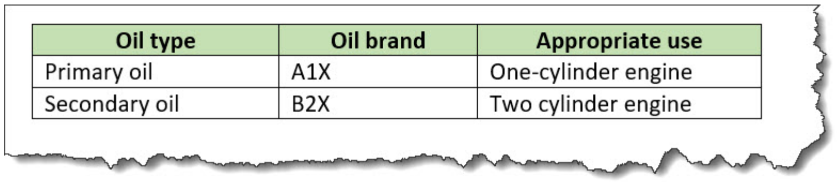
This section contains examples of how the `<properties>` element can be used.

Figure 7: Simple properties table

The following code sample shows a `<properties>` element that describes information about motor oil types:

```
<properties>
  <prophead>
    <proptypehd>Oil type</proptypehd>
    <propvaluehd>Oil brand</propvaluehd>
    <propdeschd>Appropriate use</propdeschd>
  </prophead>
  <property>
    <proptype>Primary oil</proptype>
    <propvalue>A1X</propvalue>
    <propdesc>One-cylinder engines</propdesc>
  </property>
  <property>
    <proptype>Secondary oil</proptype>
    <propvalue>B2Z</propvalue>
    <propdesc>Two-cylinder engines</propdesc>
  </property>
</properties>
```

The properties table might be rendered as follows:



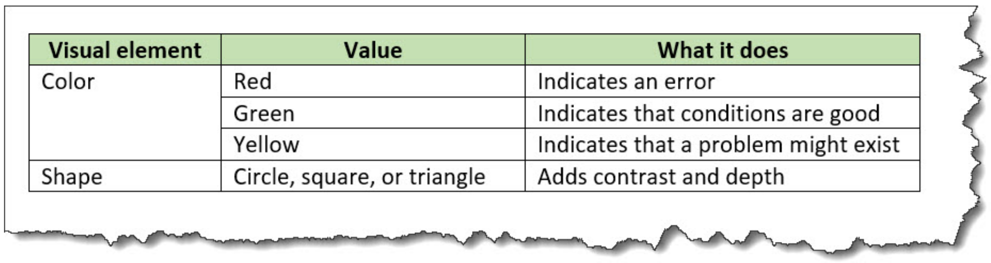
Oil type	Oil brand	Appropriate use
Primary oil	A1X	One-cylinder engine
Secondary oil	B2X	Two cylinder engine

Figure 8: Properties table with spanned cells

The following code sample shows a properties table with spanned cells:

```
<properties>
  <prophead>
    <proptypehd>Visual element</proptypehd>
    <propvaluehd>Value</propvaluehd>
    <propdeschd>What it does</propdeschd>
  </prophead>
  <property>
    <proptype rowspan="3">Color</proptype>
    <propvalue>Red</propvalue>
    <propdesc>Indicates an error</propdesc>
  </property>
  <property>
    <propvalue>Green</propvalue>
    <propdesc>Indicates that conditions are good</propdesc>
  </property>
  <property>
    <propvalue>Yellow</propvalue>
    <propdesc>Indicates that a problem might exist</propdesc>
  </property>
  <property>
    <proptype>Shape</proptype>
    <propvalue>Circle, square, or triangle</propvalue>
    <propdesc>Use to add contrast and depth</propdesc>
  </property>
</properties>
```

The properties table might be rendered as follows:



Visual element	Value	What it does
Color	Red	Indicates an error
	Green	Indicates that conditions are good
	Yellow	Indicates that a problem might exist
Shape	Circle, square, or triangle	Adds contrast and depth

4.2.5.4 <property>

The `<property>` element represents a single property in a properties table.

Specialization hierarchy

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

Attributes

The following attributes are available on this element: universal attributes (173).

Example

See `<properties>` (76).

4.2.5.5 <prophead>

The `<prophead>` element contains elements that provide labels for the columns in a properties table.

Rendering expectations

If a properties table does not contain a header row, processors typically auto-generate labels for the columns in the properties table. The text for the labels is specified in stylesheets.

Specialization hierarchy

The `<prophead>` element is specialized from `<sthead>`. It is defined in the reference module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

See `<properties>` (76).

4.2.5.6 <proptype>

The `<proptype>` element contains content that describes the type of the property.

Specialization hierarchy

The `<proptype>` element is specialized from `<stentry>`. It is defined in the reference module.

Attributes

The following attributes are available on this element: table accessibility attributes (179), universal attributes (173), and the attribute defined below.

@rowspan
Specifies the number of rows that a cell is to span inside a simple table.

Example

See `<properties>` (76).

4.2.5.7 <proptypehd>

The `<proptypehd>` element provides a label for the type column in a properties table.

Specialization hierarchy

The `<proptypehd>` element is specialized from `<stentry>`. It is defined in the reference module.

Attributes

The following attributes are available on this element: table accessibility attributes (179) and universal attributes (173)

Example

See `<properties>` (76).

4.2.5.8 <propvalue>

The `<propvalue>` element contains content that indicates a value for the property type.

Specialization hierarchy

The `<propvalue>` element is specialized from `<stentry>`. It is defined in the reference module.

Attributes

The following attributes are available on this element: table accessibility attributes (179), universal attributes (173), and the attribute defined below.

@rowspan
Specifies the number of rows that a cell is to span inside a simple table.

Example

See `<properties>` (76).

4.2.5.9 <propvaluehd>

The `<propvaluehd>` element provides a label for the value column in a properties table.

Specialization hierarchy

The `<propvaluehd>` element is specialized from `<stentry>`. It is defined in the reference module.

Attributes

The following attributes are available on this element: table accessibility attributes (179) and universal attributes (173)

Example

See `<properties>` (76).

4.2.5.10 <refbody>

The `<refbody>` element contains the main content of a reference topic.

Specialization hierarchy

The `<refbody>` element is specialized from `<body>`. It is defined in the reference module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

See `<reference>` (81).

4.2.5.11 <refbodydiv>

The `<refbodydiv>` element provides a container for contiguous content in a reference topic. There is no additional semantic meaning.

Usage information

The `<refbodydiv>` element is useful primarily for reuse and as a specialization base.

Specialization hierarchy

The `<refbodydiv>` element is specialized from `<bodydiv>`. It is defined in the reference module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

The following code sample shows how a `<refbodydiv>` element can be used to group content for reuse

```
<reference id="sample-refbodydiv" xml:lang="en">
 <title>Sample for refbody</title>
 <shortdesc>This shows how refbodydiv might be used.</shortdesc>
 <refbody>
  <refbodydiv id="widget1">
   <section>This is one part of the sample</section>
   <refsyn>Syntax for this part</refsyn>
  </refbodydiv>
  <refbodydiv id="widget2">
    <section>This is another part of the sample</section>
    <refsyn>Syntax for this part</refsyn>
  </refbodydiv>
  </refbody>
</reference>
```

Comment by Kristen J Eberlein on 26 October 2022

Can someone come up with a more realistic example?

4.2.5.12 <reference>

The `<reference>` element is the top-level element for a reference topic. A reference topic can include specialized sections for programming syntax and property tables, as well as standard sections, tables, and examples.

Usage information

For information about the purpose and content model of a reference topic, see Reference (16).

Specialization hierarchy

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

Attributes

The following attributes are available on this element: architectural attributes (178) and universal attributes (173).

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

Example

Comment by Kristen J Eberlein on 08 November 2022

This is a pretty lousy example ...

The following code sample shows how a reference topic can be used:

```
<reference id="requiredTools">
  <title>Tools required to maintain Acme machinery</title>
  <refbody>
    <section>
      <title>Small tools</title>
      <ul>
        <li>Hard hat</li>
        <li>Hammer</li>
        <li>Nail</li>
        <li>Metal polish</li>
        <!-- .... -->
      </ul>
    </section>
    <section>
      <title>Expensive tools</title>
      <!-- .... -->
    </section>
  </refbody>
</reference>
```

4.2.5.13 <refsyn>

The `<refsyn>` element contains content that describes the syntax of a command.

Specialization hierarchy

The `<refsyn>` element is specialized from `<section>`. It is defined in the reference module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

The following code sample shows how the `<refsyn>` element can be used to document the syntax for the Windows `mkdir` command:

```
<refsyn>
  <title>Syntax</title>
  <codeblock>mkdir <varname>drive</varname> <varname>directory</varname></codeblock>
  <parml>
    <plentry>
      <pt><varname>drive</varname></pt>
      <pd>Specifies the drive on which the new directory is created. This is an optional
          parameter.</pd>
    </plentry>
    <plentry>
      <pt><varname>path</varname></pt>
      <pd>Specifies the fully-qualified name of the new directory. This is a required
          parameter.</pd>
    </plentry>
  </parml>
</refsyn>
```

4.2.6 Task elements

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.

4.2.6.1 <chdesc>

The `<chdesc>` element provides the content of the second cell in a choice table row. This content describes the option that people can take to complete the step, and it explains the result of the choice, if it is not immediately obvious.

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: table accessibility attributes (179) and universal attributes (173)

Example

See `<choicetable>` (84).

4.2.6.2 <chdeschd>

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

Rendering expectations

The contents of the `<chdeschd>` element is typically rendered in a bold font.

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: table accessibility attributes (179) and universal attributes (173)

Example

See `<choicetable>` (84).

4.2.6.3 <chhead>

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

Rendering expectations

Labels provided by the `<chhead>` element override any default headings for the `<choicetable>` that might be 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 (173).

Example

See `<choicetable>` (84).

4.2.6.4 <choice>

A `<choice>` element describes a way to complete the current step.

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 (173).

Example

See `<choices>` (84)

4.2.6.5 <choices>

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

Usage information

The `<choices>` element provides information when there is more than one way to complete a step. It is a list.

Specialization hierarchy

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

Attributes

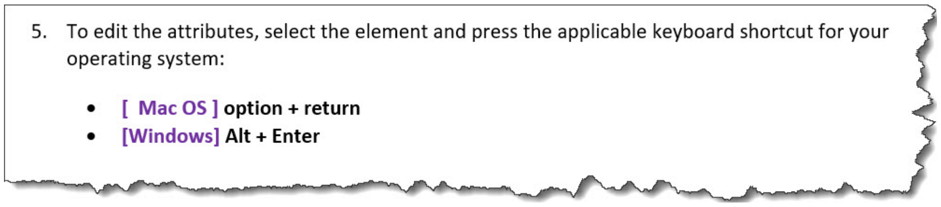
The following attributes are available on this element: universal attributes (173).

Example

The following code sample shows how the `<choices>` element can be used when different operating systems have different keyboard shortcuts. In this scenario, flagging is used to render labels for the different operating systems.

```
<step>
  <cmd>To edit the attributes, select the element and press the
       applicable keyboard shortcut for your operating system:</cmd>
  <choices>
    <choice platform="mac-os"><uicontrol>option + return</uicontrol></choice>
    <choice platform="windows"><uicontrol>Alt + Enter</uicontrol></choice>
  </choices>
  <stepresult>The <wintitle>Attributes</wintitle> view is displayed.</stepresult>
</step>
```

The rendered output might look like the following:



5. To edit the attributes, select the element and press the applicable keyboard shortcut for your operating system:
    - [Mac OS] **option + return**
    - [Windows] **Alt + Enter**

4.2.6.6 <choicetable>

A choice table provides information about a set of options for completing a step.

Usage information

A choice table provides information when there is more than one way to complete a step. It is a simple table with two columns. The first cell in a row labels the option, and the second cell in the row describes the option that a user can take to complete the step.

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

Rendering expectations

If a choice table does not contain a header row, processors typically auto-generate labels for the columns in the choice table. The text for the labels is specified in stylesheets.

Specialization hierarchy

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

Attributes

The following attributes are available on this element: universal attributes (173), display attributes (179), and simpletable attributes (179).

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

Examples

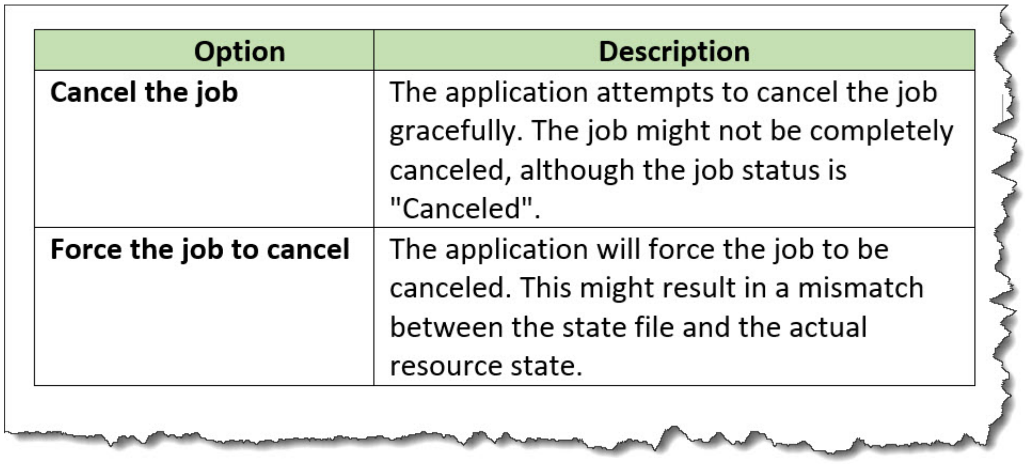
This section contains examples of how the `<choicetable>` element can be used.

Figure 9: Simple choice table

The following code sample contains a `<choicetable>` element that is used to explain the options that a user can take to cancel a job:

```
<step>
  <cmd>Select the option that you want:</cmd>
  <choicetable relcolwidth="1* 2*">
    <chrow>
      <choption>Cancel job</choption>
      <chdesc>The application attempts to cancel the job gracefully.
              The job might not be completely canceled, although the job
              status is "Canceled".</chdesc>
    </chrow>
    <chrow>
      <choption>Force the job to cancel</choption>
      <chdesc>The application will force the job to be canceled. This
              might result in a mismatch between the state file and the
              actual resource state.</chdesc>
    </chrow>
  </choicetable>
</step>
```

The choice table might be rendered in the following way. Note that the labels for the columns are contributed by the stylesheets that are used by the processor.



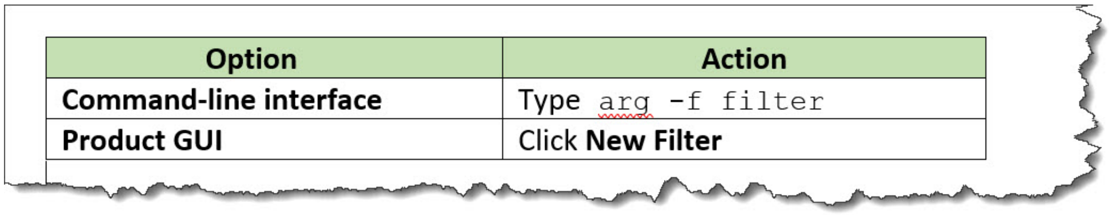
Option	Description
Cancel the job	The application attempts to cancel the job gracefully. The job might not be completely canceled, although the job status is "Canceled".
Force the job to cancel	The application will force the job to be canceled. This might result in a mismatch between the state file and the actual resource state.

Figure 10: Choice table with a header row

The following code sample contains a `<choicetable>` element that contains a header row. The choice table is used to provide users with instructions for creating a filter using either the command line or the graphical user interface. The header row is used to specify column labels of "Option" and "Action".

```
<step>
  <cmd>Create a new filter:</cmd>
  <choicetable>
    <chhead>
      <choptionhd>Option</choptionhd>
      <chdeschd>Action</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	Action
Command-line interface	Type `arg -f filter`
Product GUI	Click **New Filter**

4.2.6.7 <choption>

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.

Rendering expectations

Unless the `@keycol` attribute on the `<choicetable>` element is set to "0", the contents of the `<choiceoption>` element is typically rendered in a bold font.

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: table accessibility attributes (179) and universal attributes (173)

Example

See `<choicetable>` (84).

4.2.6.8 <choptionhd>

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

Rendering expectations

The contents of the `<chdeschd>` element is typically rendered in a bold font.

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: table accessibility attributes (179) and universal attributes (173)

Example

See `<choicetable>` (84).

4.2.6.9 <chrow>

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

Specialization hierarchy

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

Attributes

The following attributes are available on this element: universal attributes (173).

Example

See `<choicetable>` (84).

4.2.6.10 <cmd>

A command specifies the action that people take to complete a step.

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 (173) and `@keyref` (0).

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>
```

4.2.6.11 <context>

Contextual information is background information that helps people understand the purpose of the task and what they will gain by completing it.

Rendering expectations

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

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 (173).

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.

```
<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>
```

4.2.6.12 <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 (173).

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>You can use either the command line or the product GUI.
  </info>
</step>
```

4.2.6.13 <postreq>

Post-requisites are steps or tasks that people might need to perform after completing the current task.

Rendering expectations

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

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 (173).

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>
```

4.2.6.14 <prereq>

Prerequisites are things that people need to know or preliminary tasks that people need to perform before starting the current task.

Rendering expectations

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

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 (173).

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.

4.2.6.15 <result>

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

Rendering expectations

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

Specialization hierarchy

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

Attributes

The following attributes are available on this element: universal attributes (173).

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 <wintitle>File Created<wintle> window is displayed, and the SQLJ
            file is successfully created.
    </result>
  </taskbody>
</task>
```

4.2.6.16 <step>

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

Rendering expectations

When the `@importance` attribute is specified on the `<step>` 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>`,

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 (173).

For this element, the `@importance` attribute is limited to the values "optional", "required", or -dita-use-conref-target (190).

Example

The following code sample shows many of the elements that the `<step>` element can contain:

```
<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 Configuation 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>
```

4.2.6.17 <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 (173).

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>You can use either the command line or the product GUI.</info>
  <choices>
    <choice>From a command prompt, type <codeph>config -l parameter</codeph></choice>
    <choice>Click <uicontrol>New Configuration Parameters</uicontrol></choice>
  </choices>
  <stepresult>You receive a <systemoutput>'Configuration successful'</systemoutput> message.
  </stepresult>
</step>
```

4.2.6.18 <steps>

Steps are a series of actions that people perform in a specific order and manner.

Rendering expectations

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.

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

Specialization hierarchy

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

Attributes

The following attributes are available on this element: universal attributes (173).

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>
```

4.2.6.19 <steps-informal>

Informal steps are steps that 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.

Rendering expectations

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

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 (173).

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>
```

4.2.6.20 <stepsection>

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

Usage information

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.

Rendering expectations

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

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 (173).

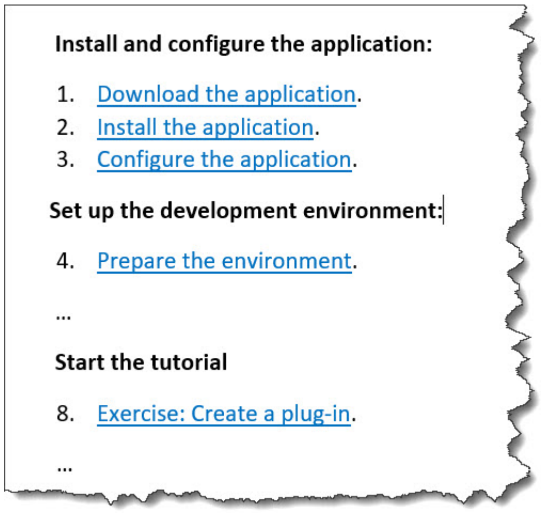
Example

The following code sample shows how `<stepsection>` element can be used to group steps in a high-level overview topic that links to other topics:

```
<steps>
<stepsection>Install and configure the application:</stepsection>
  <step>
    <cmd><xref keyref="download">Download the application</xref>.</cmd>
  </step>
  <step>
```

```
    <cmd><xref keyref="install">Install the application</xref>.</cmd>
  </step>
  <step>
    <cmd><xref keyref="configure">Configure the application</xref></cmd>
  </step>
  <stepsection>Set up the development environment:</stepsection>
  <step>
    <cmd><xref keyref="prep">Prepare the environment</xref>.</cmd>
  </step>
  <!-- ... -->
  <stepsection>Start the tutorial:</stepsection>
  <step>
    <cmd><xref keyref="create-plugin">Exercise: Create a plug-in</xref>.</cmd>
  </step>
  <!-- ... -->
</steps>
```

This topic might be rendered in the following way. Note that the numbering of the steps is not affected by the introduction of the `<stepsection>` elements.



Install and configure the application:

1. Download the application.
2. Install the application.
3. Configure the application.

Set up the development environment:

4. Prepare the environment.

...

Start the tutorial

8. Exercise: Create a plug-in.

...

4.2.6.21 <steptroubleshooting>

Step troubleshoooting is information that is intended to help people respond to the situation if a step does not complete as expected.

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 (173).

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>
```

4.2.6.22 <steps-unordered>

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

Rendering expectations

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

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 (173).

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>
```

4.2.6.23 <stepxmp>

A step example illustrates how a step is completed. The example might be text-based, an image, a code sample, a link to a video, or some other representation.

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 (173).

Example

The following code sample shows how the `<stepxmp>` can provide an example of how a user can perform a step:

```
<step>
  <cmd>Add an XML comment in the map that explains why you applied the
       filtering attribute.</cmd>
  <stepxmp><p>For example:<p>
    <codeblock>
      <!-- 18 Dec 2019 ML: The following topic is under review and should
                            not be published externally. [DH-1441]. -->
    </codeblock>
  </stepxmp>
</step>
```

4.2.6.24 <task>

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

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: architectural attributes (178) and universal attributes (173).

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>
  <!-- ... -->
</tas k>
```

4.2.6.25 <taskbody>

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

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 (173).

Examples

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

Figure 11: 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>
```

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 12: General task topic

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

```
<task id="completing-group-project">
  <title>Completing the final project</title>
  <shortdesc>This handout contains information about completing the final project
      for History 275, "Exploring your community history."</shortdesc>
  <taskbody>
    <context>The final project will account for 35% of your final grade.</context>
    <prereq>You must have an account on the college's collaboration platform.</prereq>
    <section>
      <title>Required reading</title>
      <ul>
        <li>Section 7.0 in the class course pack</li>
        <li><cite>Using Oral History in Community History Projects
            (Practices in Oral History)</cite></li>
      </ul>
    </section>
    <steps>
      <!-- ... -->
    </steps>
  </taskbody>
</task>
```

Note that there is more flexibility in the content model for `<taskbody>` in general task than there is in the strict task. In this example, `<context>` precedes `<prereq>`, and `<prereq>` is following by a section titled "Required reading".

Figure 13: General task topic used for reuse

The following code sample shows the content of a general task topic that is used to store `<prereq>` elements that are reused. While the implementation uses the strict task topic for their product documentation, using a general task topic for a reuse topic enables them to have multiple `<prereq>` elements in a single topic.

```
<task id="reuse-prereq">
  <title>Reuse topic: <xmlelement>prereq</xmlelement></title>
  <shortdesc>This topic stores <xmlelement>prereq</xmlelement> elements
             that are reused in the product documentation.</shortdesc>
  <taskbody>
    <!-- ... -->
    <prereq id="sp-10">Service Pack 10 must be installed.</prereq>
    <prereq id="admin-access">You must have administrator access in order
      to perform this procedure.</prereq>
    <!-- ... -->
  </taskbody>
</task>
```

4.2.6.26 <tasktroubleshooting>

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

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.

Rendering expectations

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

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 (173).

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>
```

4.2.6.27 <tutorialinfo>

Tutorial information is information that is useful when the task topic is rendered as a tutorial.

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 (173).

Example

The following code sample is of a task topic that is used both in a product manual and in a tutorial. Note that the `<context>` and `<tutorialinfo>` elements are intended to be rendered only when the tutorial is generated.

```
<task id="task-msg-x1z-gwb">
  <title>Taking pictures in low light without a flash</title>
  <shortdesc>Taking pictures in low light situations without a flash can be a challenge if you
      don’t know what you’re doing and can result in photos that are too dark, blurry, or
      grainy. Follow these suggestions to get excellent shots in low light situations without
      the need for your camera's flash.</shortdesc>
  <taskbody>
    <context deliveryTarget="tutorial">For example, suppose you are visiting the Louvre and
        want to capture memories of your visit. Most museums do not allow flash photography
        of their art masterpieces.  What settings will work best for that situation? To
        understand the best settings for such a situation before arriving, use this tutorial
        to experiment with the impact of your light-controlling settings on your camera.
    </context>
    <steps>
      <step>
```

```
        <cmd>Put your camera in manual mode.</cmd>
      </step>
      <step>
        <cmd>Increase your ISO setting to adjust how sensitive your camera's image sensor is
            to light.</cmd>
        <tutorialinfo deliveryTarget="tutorial">Take a picture at each of the following ISO
               settings: 100, 200, 400, and 800. Compare your results.</tutorialinfo>
      </step>
      <step>
        <cmd>Increase the aperture size, by reducing your f-stop, to adjust how much light is
            allowed in.</cmd>
        <tutorialinfo deliveryTarget="tutorial">Return your camera to an ISO setting of 100.
            Take a picture at each of the following f-stops: f/2.0, f/4, f/8, and f/16.
            Compare your results.</tutorialinfo>
      </step>
      <!-- ... --.
    </steps>
  </taskbody>
</task>
```

4.2.7 Troubleshooting elements

Troubleshooting elements provide the fundamental structure for troubleshooting topics. Troubleshooting topics describe problems and provide information about how to fix them.

Related concepts

Troubleshooting (20)

Troubleshooting topics are designed to document problems that people might encounter. They provide a topic structure that enables content authors to describe a condition, provide diagnostic information, discuss causes, and outline possible solutions.

4.2.7.1 <cause>

The `<cause>` element describes a potential source of the problem that is addressed by the troubleshooting topic.

Usage information

The `<cause>` element is a component of a potential solution. The cause might be omitted if it is implicit or if the remedy is not associated with a cause.

Specialization hierarchy

The `<cause>` element is specialized from `<section>`. It is defined in the troubleshooting module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

In the following code sample, the `<cause>` element contains information that explains the origins of the problem:

```
<troubleshooting id="simple-example">
  <title>System will not turn on</title>
  <troublebody>
    <condition>The system is plugged in and powered up, but the system will not start.
    </condition>
    <troubleSolution>
      <!-- . . . -->
      <cause>The problem is usually due to the power not being supplied to the system through
             the electrical outlet. Often, a circuit breaker has been tripped so that no
             power is available at the outlet.</p>
      </cause>
      <!-- . . . -->
    </troubleSolution>
    <!-- . . . -->
  </troublebody>
</troubleshooting>
```

4.2.7.2 <condition>

The `<condition>` element describes a state that the troubleshooting topic is intended to remedy. This information helps the user decide whether a troubleshooting topic might contain an applicable remedy for a problem.

Usage information

This section should add to or clarify information that is in the title or short description of the troubleshooting topic. If the title and short description adequately describes the condition, this element might be omitted.

Specialization hierarchy

The `<condition>` element is specialized from `<section>`. It is defined in the troubleshooting module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

In the following code sample, the `<condition>` element contains information that elaborates on the information that is provided by the title and short description:

```
<troubleshooting id="system-will-not-turn-on">
  <title>System will not turn on</title>
  <shortdesc>Everything looks right, but the system still does not start.</shortdesc>
  <troublebody>
    <condition>
      <title>Condition</title>
      <p>The system is plugged in and powered up, but the system does not start.</p>
    </condition>
    <troubleSolution>
      <!-- ... -->
    </troubleSolution>
  </troublebody>
</troubleshooting>
```

Alternately, the short description could be enhanced and the `<condition>` element eliminated:

```
<troubleshooting id="system-will-not-turn-on">
  <title>System will not turn on</title>
  <shortdesc>The system is plugged in and powered up, but the system does not start.
  </shortdesc>
  <troublebody>
    <troubleSolution>
      <!-- ... -->
    </troubleSolution>
  </troublebody>
</troubleshooting>
```

The markup pattern that implementations choose might depend on how they deliver troubleshooting information.

4.2.7.3 <diagnostics>

The `<diagnostics>` element contains information that helps readers determine the cause of a problem.

Usage information

Diagnostic information is useful when there is more than one potential cause associated with a symptom. The `<diagnostics-general>` element permits content that includes tables and flowcharts, while the `<diagnostics-steps>` element allows for the use of the `<steps>` element. Either or both elements can be present.

Specialization hierarchy

The `<diagnostics>` element is specialized from `<bodydiv>`. It is defined in the troubleshooting module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

See `<diagnostics-general>` (102) and `<diagnostics-steps>` (104).

4.2.7.4 <diagnostics-general>

The `<diagnostics-general>` element includes non-procedural information that can help determine the causes of a symptom. Results of the diagnoses might link to possible solutions.

Usage information

This element is useful for presenting non-procedural diagnostic information, for example, a diagnostic table or a flowchart. Non-procedural diagnostic information can be used when the symptoms can be observed and do not require people to take action.

Specialization hierarchy

The `<diagnostics-general>` element is specialized from `<section>`. It is defined in the troubleshooting module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

Comment by Kristen J Eberlein on 15 February 2023

We discussed the car noise example at the 14 February 2023 DITA TC meeting. The consensus was to add a second example, which Stan Doherty will develop.

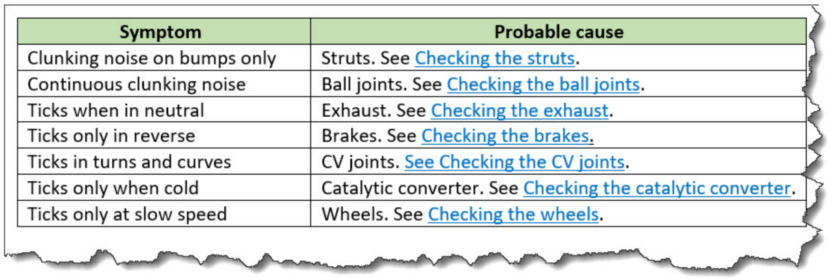
This section contains examples of how the `<diagnostics-general>` element can be used. Implementations might well have different business rules for how to document troubleshooting.

Figure 14: Example: Simple diagnosis

The following code sample shows how the `<diagnostics-general>` element can contain a table to help a reader determine the cause of a problem. The table then references the associated remedy.

```
<troubleshooting id="car-makes-funny-noises">
  <title>Car is making funny noises.</title>
  <shortdesc>You probably know how your vehicle sounds when it’s running
    properly. Listening to your car can help you troubleshoot problems. If
    you hear a strange sound, pay attention and react
    accordingly.</shortdesc>
  <troublebody>
    <diagnostics>
      <diagnostics-general>
        <simpletable frame="all" relcolwidth="1* 1*">
          <sthead>
            <stentry>Symptom</stentry>
            <stentry>Probable cause</stentry>
          </sthead>
          <strow>
            <stentry>Clunking noise on bumps only</stentry>
            <stentry>Struts. See <xref href="#./checkstruts"/>.</stentry>
          </strow>
          <strow>
            <stentry>Continuous clunking noise</stentry>
            <stentry>Ball joints. See <xref href="#./checkballjoints"/>.</stentry>
          </strow>
          <strow>
            <stentry>Ticks when in neutral</stentry>
            <stentry>Exhaust. See <xref href="#./checkexhaust"/>.</stentry>
          </strow>
          <strow>
            <stentry>Ticks only in reverse</stentry>
            <stentry>Brakes. See <xref href="#./checkbrakes"/>.</stentry>
          </strow>
          <strow>
            <stentry>Ticks in turns and curves</stentry>
            <stentry>CV joint. See <xref href="#./checkcvjoint"/>.</stentry>
          </strow>
          <strow>
            <stentry>Ticks only when cold</stentry>
            <stentry>Catalytic converter. See <xref href="#./checkcatalyticconverter"/>.
            </stentry>
          </strow>
          <strow>
            <stentry>Ticks only at slow speed</stentry>
            <stentry>Wheels. See <xref href="#./checkwheels"/>.</stentry>
          </strow>
        </simpletable>
      </diagnostics-general>
    </diagnostics>
    <!-- The rest of this topic contains <troublesolution> elements, each of which
         contains a remedy. The cross references in the above steps resolve to the
         <remedy> elements. -->
  </troublebody>
</troubleshooting>
```

The table in the `<diagnostics-general>` element might be rendered in the following way. The hyperlinks in the "Probable cause" column resolve to the `<remedy>` elements in the topic.



Symptom	Probable cause
Clunking noise on bumps only	Struts. See Checking the struts.
Continuous clunking noise	Ball joints. See Checking the ball joints.
Ticks when in neutral	Exhaust. See Checking the exhaust.
Ticks only in reverse	Brakes. See Checking the brakes.
Ticks in turns and curves	CV joints. See Checking the CV joints.
Ticks only when cold	Catalytic converter. See Checking the catalytic converter.
Ticks only at slow speed	Wheels. See Checking the wheels.

Figure 15: Example: Rigorous diagnosis

4.2.7.5 <diagnostics-steps>

The `<diagnostics-steps>` element includes step-by-step information that can help readers determine the causes of a symptom. Results of the diagnostic steps might link to potential solutions.

Usage information

This element is helpful for situations where the reader must perform a series of steps to determine the cause of the problem.

Specialization hierarchy

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

Attributes

The following attributes are available on this element: universal attributes (173).

Example

The following code sample shows how the `<diagnostics-steps>` element can provide step-by-step instructions to help someone determine the cause of a problem and the potential solution:

```
<troubleshooting id="my-network-isnt-working">
  <title>My network isn't working</title>
  <shortdesc>Users are unable to access network servers, the internet, or other
      networked devices, such as printers.</shortdesc>
  <troublebody>
    <diagnostics>
      <diagnostics-steps>
        <steps>
          <step>
            <cmd>Open the command prompt and type <userinput>ipconfig</userinput>.</cmd>
            <info>The Default Gateway (listed last) is your router’s IP. Your computer’s IP
                address is the number next to “IP Address.” If your computer’s IP address
                starts with 169, the computer is not receiving a valid IP address. See
                <xref href="#./ipaddress"/>.
            </info>
          </step>
          <step>
            <cmd>If your address does not start with 169, type
               <userinput>tracert8.8.8.8</userinput> to view each step between your router
               and the Google DNS servers.</cmd>
            <info>If the error comes up early along the pathway, see
                <xref href="#./resetnetwork"/></info>
          </step>
          <step>
            <cmd>If everything is working with Google, use the <cmdname>nslookup</cmdname>
               command to determine if there's a problem with the server you are trying
               to connect to.</cmd>
            <info>If you received results such as <msgph>Timed Out</msgph>,
                <msgph>Server Failure</msgph>, <msgph>Refused</msgph>,
                <msgph>No Response from Server</msgph>, or
                <msgph>Network is unreachable<msgph>, the problem originates in the DNS
                server for your destination.</info>
          </step>
          <step>
            <cmd>If the previous steps turn up no problems, contact your ISP to see if
              they're having issues.</cmd>
          </step>
        </steps>
      </diagnostics-steps>
    </diagnostics>
    <!-- The rest of this topic contains two <troublesolution> elements, each of which
         contains a remedy. One remedy provides instructions for "Resetting your IP address"
         and the other provides instructions for "Resetting your local network". The
         cross references in the above steps resolve to the <remedy> elements. -->
  </troublebody>
</troubleshooting>
```

4.2.7.6 <remedy>

The `<remedy>` element contains steps that are a potential solution for a problem. In addition, it also might contain information about the person or team who can perform the task.

Usage information

The `<remedy>` element is a component of a potential solution. The remedy might be omitted if there is no known remedy for the cause.

Specialization hierarchy

The `<remedy>` element is specialized from `<section>`. It is defined in the troubleshooting module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

In the following code sample, the `<remedy>` element contains instructions for how the responsible party can fix the problem:

```
<troubleshooting id="simple-example">
  <title>System will not turn on</title>
  <troublebody>
    <!-- . . . -->
    <troubleSolution>
      <!-- . . . -->
      <remedy>
        <title>Reset the circuit breaker</title>
        <responsibleParty>Maintenance technician</responsibleParty>
        <steps>
          <step><cmd>Power the system down.</cmd></step>
          <step><cmd>Reset the circuit breaker.</cmd></step>
          <step><cmd>Power the system back up.</cmd></step>
        </steps>
      </remedy>
    </troubleSolution>
  </troublebody>
</troubleshooting>
```

4.2.7.7 <responsibleParty>

The `<responsibleParty>` element identifies the individual or team whose task it is to perform a remedy procedure. It also can provide important information about the qualifications that the person or team must have.

Rendering expectations

Implementations might want to consider rendering a label for this element.

Specialization hierarchy

The `<responsibleParty>` element is specialized from `<p>`. It is defined in the troubleshooting module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

The following code sample shows how the `<responsibleParty>` element can be used to specify the prerequisites for performing a procedure:

```
<remedy>
  <responsible-party>You must have administrative privileges to perform this procedure.
  </responsible-party>
  <steps>
    <step><cmd>Run the following command from the command line:
               <codeph>msiexec.exe /I C:\Windows\Installer\XXXXX.msi</codeph></cmd>
    <step>
    <!-- ... -->
  </steps>
</remedy>
```

4.2.7.8 <troublebody>

The `<troublebody>` element contains the main content of the troubleshooting topic. The troubleshooting body can contain information about the condition, how to diagnose the cause, and one or more possible solutions.

Specialization hierarchy

The `<troublebody>` element is specialized from `<body>`. It is defined in the troubleshooting module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

See `<troubleshooting>` (107).

4.2.7.9 <troubleshooting>

The `<troubleshooting>` element is the top-level element for a troubleshooting topic. Troubleshooting topics provide information that enables readers to identify a condition, diagnose a cause, and potentially fix the problem.

Usage information

Troubleshooting topics begin with a description of a problem that the reader might want to correct. This can be followed by diagnostic information and possible solutions to the problem.

Specialization hierarchy

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

Attributes

The following attributes are available on this element: architectural attributes (178) and universal attributes (173).

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

Example

The following code sample contains a troubleshooting topic. The troubleshooting topic contains three `<troubleSolution>` elements that direct the user to perform sequential troubleshooting tasks: Resetting the alarm, reseating the system memory board, and replacing the memory board. Note that some steps are reused from other topics.

```
<troubleshooting id="E247" xml:lang="en-us">
  <title><msgph><msgnum>E247</msgnum>: Memory fault has occurred</msgph></title>
  <shortdesc>The system has detected a memory problem.</shortdesc>
  <troublebody>
    <condition>
      <p>The fault indicator flashes on the front panel, and the error log
        contains the <msgnum>E247</msgnum> error message.</p>
    </condition>
    <troubleSolution>
      <cause>p>A transient memory fault has occurred.</p></cause>
      <remedy>
        <responsibleParty>System administrator</responsibleParty>
        <steps>
          <step><cmd>From the systems management software, reset the alarm.</cmd></step>
          <step><cmd>Monitor the system periodically to see whether the alarm
              recurs.</cmd>
          </step>
        </steps>
      </remedy>
    </troubleSolution>
    <troubleSolution>
      <cause><p>A recurring memory fault indicates a possible problem with the
          system memory board.</p></cause>
      <remedy>
        <responsibleParty>Maintenance technician</responsibleParty>
        <steps conref="boardReseat.dita#boardReseat/steps">
          <step><cmd/></step>
        </steps>
      </remedy>
    </troubleSolution>
    <troubleSolution>
      <cause><p>The system memory board might be corrupted.</p></cause>
      <remedy>
        <responsibleParty>Certified technician. Note that work done by
```

```
            non-qualified individuals will void the product warranty.</responsibleParty>

         <steps conref="boardReplace.dita#boardReplace.dita/steps">
           <step><cmd/></step>
         </steps>
       </remedy>
     </troubleSolution>
   </troublebody>
</troubleshooting>
```

4.2.7.10 <troubleSolution>

Each <troubleSolution> element contains information about the cause of a problem and a potential remedy.

Usage information

The troubleshooting topic can contain multiple <troubleSolution> elements. A <troubleSolution> element can contain a cause, a remedy, or a cause and remedy pair. The cause might be omitted if it is implicit or if the remedy is not associated with a cause. The remedy might be omitted if there is no known remedy for the cause.

Specialization hierarchy

The <troubleSolution> element is specialized from <bodydiv>. It is defined in the troubleshooting module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

In the following code sample, the <troubleSolution> element contains a cause and remedy pair:

```
<troubleshooting id="e247">
  <title>E247: Memory fault has occurred</title>
  <troublebody>
    <troubleSolution>
      <cause>The <msgnum>E247</msgnum> error message is generated due to a
           transient memory fault.</cause>
      <remedy>
        <steps>
          <step><cmd>Reset the alarm.</cmd></step>
          <step><cmd>Monitor the system periodically to see whether the alarm recurs.</cmd></
step>
        </steps>
      </remedy>
    </troubleSolution>
  </troublebody>
</troubleshooting>
```

4.3 Domain specializations

Domains in this section include those generally associated with technical content, such as the programming and software domains.

4.3.1 Abbreviated-form domain elements

The abbreviated-form domain contains an element that can be used, in conjunction with a glossary entry topic, for rendering different versions of a term on first and later occurrences in a publication.

4.3.1.1 <abbreviated-form>

The `<abbreviated-form>` element represents a reference to a term that might appear in an abbreviated form. The abbreviated form often is an acronym.

Usage information

The `<abbreviated-form>` element typically is used in conjunction with a glossary entry topic that defines a term and an acronym. The glossary entry topic might also provide a surface form that specifies both the term and the acronym. The surface form is intended to be rendered on first use or in introductory contexts where the term might be unfamiliar to a reader. In other contexts, processors typically render the abbreviated form of the term. Note that the definition of an introductory context will differ for every deliverable format and is highly processor-specific.

For instance, a process composing a book deliverable might render the surface form of a term on the first reference to the `<glossentry>` topic within the book or for every reference within a copyright or a warranty-related warning. A process generating an online page might render the surface form as a hover tooltip on every instance of the term.

Rendering expectations

See Rendering of abbreviated-form elements (23).

Specialization hierarchy

The `<abbreviated-form>` element is specialized from `<term>`. It is defined in the abbreviated-form domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

This section contains examples of how the `<abbreviated-form>` element works in conjunction with a glossary entry topic that defines a term and its variations.

Figure 16: Markup for a glossary entry topic

The following code sample shows the markup for a simple glossary entry topic:

```
<glossentry id="id-attribute-value">
  <glossterm>Anti-lock Braking System</glossterm>
  <glossBody>
    <glossSurfaceForm>Anti-lock Braking System (ABS)</glossSurfaceForm>
    <glossAlt>
      <glossAcronym>ABS</glossAcronym>
    </glossAlt>
  </glossBody>
</glossentry>
```

For the purposes of rendering, the code sample contains three important elements:

<glossSurfaceform>
Defines the term as it should be rendered on first use. Typically this is the long form of a term, followed by an abbreviation or acronym. Note that other languages often do not follow the same conventions as English.

<glossAcronym>
Defines the terms as it should be rendered on second or later us. Typically this is the acronym or abbreviation that is associated with the term.

<glossterm>
Provides a fallback version of the term, which will be displayed in situations where the preferred representation is unavailable.

Figure 17: The glossary entry topic is associated with a key

In order for the `<abbreviated-form>` element to reference the glossary entry topic, the glossary entry topic must be associated with a key. This can happen using standard key definition, or a map architect can use the specialized `<glossref>` element"

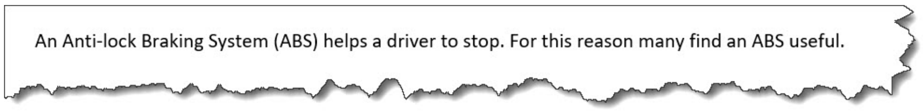
```
<glossref keys="abs" href="antilock.dita"/>
```

Figure 18: The <abbreviated-form> element references the key

The `<abbreviated-form>` element references the key defined for the glossary entry topic, for example:

```
<section>An <abbreviated-form keyref="abs"/> helps a driver to stop. For this reason
many find an <abbreviated-form keyref="abs"/> useful.
<!-- ... -->
</section>
```

The typical rendering is that the first use of the `<abbreviated-form keyref="ab">` will result in the surface form of the term, while subsequent usages will result in the acronym, as shown in the following screen capture:


An Anti-lock Braking System (ABS) helps a driver to stop. For this reason many find an ABS useful.

Do note, however, that processors implement varying levels of support for the `<abbreviated-form>` element.

4.3.2 Equation domain

The equation domain includes elements that authors can use to identify, number, and format equations within a document. This domain can be used independently of the MathML domain.

4.3.2.1 <equation-block>

The `<equation-block>` element represents an equation that is presented as a separate block within a text flow or an `<equation-figure>`.

Usage information

When an `<equation-block>` element has multiple direct child elements, each child represents an alternative form of the equation.

Rendering expectations

Block equations can be numbered.

Processing expectations

When there are multiple forms of an equation, processors can choose the form or forms that they render. For example, if there is both an image and MathML markup, an HTML-generating processor could generate both the image reference and the MathML with appropriate HTML `@class` or `@id` values to enable dynamic rendering that is based on browser capability.

Specialization hierarchy

The `<equation-block>` element is specialized from `<div>`. It is defined in the equation domain module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

The following code sample shows how an `<equation-block>` element can include two alternative forms of the same equation:

```
<equation-block>
  <!-- Imaged-based equation -->
  <image keyref="equation-image-01">
    <alt>a squared plus b squared.</alt>
  </image>
  <!-- MathML-based equation -->
  <mathml>
    <m:math>
      <m:semantics>
        <m:mrow>
          <m:msqrt>
            <m:mrow>
              <m:msup><m:mi>a</m:mi><m:mn>2</m:mn></m:msup>
              <m:mo>+</m:mo>
              <m:msup><m:mi>b</m:mi><m:mn>2</m:mn></m:msup>
            </m:mrow>
          </m:msqrt>
        </m:mrow>
      </m:semantics>
    </m:math>
  </mathml>
</equation-block>
```

4.3.2.2 <equation-figure>

The `<equation-figure>` element is a container for equations and their supporting information.

Usage information

Equation figures can have titles, descriptions, figure groups, and all other figure components. The direct children of `<equation-figure>` can be the equation content itself (for example, `<mathml>` or an image reference), or it can be one or more `<equation-block>` elements, along with other elements allowed within figures.

When an `<equation-figure>` element has multiple direct child `<mathml>`, `<equation-block>`, `<image>`, or `<pre>` elements, each child represents an alternative form of the equation.

When the intent is to combine equations and commentary within an `<equation-figure>`, use child `<equation-block>` elements to contain the equations and so clearly distinguish them from the commentary.

Rendering expectations

Equation figures can be numbered. Either standard figure numbering can be used, or `<equation-number>` elements can be placed within `<equation-block>` elements.

Processing expectations

When there are multiple forms of an equation, processors can choose the form or forms that they render. For example, if there is both an image and MathML markup, an HTML-generating processor could generate both the image reference and the MathML with appropriate HTML `@class` or `@id` values to enable dynamic rendering that is based on browser capability.

Specialization hierarchy

The `<equation-figure>` element is specialized from `<fig>`. It is defined in the equation domain module.

Attributes

The following attributes are available on this element: display attributes (179) and universal attributes (173).

Example

The following code sample shows how an `<equation-figure>` element can contain both MathML content and commentary. The MathML content is contained with a nested `<equation-block>` element, and it is followed by commentary that is contained in a nested `<p>` element.

```
<equation-figure>
 <title>An equation with commentary</title>
 <equation-block>
  <mathml>
   <m:math display='block'>
     <m:semantics>
       <m:mrow>
         <m:mfrac>
           <m:mrow><m:mi>n</m:mi><m:mo>!</m:mo></m:mrow>
           <m:mrow><m:mi>r</m:mi><m:mo>!</m:mo>
             <m:mrow>
               <m:mo>(</m:mo>
               <m:mrow><m:mi>n</m:mi><m:mo>&#x2212;</m:mo><m:mi>r</m:mi></m:mrow>
               <m:mo>)</m:mo>
             </m:mrow>
             <m:mo>!</m:mo>
           </m:mrow>
         </m:mfrac>
       </m:mrow>
     </m:semantics>
   </m:math>
  </mathml>
 </equation-block>
 <p>Where
   <equation-inline><mathml><m:math><m:mi>r</m:mi></m:math></mathml></equation-inline>
   is greater than 1.</p>
</equation-figure>
```

4.3.2.3 <equation-inline>

The `<equation-inline>` element represents an equation that is presented inline within a paragraph or similar context.

Usage information

Inline equations are not intended to be numbered.

When an `<equation-inline>` element has multiple direct child elements, each child represents an alternative form of the equation.

Processing expectations

When there are multiple forms of an equation, processors can choose the form or forms that they render. For example, if there is both an image and MathML markup, an HTML-generating processor could generate both the image reference and the MathML with appropriate HTML `@class` or `@id` values to enable dynamic rendering that is based on browser capability.

Specialization hierarchy

The `<equation-inline>` element is specialized from `<ph>`. It is defined in the equation domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Examples

This section contains examples of how the `<equation-inline>` element can be used.

Figure 19: An inline equation

The following code sample shows how a paragraph can contain an `<equation-inline>` element that holds MathML markup:

```
<p>Consider the following equation:
  <equation-inline>
    <mathml>
      <m:math display='inline'>
        <m:semantics>
          <m:mrow>
            <m:msqrt>
              <m:mrow>
                <m:msup><m:mi>a</m:mi><m:mn>2</m:mn></m:msup>
                <m:mo>+</m:mo>
                <m:msup><m:mi>b</m:mi><m:mn>2</m:mn></m:msup>
              </m:mrow>
            </m:msqrt>
          </m:mrow>
        </m:semantics>
      </m:math>
    </mathml>
  </equation-inline>
It is simple arithmetic that school children understand.</p>
```

Figure 20: An inline equation that is image-based

The following code sample shows how the `<equation-inline>` element can contain an image:

```
<p>The Pythagorean Theorem describes the relationship among the three sides of a
   right triangle. In any right triangle, the sum of the areas of the squares formed on the
   legs of the triangle equals the area of the square formed on the hypotenuse:
  <equation-inline>
    <image keyref="equation-image-01">
      <alt>a squared plus b squared.</alt>
    </image>
  </equation-inline>
</p>
```

4.3.2.4 <equation-number>

The `<equation-number>` element indicates that a block equation should be numbered. It optionally specifies the number to use for the block equation.

Usage information

In normal usage, a block equation has a single number. However, the `<equation-number>` element can occur multiple times within the `<equation-block>` element. This enables the use of numbers with different (and exclusive) conditional properties.

When the `<equation-figure>` element contains content, the content of the element should be the number value without any rendering-specific punctuation, for example, "3.2a" rather than "(3.2a)".

Rendering expectations

In this context, white-space content is considered equivalent to empty content. When the `<equation-number>` element has empty content, the equation number **SHOULD** be generated. When the `<equation-number>` element is not empty, the content **SHOULD** be used as the equation number. Processors **MAY** add punctuation or decoration to the number.

The details of equation numbering and number presentation are processor-specific. A common practice is to present the equation number to the right of the equation, centered vertically within the vertical extent of the block equation.

Specialization hierarchy

The `<equation-number>` element is specialized from `<ph>`. It is defined in the equation domain module.

Attributes

The following attributes are available on this element: universal attributes (173).

Examples

This section contains examples of how the `<equation-number>` element can be used:

Figure 21: An equation where the number will be generated

The following code sample shows how an `<equation-number>` element can be used to indicate to a processor that an equation number should be auto-generated:

```
<equation-block id="eq-001">
  <equation-number/>
  <image keyref="equation-image-01">
    <alt>a squared plus b squared.</alt>
  </image>
</equation-block>
```

Figure 22: An equation where the equation number is explicitly specified

The following code sample shows how an `<equation-number>` element can specify the value for an equation number:

```
<equation-block id="eq-3.2a">
  <equation-number>3.2a</equation-number>
  <image keyref="equation-image-01">
    <alt>a squared plus b squared.</alt>
  </image>
</equation-block>
```

4.3.3 Glossary-reference domain elements

4.3.3.1 <glossref>

Glossary reference topic references are a convenience element for creating references to glossary topics. It has a required `@keys` attribute, which forces the author to create a key by which inline terms can reflect the glossary terms and become navigable links to the glossary definition. For example, when `<glossentry>` topics are used to define acronyms, this reminds authors to create a key that `<abbreviated-form>` elements can use to refer to the short and expanded versions of the acronyms.

Usage information

Note that the key names defined in the `@keys` attribute do not need to match the target term or acronym. Using more qualified values or distinct key scopes for the keys will reduce conflicts in situations where the same key name might be associated with different terms. For example, an information set could use “cars.abs” as the key for the term Anti-lock Braking System, and “ship.abs” to refer to the term American Bureau of Shipping.

Specialization hierarchy

The `<glossref>` element is specialized from `<topicref>`. It is defined in the glossary reference domain module.

Attributes

The following attributes are available on this element: link-relationship attributes (179), universal attributes (173), `@chunk` (0), `@collection-type` (0), `@keyref` (0), `@linking` (0), `@processing-role` (0), `@search` (0), and `@toc` (0).

For this element:

- The `@href` attribute is a reference to a glossary definition, typically a `<glossentry>` topic.
- The `@keys` attribute is required.
- The `@linking` attribute has a default value of "none".
- The `@toc` attribute has a default value of "no".
- The `@search` attribute has a default value of "no".

Example

The following code sample demonstrates using a `<glossref>` element to include a car-specific glossary entry in the map. The glossary reference has the key "abs". The `<glossref>` element is within the key scope "cars", so references to this glossary entry from outside the "cars" key scope will be of the form `<keyword keyref="cars.abs"/>`.

```
<map>
  <!-- ... -->
  <topicref href="car-maintenance.dita"/>
  <!-- ... -->
  <topichead keyscope="cars">
   <topicmeta><navtitle>Car Terminology</navtitle></topicmeta>
   <glossref keys="abs" href="glossary/a/antiLockBrake.dita"/>
   <!-- ... key declarations for other referenced acronyms ... -->
  </topichead>
</map>
```

When this is map is rendered using typical output processing, the table of contents will have an entry for the topic head "Car Terminology" but will not have entries for the glossary entries because `<glossref>` sets the value of `@toc` to "no" by default.

4.3.4 Hardware domain

The hardware domain elements are used to document physical devices.

4.3.4.1 <hwcontrol>

A hardware control is a key, button, switch, indicator, or other physical device that controls a piece of hardware.

Specialization hierarchy

The `<hwcontrol>` element is specialized from `<ph>`. It is defined in the hardware domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how the `<hwcontrol>` element can be used to identify a hardware controls:

```
<step>
  <cmd>After entering the amount you received, press <hwcontrol>Amt Tend</hwcontrol>.</cmd>
  <stepresult>This opens the cash drawer. The display shows the amount of change to give the
      customer.</stepresult>
</step>
```

4.3.4.2 <partno>

A part number is a unique identifier that is assigned to a hardware component.

Specialization hierarchy

The `<partno>` element is specialized from `<ph>`. It is defined in the hardware domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how `<partno>` can be used to identify a part number:

```
<p>The basic model, <partno>DB-123-456</partno>, is an entry model. Most users
can take advantage of all features with little to no set up. The <partno>DB-123-456</partno>
is available with all systems.</p>
```

4.3.5 Markup domain

The markup domain contains an element that can be used for the mention of named components in markup languages.

4.3.5.1 <markupname>

The `<markupname>` element identifies named markup components, for example, elements or attributes in HTML and SGML, named groups in XSD schemas, and named patterns in RELAX NG schemas.

Usage information

The `<markupname>` element serves as the specialization basis for the elements in the XML mention domain. When the XML mention domain is present, use its more specific elements instead of `<markupname>` if appropriate.

Specialization hierarchy

The `<markupname>` element is specialized from `<keyword>`. It is defined in the markup domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how the `<markupname>` element can be used to tag an attribute group:

```
The <markupname>p.attributes</markupname> attribute group defines
the allowed attributes for the <xmlelement>p</xmlelement> element.
```

4.3.6 MathML domain

The MathML domain elements enable the use of embedded or referenced MathML markup. Referenced MathML markup must be stored in separate, non-DITA XML documents. MathML is a W3C standard.

When MathMLelements are embedded in DITA documents that are validated using DTDs, the MathML elements must use a namespace prefix in order to avoid conflict with the DITA-defined elements of the same name. Documents validated using RELAX NG can default the MathML namespace on the MathML `<math>` element. MathML elements that are referenced using the `<mathmlref>` element do not need to have a namespace prefix, because they are parsed separately from the DITA documents that refer to them.

By default, the MathML domain is configured to use the namespace prefix "m" for the MathML elements.

Related information

Mathematical Markup Language (MathML), Version 3.0

4.3.6.1 <mathml>

The `<mathml>` element contains MathML markup or other content that contributes to a semantic equation.

Usage information

The `<mathml>` element can contain MathML elements, references to MathML elements stored in separate, non-DITA documents, or `<data>` elements.

The `<mathml>` element is not intended to represent a semantic equation, only content that contributes to a semantic equation. Use the equation domain elements or their equivalent to represent equations semantically, for example, to enable numbering of equations.

The MathML markup must have a root element of `<math>` within the MathML namespace: `http://www.w3.org/1998/Math/MathML`.

Specialization hierarchy

The `<mathml>` element is specialized from `<foreign>`. It is defined in the MathML domain module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

The following code sample shows how to use a `<mathml>` element to include MathML content:

```
<equation-block>
  <mathml>
    <m:math>
      <m:semantics>
        <m:mrow>
          <m:msqrt>
            <m:mrow>
              <m:msup>
                <m:mi>a</m:mi>
                <m:mn>2</m:mn></m:msup>
                <m:mo>+</m:mo>
                <m:msup>
                  <m:mi>b</m:mi>
```

```
                   <m:mn>2</m:mn>
                 </m:msup>
               </m:mrow>
             </m:msqrt>
           </m:mrow>
         </m:semantics>
       </m:math>
     </mathml>
</equation-block>
```

4.3.6.2 <mathmlref>

The `<mathmlref>` element references a non-DITA XML document that contains MathML markup.

Usage information

The `<mathmlref>` element enables the use MathML markup by reference. The reference must be to a MathML `<math>` element. The reference can be one of the following:

- A URI that addresses an XML document. The XML document has a MathML `<math>` element as the root element.
- A URI that addresses an XML document and contains a fragment identifier that is the XML ID of a `<math>` element within the document.

The reference can be direct, using the `@href` attribute, or indirect, using the `@keyref` attribute. For indirect referencing, only the key name should be specified. The ID of the `<math>` element must be specified as part of the value for the `@href` attribute on the key definition.

For example, to refer to the `<math>` element with the `@id` of "math-fragment-02" within a larger document using a key reference, you would define the key in the following way:

```
<keydef keys="math-fragment-0002" href="mathml/mathml-library.xml#math-fragment-02"/>
```

You reference this key by using just the key name:

```
<mathref keyref="math-fragment-0002"/>
```

Processing expectations

Processors **SHOULD** process the MathML as though the `<m:math>` element occurs directly in the content of the containing `<mathml>` element.

Specialization hierarchy

The `<mathmlref>` element is specialized from `<include>`. It is defined in the MathML domain module.

Attributes

The following attributes are available on this element: inclusion attributes (179), universal attributes (173), `@format` (0), `@href` (0), `@keyref` (0), and `@scope` (0).

For this element:

- The `@format` attribute has a default value of "mml".
- The `@href` attribute is a reference to a MathML document or `<mathml>` element. If the `<mathml>` element is the root element of the referenced resource, then no fragment identifier is

required. Otherwise, a fragment identifier must be specified, where the fragment identifier is the XML ID of the `<mathml>` element.

- The `@parse` attribute has a default value of "xml".

Examples

This section contains examples of how the `<mathmlref>` element can be used.

Figure 23: Referencing a MathML <math> root element

The following code sample shows how a `<mathmlref>` element can be used to reference a MathML `<math>` element that is the root element of its containing document:

```
<equation-block>
  <mathml>
    <mathmlref href="../mathml-source/mathml-root-mathml.mml"/>
  </mathml>
</equation-block>
```

The `mathml-root-mathml.mml` file contains the following content. Note that the `<math>` element sets the MathML namespace as the default namespace, so there are no namespace prefixes on the MathML markup.

```
<?xml version="1.0" encoding="UTF-8"?>
<math xmlns="http://www.w3.org/1998/Math/MathML" xmlns:xlink="http://www.w3.org/1999/xlink">
  <mstyle displaystyle="false" scriptlevel="0">
    <mrow>
      <mfrac>
        <mrow>
          <mi mathcolor="gray">sin</mi>
          <mo rspace="verythinmathspace"/>
          <mi>θ</mi>
        </mrow>
        <mi>π</mi>
      </mfrac>
    </mrow>
  </mstyle>
</math>
```

Figure 24: Referencing a specific <math> element within a document

The following code sample shows how a `<mathmlref>` element can reference a specific `<math>` element in a containing XML file:

```
<equation-block>
  <mathml>
    <mathmlref href="../mathml-source/mathml-equation-library.xml#mathfrag-02"/>
  </mathml>
</equation-block>
```

The `mathml-equation-library.xml` file contains the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <part>
    <math id="timeinday" xmlns="http://www.w3.org/1998/Math/MathML">
      <mi>x</mi>
    </math>
    <math id="mathfrag-02" xmlns="http://www.w3.org/1998/Math/MathML">
      <math>
        <mrow>
          <mi>y</mi>
          <mo>=</mo>
          <mn>5</mn>
          <mi>x</mi>
          <mo>+</mo>
          <mn>2</mn>
        </mrow>
      </math>
    </math>
  </part>
  <!-- ... -->
</root>
```

4.3.7 Programming domain

The programming domain elements are used to describe the syntax for programming languages.

4.3.7.1 <apiname>

An API name is the name of an application programming interface (API), such as a Java class name or method name.

Specialization hierarchy

The `<apiname>` element is specialized from `<keyword>`. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0   ).

Example

The following code sample shows how the `<apiname>` element can be used to identify the `document.write` method:

```
<p>Use the <apiname>document.write</apiname> method to create text
output in the dynamically constructed view.</p>
```

4.3.7.2 <codeblock>

A code block is a set of lines from a program.

Rendering expectations

Processors **SHOULD** preserve line the breaks and spaces that are present in the content of a `<codeblock>` element.

The content of the `<codeblock>` element is typically rendered in a monospaced font.

Specialization hierarchy

The `<codeblock>` element is specialized from `<pre>`. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: display attributes (179), universal attributes (173), and `@xml:space` (190).

Example

The following code sample shows how the `<codeblock>` element can be used to tag an excerpt from an XSLT stylesheet:

```
<codeblock>
  &lt;xsl:template match="*[contains(@outputclass,'green')]">
    &lt;xsl:attribute name="color">#006400;&lt;/xsl:attribute>
  &lt;/xsl:template>
</codeblock>
```

For a sample of how this element can be combined with `<coderef>` to embed external code samples, see coderef (122).

4.3.7.3 <codeph>

A code phrase is a small portion of source code, machine code, or text that is displayed in-line.

Rendering expectations

The content of the `<codeph>` element is typically rendered in a monospaced font.

Specialization hierarchy

The `<codeph>` element is specialized from `<ph>`. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

In the following code sample, the `<codeph>` element identifies a code snippet. The code snippet will be rendered in-line in the paragraph.

```
<p>The second line of the sample program code, <codeph>Do forever</codeph>,
represents the start of a loop construct.</p>
```

4.3.7.4 <coderef>

A code reference is the mechanism for referencing an external text file that contains program code.

Rendering expectations

When evaluated, the `<coderef>` element causes the target code to be displayed inline. If the target code contains non-XML characters such as '<' or '&', those characters need to be handled so that they can be displayed correctly by the final rendering engine.

Specialization hierarchy

The `<coderef>` element is specialized from `<include>`. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: inclusion attributes (179), link-relationship attributes (179), universal attributes (173), and `@keyref` (0).

For this element, the `@parse` attribute has a default value of "text".

Example

In the following code sample, the `<coderef>` element references the content of the `process-dita.xsl` file. In the rendered output, the XSL code will be presented in a code block.

```
<example>
  <title>Processing DITA</title>
  <p>This code is an example of how to process DITA.</p>
  <codeblock>
    <coderef href="process-dita.xsl"/>
  </codeblock>
</example>
```

4.3.7.5 <option>

The `<option>` element specifies a permitted value for a parameter or configuration.

Specialization hierarchy

The `<option>` element is specialized from `<keyword>`. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

This section contains examples of how the `<option>` element can be used.

Figure 25: Parameter values

In the following code sample, the `<option>` element is used to specify permitted values for a parameter:

```
<p> You can use the <parmname>map-order</parmname> parameter to specify
if the order of the frontmatter and backmatter content is retained. The allowed
values are <option>keep</option> and <option>trash</option>; the default value is
<option>trash</option>.</p>
```

Figure 26: Configuration option

In the following code sample, the `<option>` element is used to specify an option that modifies a configuration:

```
<p>You can purchase the <option>Sports Model</option> package in order
to configure your automobile with the latest sports suspension, bonnet stripes,
a roof spoiler and an aero body kit.</p>
```

4.3.7.6 <parmname>

A parameter name is the name of a parameter that is passed to an API or a command-line interface (CLI).

Specialization hierarchy

The `<parmname>` element is specialized from `<keyword>`. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0   ).

Example

The following code sample shows how the `<parmname>` element can be used to identify a parameter that is used with the `config` command:

```
<p>You can use the <parmname>-env</parmname> parameter of the <cmdname>config</cmdname>
command to update the field value.</p>
```

4.3.7.7 <parml>

A parameter list is a specialized definition list that is designed for documenting parameters.

Specialization hierarchy

The `<parml>` element is specialized from `<dl>`. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@compact` (182).

Example

The following code sample shows how a set of sample code is followed by a parameter list that defines those parameters:

```
<p>This code example is a basic method signature:</p>
<codeblock>returnType methodName(pList1, pList2)</codeblock>
<p>The method requires the following parameters:</p>
<parml>
  <plentry>
    <pt>pList1</pt>
    <pd>The first variable declaration that is passed to methodName</pd>
  </plentry>
  <plentry>
    <pt>pList2</pt>
    <pd>The second variable declaration that is passed to methodName</pd>
  </plentry>
</parml>
```

4.3.7.8 <plentry>

A parameter-list entry contains one or more parameter terms and definitions.

Specialization hierarchy

The `<plentry>` element is specialized from `<dlentry>`. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

See `<parml>` (124).

4.3.7.9 <pt>

A parameter term is a term that is defined in a parameter-list entry.

Specialization hierarchy

The `<pt>` element is specialized from `<dt>`. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

See `<parml>` (124).

4.3.7.10 <pd>

A parameter definition is a definition of a term that is defined in a parameter-list entry.

Specialization hierarchy

The `<pd>` element is specialized from `<dd>`. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

See `<parml>` (124).

4.3.8 Release management domain

The release-management domain elements contain information about the changes that have been made to a DITA topic or map. This information can be retrieved and used to assemble a list of changes.

4.3.8.1 <change-completed>

The `<change-completed>` element specifies the date on which the change was completed.

Usage information

The recommended best practice is to use date strings that conform to the ISO 8601 standard, unless an epoch timestamp is used. The string might contain a date and time (for example, `2017-04-05T12:30-02:00`) or just a date (for example, `2019-03-04`).

Specialization hierarchy

The `<change-completed>` element is specialized from `<data>`. It is defined in the release-management domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@name` (186).

Example

The following code sample shows how the `<change-completed>` element can be used to note when the change was completed:

```
<change-historylist>
  <change-item product="productA productB"><change-person>Tom Cihak</change-person>
    <change-organization>JEDEC</change-organization>
    <change-started>2019-01-15</change-started>
    <change-completed>2019-03-23</change-completed>
    <change-summary>Made change 1 to both products</change-summary>
    <data>Details of change 1</data>
  </change-item>
<change-historylist>
```

4.3.8.2 <change-historylist>

The `<change-historylist>` element contains one or more changes that are applicable to the DITA topic or map.

Specialization hierarchy

The `<change-historylist>` element is specialized from `<metadata>`. It is defined in the release-management domain module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

The following code sample shows how the `<change-historylist>` element can be used to document one or more changes in a DITA topic or map. This example includes three changes. This topic is used in documentation for two products, A and B.

```
<change-historylist>
    <change-item product="productA productB">
      <change-person>Tom Cihak</change-person>
      <change-organization>JEDEC</change-organization>
      <change-started>2019-01-15</change-started>
      <change-completed>2019-03-23</change-completed>
      <change-summary>Made change 1 to both products</change-summary>
      <data>Details of change 1</data>
    </change-item>
    <change-item product="productA">
      <change-person>Tom Cihak</change-person>
      <change-completed>2019-06-07</change-completed>
      <change-summary>Made change 2 to product A</change-summary>
      <data>Details of change 2</data>
    </change-item>
    <change-item product="productA productB">
      <change-person>Tom Cihak</change-person>
      <change-revisionid>r23.4</change-revisionid>
      <change-request-reference>
        <change-request-system>example.com/my/queue/</change-request-system>
        <change-request-id>TCKT-1313</change-request-id>
      </change-request-reference>
      <change-completed>2019-07-20</change-completed>
      <change-summary>Made change 3 to both products</change-summary>
      <data>Details of change 3</data>
    </change-item>
  </change-historylist>
```

4.3.8.3 <change-item>

The `<change-item>` element represents a record of a change to a DITA topic or map.

Specialization hierarchy

The `<change-item>` element is specialized from `<data>`. It is defined in the release-management domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@name` (186).

Example

The following code sample shows how the `<change-item>` element can be used to detail a change. This example includes two changes, one that is applicable to products A and B and one that is applicable to only product A.

```
<change-historylist>
  <change-item product="productA productB">
    <change-person>Tom Cihak</change-person>
    <change-organization>JEDEC</change-organization>
    <change-started>2019-01-15</change-started>
    <change-completed>2019-03-23</change-completed>
    <change-summary>Made change 1 to both products</change-summary>
    <data>Details of change 1</data>
  </change-item>
  <change-item product="productA">
    <change-person>Tom Cihak</change-person>
    <change-completed>2019-06-07</change-completed>
    <change-summary>Made change 2 to product A</change-summary>
    <data>Details of change 2</data>
  </change-item>
</change-historylist>
```

4.3.8.4 <change-organization>

The `<change-organization>` element specifies the name of the business unit that required the change.

Specialization hierarchy

The `<change-organization>` element is specialized from `<data>`. It is defined in the release-management domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@name` (186).

Example

The following code sample shows how the `<change-organization>` element can be used to specify the business unit that required the change:

```
<change-historylist>
    <change-item product="productA productB">
      <change-person>Tom Cihak</change-person>
      <change-organization>JEDEC</change-organization>
      <change-started>2019-01-15</change-started>
      <change-completed>2019-03-23</change-completed>
        <change-summary>Made change 1 to both products</change-summary>
        <data>Details of change 1</data>
      </change-item>
    </change-historylist>
```

4.3.8.5 <change-person>

The `<change-person>` element specifies the name of the person who made the change.

Specialization hierarchy

The `<change-person>` element is specialized from `<data>`. It is defined in the release-management domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@name` (186).

Example

The following code sample shows how the `<change-person>` element can be used to specify who made the change:

```
<change-historylist>
    <change-item product="productA productB">
      <change-person>Tom Cihak</change-person>
      <change-organization>JEDEC</change-organization>
      <change-started>2019-01-15</change-started>
      <change-completed>2019-03-23</change-completed>
      <change-summary>Made change 1 to both products</change-summary>
      <data>Details of change 1</data>
    </change-item>
  </change-historylist>
```

4.3.8.6 <change-request-id>

The `<change-request-id>` element specifies an identifier associated with the change request, such as an issue ID or ticket number.

Specialization hierarchy

The `<change-request-id>` element is specialized from `<data>`. It is defined in the release-management domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@name` (186).

Example

The following code sample shows how the `<change-request-id>` element can be used to specify a ticket ID that is applicable to the change request:

```
<change-historylist>
  <change-item product="productA productB">
    <change-request-reference>
      <change-request-system>example.com/my/queue/</change-request-system>
      <change-request-id>TCKT-1313</change-request-id>
    </change-request-reference>
  </change-item>
</change-historylist>
```

4.3.8.7 <change-request-reference>

The `<change-request-reference>` element contains details about the change request, such as an ID or a reference to the system used to track the request.

Specialization hierarchy

The `<change-request-reference>` element is specialized from `<metadata>`. It is defined in the release-management domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@name` (186).

Example

The following code sample shows how the `<change-request-reference>` element can be used to specify the ticketing system used to manage the change request as well as the applicable ticket ID:

```
<change-historylist>
    <change-item product="productA productB">
      <change-request-reference>
        <change-request-system>example.com/my/queue/</change-request-system>
        <change-request-id>TCKT-1313</change-request-id>
      </change-request-reference>
    </change-item>
  </change-historylist>
```

4.3.8.8 <change-request-system>

The `<change-request-system>` element specifies the name of an information system that manages or serves the referenced change request, for example, an issue tracking system.

Specialization hierarchy

The `<change-request-system>` element is specialized from `<data>`. It is defined in the release-management domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@name` (186).

Example

The following code sample shows how the `<change-request-system>` element can be used to specify the the ticketing system that manages the change request:

```
<change-historylist>
    <change-item product="productA productB">
      <change-request-reference>
        <change-request-system>example.com/my/queue/</change-request-system>
        <change-request-id>TCKT-1313</change-request-id>
      </change-request-reference>
    </change-item>
  </change-historylist>
```

4.3.8.9 <change-revisionid>

The `<change-revisionid>` element specifies a string to identify the change.

Specialization hierarchy

The `<change-revisionid>` element is specialized from `<data>`. It is defined in the release-management domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@name` (186).

Example

The following code sample shows how the `<change-revisionid>` element can be used to identify a particular change. In this example, change revision ID is used to note the release number for which the change is applicable.

```
<change-historylist>
  <change-item product="productA productB">
    <change-person>Tom Cihak</change-person>
    <change-revisionid>r23.4</change-revisionid>
    <change-completed>2019-07-20</change-completed>
    <change-summary>Made change 3 to both products</change-summary>
    <data>Details of change 3</data>
  </change-item>
</change-historylist>
```

4.3.8.10 <change-started>

The `<change-started>` element specifies the date on which a change began.

Usage information

The recommended best practice is to use date strings that conform to the ISO 8601 standard, unless an epoch timestamp is used. The string might contain a date and time (for example, `2017-04-05T12:30-02:00`) or just a date (for example, `2019-03-04`).

Specialization hierarchy

The `<change-started>` element is specialized from `<data>`. It is defined in the release-management domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@name` (186).

Example

The following code sample shows how the `<change-started>` element can be used to specify the first day a particular change began:

```
<change-historylist>
    <change-item product="productA productB">
      <change-person>Tom Cihak</change-person>
      <change-organization>JEDEC</change-organization>
      <change-started>2019-01-15</change-started>
      <change-completed>2019-03-23</change-completed>
      <change-summary>Made change 1 to both products</change-summary>
      <data>Details of change 1</data>
    </change-item>
  </change-historylist>
```

4.3.8.11 <change-summary>

The <change-summary> element includes a brief description of the change.

Usage information

The <change-summary> element contains the portion of the release note that might appear in a document.

Comment by tammy
How/when would the change-summary appear in a document?

Specialization hierarchy

The <change-summary> element is specialized from <data>. It is defined in the release-management domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and @name (186).

Example

The following code sample shows how the <change-summary> element can be used to provide a brief description of a change:

```
<change-historylist>
  <change-item product="productA">
    <change-person>Tom Cihak</change-person>
    <change-completed>2019-06-07</change-completed>
    <change-summary>Made change 2 to product A</change-summary>
    <data>Details of change 2</data>
  </change-item>
<change-historylist>
```

4.3.9 Software domain

The software domain elements are used to describe the presentation and operation of a software program.

4.3.9.1 <cmdname>

A command name is the name of a software command.

Specialization hierarchy

The <cmdname> element is specialized from <keyword>. It is defined in the software domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and @keyref (0).

Example

The following code sample shows a `<cmdname>` element that identifies the name of the `rm` command.

```
<p>You can use the <cmdname>rm</cmdname> command to permanently delete an object.</p>
```

4.3.9.2 <filepath>

A file path is the location of a resource, such as the system path and file name of a file on a storage device.

Rendering expectations

The content of the `<filepath>` element is typically rendered in a monospaced font.

Specialization hierarchy

The `<filepath>` element is specialized from `<ph>`. It is defined in the software domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0   ).

Example

In the following code sample, the `<filepath>` element is used to tag both file names and system paths:

```
<p>Uncompress the <filepath>gbbrsh.gz</filepath> file to the
<filepath>/usr</filepath> directory. Ensure that the
<filepath>/usr/tools/data.cfg</filepath> path is listed in
the execution path system variable.</p>
```

4.3.9.3 <msgblock>

A message block is a multi-line message or set of messages that is produced by an application or device.

Usage information

The `<msgblock>` element can contain multiple message numbers and message descriptions, each enclosed in `<msgnum>` and `<msgph>` elements. It can also contain the message content directly.

Rendering expectations

Processors **SHOULD** preserve the line breaks and spaces that are present in the content of a `<msgblock>` element.

The content of the `<msgblock>` element is typically rendered in a monospaced font.

Specialization hierarchy

The `<msgblock>` element is specialized from `<pre>`. It is defined in the software domain module.

Attributes

The following attributes are available on this element: display attributes (179), universal attributes (173), and `@xml:space` (190).

Example

The following code sample shows a `<msgblock>` element that contains a multi-line message that is returned by an application:

```
<p>A sequence of failed password attempts generates the following message stream:</p>
<msgblock>
I:0
S:3
I:1
S:3
I:1
S:4
S:99 (lockup)
</msgblock>
```

4.3.9.4 <msgnum>

A message number is the identifier of a message that is produced by an application or program.

Specialization hierarchy

The `<msgnum>` element is specialized from `<keyword>`. It is defined in the software domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows a `<msgnum>` element that identifies the number of the message that is returned by an application:

```
<p>A server log entry of <msgnum>I:0</msgnum> is equivalent to the
text message <msgph>informational: successful</msgph>.</p>
```

4.3.9.5 <msgph>

A message phrase is the text of a message that is produced by an application or program.

Specialization hierarchy

The `<msgph>` element is specialized from `<ph>`. It is defined in the software domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how the `<msgph>` element can be used to tag a message that is returned by the server:

```
<p>A server log entry of <msgnum>I:0</msgnum> is equivalent to the
text message, <msgph>informational: successful</msgph>.</p>
```

4.3.9.6 <systemoutput>

System output is content that a computer or device generates in response to a command or situation.

Specialization hierarchy

The `<systemoutput>` element is specialized from `<ph>`. It is defined in the software domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

In the following code sample, the `<systemoutput>` element identifies an application response to user input:

```
<p>After you type <userinput>mealplan dinner</userinput>, the meal planning program
will print <systemoutput>For what day?</systemoutput>.
Reply by typing the day of the week for which you want a meal plan,
for example, <userinput>Thursday</userinput>.</p>
```

4.3.9.7 <userinput>

User input is text that a user enters, such as a response to an application or system prompt.

Specialization hierarchy

The `<userinput>` element is specialized from `<ph>`. It is defined in the software domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

In the following code sample, the `<userinput>` element identifies text that a user should type at the command prompt:

```
<p>From a DOS command prompt, type <userinput>dir</userimput> to view a list
of files in the current directory.</p>
```

4.3.9.8 <varname>

A variable name is a placeholder for content that might change based on how something is used, such as a variable supplied to a software application or a user-defined path in a command.

Rendering expectations

The content of the `<varname>` element is typically rendered in an italic font.

Specialization hierarchy

The `<varname>` element is specialized from `<keyword>`. It is defined in the software domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how the `<varname>` element is used to identify variables that represent the "installation directory," "project directory," and "file name":

```
<filepath>
  <varname>install-dir</varname>/projects/working/<varname>project-dir</varname>
       /source/<varname>filename</varname>.java
</filepath>
```

4.3.10 SVG domain

The SVG domain elements enable the use of embedded or referenced SVG markup. Referenced SVG markup must be stored in separate, non-DITA XML documents. SVG is a W3C standard.

For SVG markup that is stored directly in DITA documents that are validated using DTDs, the SVG elements must use a namespace prefix in order to avoid conflict with DITA-defined elements of the same name. Documents validated using RELAX NG can default the SVG namespace on the SVG `<svg>` element. SVG elements that are referenced using the `<svgref>` element do not need to have a namespace prefix, because they are parsed separately from the DITA documents that refer to them. By default, the SVG domain is configured to use the namespace prefix "svg" for the SVG elements.

Related information

Scalable Vector Graphics (SVG) 1.1 (Second Edition)

4.3.10.1 <svg-container>

The `<svg-container>` element stores content that contributes to a scalable vector graphic (SVG).

Usage information

The `<svg-container>` element can contain SVG elements, references to SVG elements that are stored in separate, non-DITA documents, or `<data>` elements.

The SVG markup must have a root element of `<svg>` with the SVG namespace: "http://www.w3.org/2000/svg".

Specialization hierarchy

The `<svg-container>` is specialized from `<foreign>`. It is defined in the SVG domain module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

The following code sample shows how `<svg-container>` elements can be used in a DITA topic. It is used to generate both inline SVG markup and a titled figure that contains SVG markup:

```
<topic id="svg-test-topic-01">
  <title>SVG Domain Test: Namespace Prefixed SVG Elements</title>
    <body>
```

```
      <!-- SVG inline -->
      <svg-container>
        <svg:svg width="100" height="100">
          <svg:defs>
            <svg:filter id="f1" x="0" y="0">
              <svg:feGaussianBlur in="SourceGraphic" stdDeviation="15"/>
            </svg:filter>
          </svg:defs>
          <svg:rect width="90" height="90" stroke="green" stroke-width="3" fill="yellow"
                    filter="url(#f1)"/>
        </svg:svg>
      </svg-container>
      <!-- ... -->
      <fig>
        <title>Figure with SVG container</title>
        <svg-container>
          <svg:svg width="4in" height="6in" version="1.1">
            <svg:circle cx="150" cy="200" r="100" fill="url(#grad_blue)"/>
            <svg:rect x="70" y="320" height="40" width="80" fill="aqua"/>
            <svg:text x="90" y="350" font-size="30" fill="green">Go</svg:text>
          </svg:svg>
        </svg-container>
      </fig>
    </body>
</topic>
```

4.3.10.2 <svgref>

The `<svgref>` element references a non-DITA XML document that contains scalable vector graphic (SVG) markup.

Usage information

The `<svgref>` element enables the use of SVG markup by reference. The reference must be to a SVG `<svg>` element that is stored in a separate, non-DITA XML document. The reference can be one of the following:

- A URI that addresses an XML document which has a SVG `<svg>` element as the root element
- A URI that addresses an XML document and contains a fragment identifier that is the XML ID of a `<svg>` element within the document

The reference can be direct, using the `@href` attribute, or indirect, using the `@keyref` attribute. For indirect referencing, only the key name should be specified. The ID of the `<svg>` element must be specified as part of the value for the `@href` attribute on the key definition.

For example, to refer to the `<svg>` element with the `@id` of "svg-fragment-02" within a larger document using a key reference, you would define the key in the following way:

```
<keydef keys="svg-fragment-0002" href="svg/svg-library.xml#svg-fragment-02"/>
```

You reference this key by using just the key name:

```
<svg-container>
  <svgref keyref="svg-fragment-0002"/>
</svg-container>
```

Processing information

Processors **SHOULD** process the SVG as though the `<svg>` element occurs directly in the content of the containing `<svg-container>` element.

Specialization hierarchy

The `<svgref>` is specialized from `<include>`. It is defined in the SVG domain module.

Attributes

The following attributes are available on this element: inclusion attributes (179), link-relationship attributes (179), universal attributes (173), and `@keyref` (0).

For this element:

- The `@format` attribute has a default value of "svg".
- The `@href` attribute is a reference to an SVG document or SVG element. If the `<svg>` element is the root element of the referenced resource, then no fragment identifier is required. Otherwise, a fragment identifier must be specified, where the fragment identifier is the XML ID of the `<svg>` element.
- The `@parse` attribute has a default value of "xml".

Examples

This section contains examples of how the `<svgref>` element can be used.

Figure 27: Referencing an SVG that is a root element

The following code sample shows how an `<svgref>` element can be used to reference an `<svg>` element that is the root element of its containing document:

```
<fig>
  <title>Figure with an SVG container</title>
  <svg-container>
    <svgref href="media/svg/svg-graphic-01.xml"/>
  </svg-container>
</fig>
```

The `svg-graphic-01.xml` file contains the following content. Note that the `<svg>` element sets the SVG namespace as the default namespace, so there are no namespace prefixes on the SVG markup.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" width="100" height="100">
  <defs>
    <filter id="f1" x="0" y="0">
      <feGaussianBlur in="SourceGraphic" stdDeviation="15"/>
    </filter>
  </defs>
  <rect width="90" height="90" stroke="green" stroke-width="3" fill="yellow"
        filter="url(#f1)"/>
</svg>
```

Figure 28: Referencing a specific SVG within a document

The following code sample shows an `<svgref>` element can be used to reference a specific `<svg>` element in a containing XML file:

```
<fig>
  <title>Figure with SVG container</title>
  <svg-container>
    <svgref href="media/svg/svg-library.xml#frag-0001" />
  </svg-container>
</fig>
```

The `svg-library.xml` file contains the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <part>
    <svg id="frag-0001" xmlns="http://www.w3.org/2000/svg" width="100" height="100">
      <defs>
        <filter id="f1" x="0" y="0">
          <feGaussianBlur in="SourceGraphic" stdDeviation="15"/>
        </filter>
      </defs>
      <rect width="90" height="90" stroke="green" stroke-width="3" fill="yellow"
            filter="url(#f1)"/>
    </svg>
    <!-- ... -->
  </part>
</root>
```

4.3.11 Syntax diagram domain

The syntax-diagram domain elements are used to diagram expressions or syntax phrases for programming languages or command line processors.

The syntax diagram domain is specialized from the programming domain, and uses elements from that domain within the content model of syntax phrases.

4.3.11.1 <delim>

A delimiter is a character that marks the beginning or end of a section within a syntax diagram.

Usage information

Typical delimiter characters are parentheses, commas, tabs, vertical bars, or other special characters.

Specialization hierarchy

The `<delim>` element is specialized from `<ph>`. It is defined in the syntax-diagram domain module, which is a specialization of the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173).

For this element, the `@importance` attribute indicates whether this item in a syntax diagram is optional or required. The attribute value is limited to "optional", "required", or "-dita-use-conref-target".

Example

The following code sample shows how the `<delim>` element can be used to specify that the equal sign (=) is used to mark the end of the group sequence:

```
<syntaxdiagram>
  <title>Integer addition</title>
  <groupseq>
    <var>integer</var>
    <oper>+</oper>
    <var>integer</var>
    <delim>=</delim>
    <var>total</var>
  </groupseq>
</syntaxdiagram>
```

4.3.11.2 <fragment>

The `<fragment>` element contains a labeled subpart of the syntax within a syntax diagram.

Usage information

The `<fragment>` element enables breaking out logical chunks of a large syntax diagram into named fragments.

Specialization hierarchy

The `<fragment>` element is specialized from `<figgroup>`. It is defined in the syntax-diagram domain module, which is a specialization of the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

The following code sample shows how the `<fragment>` element can be used to break out a set of related logging options from the larger set of syntax. This allows the logging options to be displayed separately in a titled group, out of the flow of the primary diagram.

```
<syntaxdiagram>
  <title>Syntax for runprogram command</title>
  <groupseq>
    <kwd>runprogram</kwd>
    <groupchoice>
      <groupcomp><oper>-</oper><kwd>i</kwd><sep>:</sep><var>program-name.py</var></groupcomp>
      <groupcomp><oper>--</oper><kwd>input</kwd><sep>=</sep><var>program-name.py</var></
groupcomp>
    </groupchoice>
  </groupseq>
  <fragment>
    <title>Logging options</title>
    <groupseq importance="optional"><oper>--</oper><kwd>debug</kwd></groupseq>
    <groupseq importance="optional"><oper>--</oper><kwd>verbose</kwd></groupseq>
  </fragment>
</syntaxdiagram>
```

4.3.11.3 <fragref>

A fragment reference is the mechanism for referencing a syntax fragment within the same syntax diagram.

Usage information

The `<fragref>` element is used to reference a syntax fragment multiple times or pull a large section of syntax out of line for easier reading.

Specialization hierarchy

The `<fragref>` element is specialized from `<xref>`. It is defined in the syntax-diagram domain module, which is a specialization of the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@href` (0).

For this element:

- The `@href` attribute is a reference to a syntax diagram `<fragment>`. The referenced `<fragment>` must be in the same diagram as the `<fragref>` element.
- The `@importance` attribute is limited to the values "optional", "required", or "-dita-use-conref-target".

Example

The following code sample shows how the `<fragref>` element can be used to break out a set of related logging options from the larger set of syntax. The `<fragref>` element is part of the program sequence after an input file, but the syntax for logging is defined outside of the main diagram.

```
<syntaxdiagram frame="none">
  <title>CopyFile</title>
  <groupseq><kwd>COPYF</kwd></groupseq>
  <groupcomp><var>input-filename</var><kwd>*INFILE</kwd></groupcomp>
  <groupseq><var>output-filename</var><kwd>*OUTFILE</kwd></groupseq>
  <fragref href="#./overlay"/>
  <groupchoice><var>input-filename</var><kwd>*INFILE</kwd></groupchoice>
  <groupchoice><var>output-filename</var><kwd>*OUTFILE</kwd></groupchoice>
  <fragment id="overlay">
    <title>Overlay</title>
    <groupchoice><kwd>*OVERLAP</kwd><kwd>*Prompt</kwd></groupchoice>
  </fragment>
</syntaxdiagram>
```

4.3.11.4 <groupchoice>

The `<groupchoice>` element provides a set of choices between groups of pieces of syntax.

Usage information

Each syntax group is a logical set of pieces of syntax that go together. A group choice specifies that the user must make a choice about which part of the syntax to use. Groups are often nested.

Specialization hierarchy

The `<groupchoice>` element is specialized from `<figgroup>`. It is defined in the syntax-diagram domain module, which is a specialization of the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173).

For this element, the `@importance` attribute indicates whether this item in a syntax diagram is optional, required, or used by default. The attribute value is limited to "optional", "required", "default", or "-dita-use-conref-target".

Example

The following code sample shows how the `<groupchoice>` element can be used to specify that there are two ways to specify an input file name to a command line program:

```
<syntaxdiagram>
  <title>Syntax for runprogram command</title>
  <groupseq>
    <kwd>runprogram</kwd>
    <groupchoice>
      <groupcomp><oper>-</oper><kwd>i</kwd><sep>:</sep><var>program-name.py</var></groupcomp>
      <groupcomp><oper>--</oper><kwd>input</kwd><sep>=</sep><var>program-name.py</var></
groupcomp>
    </groupchoice>
  </groupseq>
</syntaxdiagram>
```

4.3.11.5 <groupcomp>

The `<groupcomp>` element groups a set of pieces of syntax as a single unit.

Usage information

Each syntax group is a logical set of pieces of syntax that go together. The group composite means that the items that make up the syntax diagram will be rendered close together rather than being separated by a horizontal or vertical line, which is the usual formatting method.

Comment by Frank Wegmann on 19 February 2023

The usage information mentions rendering and formatting in one and the same sentence, while this is only about rendering here, if I'm not mistaken regarding our language use. Please advise whether we should break it out in a section Rendering expectations.

Furthermore, I cannot observe this rendering expectation in the current DITA-OT implementation: I get the same output for the code sample below, when e.g. using `<groupseq>` instead of `<groupcomp>`.

Specialization hierarchy

The `<groupcomp>` element is specialized from `<figgroup>`. It is defined in the syntax-diagram domain module, which is a specialization of the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173).

For this element, the `@importance` attribute indicates whether this item in a syntax diagram is optional, required, or used by default. The attribute value is limited to "optional", "required", "default", or "-dita-use-conref-target".

Example

The following code sample shows how the `<groupcomp>` element can be used to indicate how pieces of syntax are grouped together. Two composite groups represent two alternate ways to specify an input file to a command line program, using either `-i:program-name.py` or `--input=program-name.ph`.

```
<syntaxdiagram>
  <title>Syntax for runprogram command</title>
  <groupseq>
    <kwd>runprogram</kwd>
    <groupchoice>
      <groupcomp><oper>-</oper><kwd>i</kwd><sep>:</sep><var>program-name.py</var></groupcomp>
      <groupcomp><oper>--</oper><kwd>input</kwd><sep>=</sep><var>program-name.py</var></
groupcomp>
    </groupchoice>
  </groupseq>
</syntaxdiagram>
```

4.3.11.6 <groupseq>

The `<groupseq>` element specifies the sequence of groups with pieces of syntax.

Usage information

Each syntax group is a logical set of pieces of syntax that go together. Within the syntax definition, groups of keywords, delimiters and other syntax units act as a combined unit, and they occur in a specific sequence, as delimited by the `<groupseq>` element.

Specialization hierarchy

The `<groupseq>` element is specialized from `<figgroup>`. It is defined in the syntax-diagram domain module, which is a specialization of the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173).

For this element, the `@importance` attribute indicates whether this item in a syntax diagram is optional, required, or used by default. The attribute value is limited to "optional", "required", "default", or "-dita-use-conref-target".

Example

The following code sample shows how the `<groupseq>` element can be used to indicate that a short set of command line syntax is specified in a sequential order. The `runprogram` tool name is followed by a choice of how to specify an input file.

```
<syntaxdiagram>
  <title>Syntax for runprogram command</title>
  <groupseq>
    <kwd>runprogram</kwd>
    <groupchoice>
      <groupcomp><oper>-</oper><kwd>i</kwd><sep>:</sep><var>program-name.py</var></groupcomp>
      <groupcomp><oper>--</oper><kwd>input</kwd><sep>=</sep><var>program-name.py</var></
groupcomp>
    </groupchoice>
  </groupseq>
</syntaxdiagram>
```

4.3.11.7 <kwd>

The `<kwd>` element identifies a keyword within a syntax diagram or phrase.

Usage information

A `<kwd>` might be entered by a user typing in the syntax, or rendered by an application as part of a syntax prompt. The keyword value is typed or rendered exactly as specified in the syntax diagram or phrase.

Specialization hierarchy

The `<kwd>` element is specialized from `<keyword>`. It is defined in the syntax-diagram domain module, which is a specialization of the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

For this element, the `@importance` attribute indicates whether this item in a syntax diagram is optional, required, or used by default. The attribute value is limited to "optional", "required", "default", or "-dita-use-conref-target".

Example

The following code sample shows how the `<kwd>` element can be used to identify text that must be provided to the application exactly as specified:

```
<syntaxdiagram id="validate">
  <title>Validate account setup</title>
  <groupseq>
    <kwd>clicmd</kwd>
    <groupcomp>
      <oper>--</oper><kwd>user</kwd><sep>=</sep><var>userid</var>
    </groupcomp>
    <groupcomp>
      <oper>--</oper><kwd>validate</kwd>
    </groupcomp>
  </groupseq>
</syntaxdiagram>
```

4.3.11.8 <oper>

The `<oper>` element identifies an operator within a syntax definition.

Usage information

Typical operators are equals (=), plus (+), or multiply (*).

Specialization hierarchy

The `<oper>` element is specialized from `<ph>`. It is defined in the syntax-diagram domain module, which is a specialization of the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173).

For this element, the `@importance` attribute indicates whether this item in a syntax diagram is optional, required, or used by default. The attribute value is limited to "optional", "required", "default", or "-dita-use-conref-target".

Example

The following code sample shows how the `<oper>` element can be used to specify that the operator in an operation is plus (+):

```
<syntaxdiagram>
  <title>Integer addition</title>
  <groupseq>
    <var>integer</var>
    <oper>+</oper>
    <var>integer</var>
    <delim>=</delim>
    <var>total</var>
  </groupseq>
</syntaxdiagram>
```

4.3.11.9 <repsep>

The `<repsep>` element identifies a character that indicates that a group of syntax elements can (or should) be repeated in a syntax diagram.

Usage information

If the `<repsep>` element contains a separator character such as a plus (+), this indicates that the character must be used between repetitions of the syntax elements.

Specialization hierarchy

The `<repsep>` element is specialized from `<ph>`. It is defined in the syntax-diagram domain module, which is a specialization of the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173).

For this element, the `@importance` attribute indicates whether this item in a syntax diagram is optional or required. The attribute value is limited to "optional", "required", or "-dita-use-conref-target".

Example

In the following code sample, a file listing can be requested for multiple volumes. The `<repsep>` element identifies that each requested volume can be separated with a comma (,):

```
<syntaxdiagram>
  <title>Request file listing</title>
  <groupseq>
    <kwd>clicmd</kwd>
    <groupcomp><oper>--</oper><kwd>user</kwd><sep>=</sep><var>userid</var></groupcomp>
    <groupcomp>
      <repsep>,</repsep>
      <oper>--</oper><kwd>filelist</kwd><sep>=</sep><var>volumeid</var>
    </groupcomp>
  </groupseq>
</syntaxdiagram>
```

4.3.11.10 <sep>

The `<sep>` element defines a character that separates pieces of syntax in a syntax diagram.

Usage information

The separator occurs between keywords, operators, or groups in a syntax definition.

Specialization hierarchy

The `<sep>` element is specialized from `<ph>`. It is defined in the syntax-diagram domain module, which is a specialization of the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173).

For this element, the `@importance` attribute indicates whether this item in a syntax diagram is optional, required, or used by default. The attribute value is limited to "optional", "required", "default", or "-dita-use-conref-target".

Example

The following code sample shows how the `<sep>` element can be used to separate a parameter name from a parameter value:

```
<syntaxdiagram id="validate">
  <title>Validate account setup</title>
  <groupseq>
    <kwd>clicmd</kwd>
    <groupcomp>
      <oper>--</oper>
      <kwd>user</kwd>
      <sep>=</sep>
      <var>userid</var>
    </groupcomp>
    <groupcomp>
      <oper>--</oper>
      <kwd>validate</kwd>
    </groupcomp>
  </groupseq>
</syntaxdiagram>
```

4.3.11.11 <synblk>

A syntax block organizes small pieces of a syntax definition into a larger piece.

Specialization hierarchy

The `<synblk>` element is specialized from `<figgroup>`. It is defined in the syntax-diagram domain module, which is a specialization of the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

The following code sample shows how `<synblk>` elements can be used to group sets of related options for user profile parameters. These syntax blocks might be used in many different sets of syntax.

```
<synblk id="profileopts">
  <title>Required profile options</title>
  <groupcomp><oper>--</oper><kwd>user</kwd><sep>=</sep><var>userid</var></groupcomp>
  <groupcomp><oper>--</oper><kwd>acctkey</kwd><sep>=</sep><var>keyfile</var></groupcomp>
  <groupcomp><oper>--</oper><kwd>region</kwd><sep>=</sep><var>homeregion</var></groupcomp>
</synblk>
```

This block can now be reused in syntax descriptions that always begin with the three profile parameters described in that syntax block:

```
<syntaxdiagram>
  <title>Request file listing</title>
  <groupseq>
    <kwd>clicmd</kwd>
    <synblk conkeyref="syntax-library/profileopts"/>
    <groupcomp><oper>--</oper><kwd>filelist</kwd><sep>=</sep><var>volumeid</var></groupcomp>
  </groupseq>
</syntaxdiagram>
```

4.3.11.12 <synnote>

The syntax note provides additional information within a syntax diagram.

Usage information

The syntax note explains aspects of the syntax that cannot be expressed in the markup itself.

Rendering expectations

The note typically appears at the bottom of the syntax diagram instead of at the bottom of the page.

Specialization hierarchy

The `<synnote>` element is specialized from `<fn>`. It is defined in the syntax-diagram domain module, which is a specialization of the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and the attribute defined below.

@callout
: Specifies the character or character string that is used for the footnote link.

Example

The following code sample show how the `<synnote>` element can be used to remind the reader where to find information for a required parameter:

```
<syntaxdiagram id="validate">
  <title>Validate account setup</title>
  <groupseq>
    <kwd>clicmd</kwd>
    <groupcomp>
      <oper>--</oper><kwd>user</kwd><sep>=</sep><var>userid</var>
      <synnote>Your user ID can be found in your account activation email.</synnote>
    </groupcomp>
    <groupcomp>
      <oper>--</oper><kwd>validate</kwd>
    </groupcomp>
  </groupseq>
</syntaxdiagram>
```

4.3.11.13 <synnoteref>

A syntax note reference is the mechanism for referencing a syntax note within the same syntax diagram.

Usage information

The same notation can be used in more than one syntax definition.

Specialization hierarchy

The `<synnoteref>` element is specialized from `<xref>`. It is defined in the syntax-diagram domain module, which is a specialization of the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@href` (0).

For this element, the `@href` attribute is a reference to a syntax note within the same syntax diagram.

Example

In the following code sample, a syntax note is reused twice in the sample diagram, instructing the reader how to request a modified user name or password:

```
<syntaxdiagram id="validate">
  <title>Validate account setup</title>
  <groupseq>
    <kwd>clicmd</kwd>
    <groupcomp>
      <oper>--</oper><kwd>user</kwd><sep>=</sep><var>account_id</var><synnoteref href="#./
reset"/>
    </groupcomp>
    <groupcomp>
      <oper>--</oper><kwd>pwd</kwd><sep>=</sep><var>password_key</var><synnoteref href="#./
reset"/>
    </groupcomp>
    <groupcomp>
      <oper>--</oper><kwd>validate</kwd>
    </groupcomp>
  </groupseq>
  <synnote id="reset">If you have forgotten your account ID or password key,
    please contact customer support.</synnote>
</syntaxdiagram>
```

4.3.11.14 <synph>

A syntax phrase is a small group of pieces of syntax.

Usage information

The `<synph>` element is used when a complete syntax diagram is not needed, but some of the syntax elements, such as `<kwd>`, `<oper>`, or `<delim>` are used within the text flow of the topic content.

Specialization hierarchy

The `<synph>` element is specialized from `<ph>`. It is defined in the syntax-diagram domain module, which is a specialization of the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173).

Example

The following code sample shows how the `<synph>` element can be used to identify a syntax phrase that must be run before a task:

```
<task id="setup-volume">
  <title>Setting up a new volume</title>
  <shortdesc>This task will help you set up a new volume for your account.</shortdesc>
  <taskbody>
    <prereq>Before starting this procedure, ensure that you have requested
      the volume using the <synph><kwd>request</kwd> <var>volumename</var></synph>
      command.</prereq>
    <!-- ... -->
  </taskbody>
</task>
```

4.3.11.15 <syntaxdiagram>

A syntax diagram represents the syntax of a command, function call, or programming language statement.

Rendering expectations

Traditionally, the syntax diagram is formatted with "railroad tracks" that connect the units of the syntax together, but the presentation might vary depending on the output media.

Specialization hierarchy

The `<syntaxdiagram>` element is specialized from `<fig>`. It is defined in the syntax-diagram domain module, which is a specialization of the programming domain module.

Attributes

The following attributes are available on this element: display attributes (179) and universal attributes (173).

Example

The following code sample shows how a `<syntaxdiagram>`can be used to illustrate the syntax of a basic file-copy command. The initial `COPYF` command is followed by the input directory and file name. The input is followed by a choice of either an output directory (and optional file name) or a file name.

```
<syntaxdiagram>
  <title>CopyFile</title>
  <groupseq><kwd>COPYF</kwd></groupseq>
  <groupcomp><var>input-directory</var><kwd>*INFILE</kwd></groupcomp>
  <groupchoice>
    <groupcomp><var>output-directory</var><kwd importance="optional">*OUTFILE</kwd></
groupcomp>
    <groupcomp><kwd>*OUTFILE</kwd></groupcomp>
  </groupchoice>
</syntaxdiagram>
```

4.3.11.16 <var>

The `<var>` element identifies a variable within a syntax diagram or phrase.

Specialization hierarchy

The `<var>` element is specialized from `<ph>`. It is defined in the syntax-diagram domain module, which is a specialization of the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (173).

For this element, the `@importance` attribute indicates whether this item in a syntax diagram is optional, required, or used by default. The attribute value is limited to "optional", "required", "default", or "-dita-use-conref-target".

Example

The following code sample shows how the `<var>` element can be used to identify variables for which the user will substitute the names of the input and output files:

```
<syntaxdiagram frame="bottom">
 <title>CopyFile</title>
 <groupseq><kwd>COPYF</kwd></groupseq>
 <groupcomp><var>input-filename</var><kwd>*INFILE</kwd></groupcomp>
 <groupseq><var>output-filename</var><kwd>*OUTFILE</kwd></groupseq>
</syntaxdiagram>
```

4.3.12 User interface domain

The user-interface domain elements are used to describe the graphical user interface of a software program.

4.3.12.1 <menucascade>

A menu cascade is a sequence of menu choices in a nested menu, such as **File** > **New**.

Rendering expectations

Processors **SHOULD** separate the contents of the `<uicontrol>` elements in some manner to represent the menu cascade. For example, a visual rendering could separate each UI control with an arrow character.

Specialization hierarchy

The `<menucascade>` element is specialized from `<ph>`. It is defined in the user-interface domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how the `<menucascade>` element can be used to identify a series of menu choices that enable users to switch to dark mode in an editor:

```
<menucascade>
 <uicontrol>View</uicontrol>
 <uicontrol>Editor layout</uicontrol>
 <uicontrol>Display mode</uicontrol>
 <uicontrol>Dark</uicontrol>
</menucascade>
```

4.3.12.2 <screen>

The `<screen>` element contains a textual representation of a terminal console or other text-based computer interface.

Rendering expectations

Processors **SHOULD** preserve the line breaks and spaces that are present in the content of a `<screen>` element.

The contents of the `<screen>` element is typically enclosed within a box to suggest a computer display screen. It also is typically rendered in a monospaced font.

Specialization hierarchy

The `<screen>` element is specialized from `<pre>`. It is defined in the user-interface domain module.

Attributes

The following attributes are available on this element: display attributes (179), universal attributes (173), and `@xml:space` (190).

Example

In the following code sample, the `<screen>` element is used to illustrate the steps needed to clone a git repository and check status:

```
<screen>
workspace $ git clone git@example.com:oasis-tcs/dita-techcomm.git
Cloning into 'dita-techcomm'...
remote: Enumerating objects: 1023, done.
remote: Counting objects: 100% (31/31), done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 1023 (delta 6), reused 21 (delta 4), pack-reused 992
Receiving objects: 100% (1023/1023), 9.87 MiB | 729.00 KiB/s, done.
Resolving deltas: 100% (367/367), done.
workspace $ cd dita-techcomm
dita-techcomm $ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
dita-techcomm $
</screen>
```

4.3.12.3 <shortcut>

A shortcut is a keyboard shortcut that can perform a menu or window action.

Rendering expectations

The contents of the `<shortcut>` element is typically underlined.

Specialization hierarchy

The `<shortcut>` element is specialized from `<keyword>`. It is defined in the user-interface domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0   ).

Example

In the following code sample, the `<shortcut>` element identifies the keyboard shortcuts for navigating a menu to save a file:

```
<menucascade>
 <uicontrol><shortcut>F</shortcut>ile</uicontrol>
```

```
  <uicontrol><shortcut>S</shortcut>ave</uicontrol>
</menucascade>
```

4.3.12.4 <uicontrol>

A user interface control is a label for an item that allows a person or tool to control an interface, such as a button, field, menu item, or other object.

Usage information

The `<uicontrol>` element is also used inside a `<menucascade>` element to identify a sequence of menu choices in a nested menu, such as **File** > **New**.

Specialization hierarchy

The `<uicontrol>` element is specialized from `<ph>`. It is defined in the user-interface domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how the `<uicontrol>` element can be used to identify a button that a user is directed to press:

```
<p>Press <uicontrol>OK</uicontrol> to continue.</p>
```

4.3.12.5 <wintitle>

A window title is the name of a window or dialog.

Specialization hierarchy

The `<wintitle>` element is specialized from `<keyword>`. It is defined in the user-interface domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how the `<wintitle>` element can be used to tag the name of the "Configuration Options" window:

```
<step>
  <cmd>Click <uicontrol>Configure</uicontrol>.</cmd>
  <stepresult>The <wintitle>Configuration Options</wintitle> window
  opens with your last set of selections highlighted.</stepresult>
</step>
```

4.3.13 XML mention domain

The XML-mention domain elements are used to describe and document XML document types and applications. They also can enable typographic styling, search and retrieval, and automatic indexing for XML constructs.

Note Although the original XML 1.0 Recommendation reserved element names beginning with "xml" or "XML" for the use of the XML standard itself, the subsequent introduction of namespaces made the restriction unnecessary. The restriction was formally removed in the XML 1.0 Fifth Edition Specification Errata. The OASIS DITA Technical Committee acknowledges this revised policy in its use of the prefix "xml" for the XML mention domain.

4.3.13.1 <numcharref>

A numeric character reference is a common markup construction that is used in markup languages such as HTML, SGML, and XML. It consists of a short sequence of characters that represents a single character.

Usage information

The content of the `<numcharref>` element should be the numeric value without any leading or trailing characters, for example, "10" or "x0a".

Rendering expectations

The contents of the `<numcharref>` element is typically rendered with a leading ampersand (&) and a trailing semi-colon (;).

Specialization hierarchy

The `<numcharref>` element is specialized from `<markupname>`; the `<numcharref>` element is defined in the XML-mention domain module. The `<markupname>` element is specialized from `<keyword>`, and the `<markupname>` element is defined in the markup-name domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how a `<numcharref>` element can be used to tag the numeric character reference for the a-acute Unicode character:

```
<p>Numeric character references represent characters from the Universal
Character Set (UCS) of Unicode. They are used to reference characters  that
cannot easily be directly encoded in a document, such as a copyright
symbol. When a markup-aware processor encounters a numeric character
reference, for example, <numcharref>225</numcharref>, it renders the
reference as the Unicode character that it represents: lower-case a with acute.</p>
```

4.3.13.2 <parameterentity>

A parameter entity is a syntactic construction that names a collection of elements, attributes, and attribute values. This enables reuse of the collection in grammar files.

Usage information

The content of the `<parameterentity>` element should be the entity name without a leading percentage sign or trailing semi-colon, for example, "keyword.content".

Rendering expectations

The contents of the `<parameterentity>` element is typically rendered with a leading percentage sign (%) and a trailing semi-colon (;).

Specialization hierarchy

The `<parameterentity>` element is specialized from `<markupname>`; the `<parameterentity>` element is defined in the XML-mention domain module. The `<markupname>` element is specialized from `<keyword>`, and the `<markupname>` element is defined in the markup-name domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Examples

The following code sample shows how the `<parameterentity>` element can be used to tag the name of the `%xml-d-dec;` parameter entity:

```
<p>To include the XML-mention domain in a DTD document-type shell, declare and
reference the <parameterentity>xml-d-dec</parameterentity> parameter entity.</p>
```

4.3.13.3 <textentity>

A text entity is an XML construction that resolves to another value when the document is parsed.

Usage information

The content of the `<textentity>` element should be the entity name without the ampersand and semi-colon delimiters, for example, "hi-d-att".

Rendering expectations

The contents of the `<textentity>` element is typically rendered with a leading ampersand (&) and a trailing semi-colon (;).

Specialization hierarchy

The `<textentity>` element is specialized from `<markupname>`; the `<textentity>` element is defined in the XML-mention domain module. The `<markupname>` element is specialized from `<keyword>`, and the `<markupname>` element is defined in the markup-name domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how the `<textentity>` element is used to tag the token that the `@deliveryTarget` attribute domain contributes to the `@specializations` attribute:

```
<p>The <textentity>deliveryTargetAtt-d-att</textentity> entity holds the contribution for the
<xmlatt>specializations</xmlatt> attribute.</p>
```

4.3.13.4 <xmlatt>

An XML attribute is a name and value pair that is associated with an XML element. It defines properties of the XML element.

Usage information

The content of the `<xmlatt>` element should be the attribute name without the at symbol (@), for example, `processing-role`.

Rendering expectations

The contents of the `<xmlatt>` element is typically rendered with a preceding at symbol (@).

Specialization hierarchy

The `<xmlatt>` element is specialized from `<markupname>`; the `<xmlatt>` element is defined in the XML-mention domain module. The `<markupname>` element is specialized from `<keyword>`, and the `<markupname>` element is defined in the markup-name domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how the `<xmlatt>` element can be used to tag mentions of the `@collection-type` and `@linking` attributes:

```
<p>The <xmlatt>collection-type</xmlatt> and <xmlatt>linking</xmlatt>
attributes affect how related links are generated for topics that are
referenced in the DITA map.</p>
```

4.3.13.5 <xmlelement>

An XML element is the basic building block of an XML document. It can contain text, other elements, processing instructions, and more.

Usage information

The content of the `<xmlelement>` element should be the element type name without leading or trailing angle brackets.

Rendering expectations

The contents of the `<xmlelement>` element is typically rendered with leading (<) and trailing (>) angle brackets.

Specialization hierarchy

The `<xmlelement>` element is specialized from `<markupname>`; the `<xmlelement>` element is defined in the XML-mention domain module. The `<markupname>` element is specialized from `<keyword>`, and the `<markupname>` element is defined in the markup-name domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how the `<xmlelement>` element can be used to tag the `<uicontrol>` element from the user-interface domain.

```
<p>Use the <xmlelement>uicontrol</xmlelement> element
to indicate the names of buttons, entry fields, menu items, or
other objects that enable a user to interact with a graphical user
interface.</p>
```

4.3.13.6 <xmlnsname>

The `<xmlnsname>` element identifies mentions of namespace names.

Specialization hierarchy

The `<xmlnsname>` element is specialized from `<markupname>`; the `<xmlnsname>` element is defined in the XML-mention domain module. The `<markupname>` element is specialized from `<keyword>`, and the `<markupname>` element is defined in the markup-name domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how an `<xmlnsname>` element can be used to tag the namespace for SVG:

```
<p>SVG markup is specified in an <xmlelement>svg</xmlelement> element with
the namespace <xmlnamespace>http://www.w3.org/2000/svg</xmlnamespace>.</p>
```

4.3.13.7 <xmlpi>

The `<xmlpi>` element identifies mentions of processing instruction names.

Specialization hierarchy

The `<xmlpi>` element is specialized from `<markupname>`; the `<xmlpi>` element is defined in the XML-mention domain module. The `<markupname>` element is specialized from `<keyword>`, and the `<markupname>` element is defined in the markup-name domain module.

Attributes

The following attributes are available on this element: universal attributes (173) and `@keyref` (0).

Example

The following code sample shows how an `<xmlpi>` element can be used to tag the name of a processing instruction:

```
<p>While DITA does not define any processing instructions, applications might
use some DocBook processing instructions, such as <xmlpi>dbhtml_bgcolor</xmlpi>.</p>
```

5 Conformance

Comment by BobThomas
Conformance statements to technical content will go in this topic.

Appendix A Acknowledgments

Many members of the OASIS DITA Technical Committee participated in the creation of this specification and are gratefully acknowledged.

Robert D. Anderson, Oracle
Deb Bissantz, Vasont Systems
Carsten Brennecke, SAP SE
Bill Burns, HP Inc.
Kristen James Eberlein, Eberlein Consulting LLC
Nancy Harrison, Individual member
Alan Houser, Individual member
Scott Hudson, ServiceNow
Robert Johnson, Jana
Eliot Kimber, ServiceNow
Zöe Lawson, Synopsys, Inc.
Chris Nitchie, Individual member
Christina Rothwell, HPE Inc.
Eric Sirois, IXIASOFT
Keith Schengili-Roberts, Individual member
Tony Self, Individual member
Leroy Steinbacher, Adobe
Dawn Stevens, Comtech Services, Inc.
Amber Swope, Individual member
Bob Thomas, Individual member
Frank Wegmann, Individual member

Appendix B 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.

Rule number	Conformance statement
Rule 1 (23)	When a processor encounters an `<abbreviated-form>` element that references a DITA topic, if the referenced topic is not a `<glossentry>` topic or a specialization of `<glossentry>`, the title of the topic **SHOULD** be rendered.
Rule 2 (23)	When a processor encounters an `<abbreviated-form>` element that references a DITA topic, if the referenced topic is a `<glossentry>` topic or a specialization of `<glossentry>`, processors **SHOULD** render the `<abbreviated-form>` element in the following ways: **First usage** Render the contents of the `<glossSurfaceForm>` elements, if it is not empty. If the `<glossSurfaceForm>` is empty or does not exist, render the contents of the `<glossterm>` element. **Second and later usage** Render the contents of the `<glossAcronym>` elements, if it is not empty. If the `<glossAcronym>` is empty or does not exist, render the contents of the `<glossterm>` element.
Rule 3 (93)	Processors which render the content of `<stepsection>` elements among the `<step>` elements **MUST NOT** number the `<stepsection>` elements.
Rule 4 (114)	In this context, white-space content is considered equivalent to empty content. When the `<equation-number>` element has empty content, the equation number **SHOULD** be generated. When the `<equation-number>` element is not empty, the content **SHOULD** be used as the equation number. Processors **MAY** add punctuation or decoration to the number.
Rule 5 (119)	Processors **SHOULD** process the MathML as though the `<m:math>` element occurs directly in the content of the containing `<mathml>` element.
Rule 6 (121)	Processors **SHOULD** preserve line the breaks and spaces that are present in the content of a `<codeblock>` element.
Rule 7 (132)	Processors **SHOULD** preserve the line breaks and spaces that are present in the content of a `<msgblock>` element.
Rule 8 (136)	Processors **SHOULD** process the SVG as though the `<svg>` element occurs directly in the content of the containing `<svg-container>` element.
Rule 9 (149)	Processors **SHOULD** separate the contents of the `<uicontrol>` elements in some manner to represent the menu cascade. For example, a visual rendering could separate each UI control with an arrow character.
Rule 10 (149)	Processors **SHOULD** preserve the line breaks and spaces that are present in the content of a `<screen>` element.

Appendix C 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.

Appendix C.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 contains 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. This attribute is in the namespace `http://dita.oasis-open.org/architecture/2005/`. This attribute is specified in the topic and map modules, and it uses a default value of the current version of DITA. The current default is "2.0".

@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. This namespace is declared as such in the RNG modules for `<topic>` and `<map>`, but it is specified as an attribute in the equivalent DTD-based modules. The value is fixed to "http://dita.oasis-open.org/architecture/2005/".

Common map attributes

This group contains 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.

@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.

@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.

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

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

@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.

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" ...

@keyscope (common map attributes)

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

See STUB CONTENT (190) 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.

@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".

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 Using the -dita-use-conref-target value 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.

@processing-role (common map attributes)

Specifies whether the referenced resource is processed normally or treated as a resource that is only included in order to resolve references, such as key or content references. The following values are valid:

normal

Indicates that the resource 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.

resource-only

Indicates that the resource should be used only for processing purposes. 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.

-dita-use-conref-target
: See Using the -dita-use-conref-target value 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.

@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.

@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 (190) 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.

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 Using the -dita-use-conref-target value 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 Using the -dita-use-conref-target value 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 Using the -dita-use-conref-target value for more information.

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

Data-element attributes

This group contains 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?

@value (data-element attributes)

Specifies a value associated with the current property or element.

Date attributes

This group contains 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 contains attributes that affect the rendering of many elements.

@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 Using the -dita-use-conref-target value 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 Using the -dita-use-conref-target value 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 (190).

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 contains 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 Using the -dita-use-conref-target value for more information.

See STUB CONTENT (190) 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 (190) 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 (190) 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 (190) 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.

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 ...

@encoding (inclusion attributes)

Comment by Kristen J Eberlein on 29 April 2019

Can we replace "should" in the following definition?

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 contains 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 (190) for detailed information on supported values and processing implications.

@href (link-relationship attributes)
Specifies a reference to a resource. See STUB CONTENT (190) 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 (190) for detailed information on supported values and processing implications.

@type (link-relationship attributes)
Describes the target of a reference. See STUB CONTENT (190) for detailed information on supported values and processing implications.

Localization attributes

Comment by Kristen J Eberlein on 29 September 2022

The definition of the localizations attribute matches how they are described in the architectural topics. Wherever possible, the definition is reused. Where it is not reused (because the definition in the archSpec topics is in a shortdesc), I've checked to ensure that wording is identical.

This group contains the attributes that are related to translation and localization.

@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 Using the -dita-use-conref-target value for more information.

See The dir attribute for more information.

@translate
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.

> **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.

@xml:lang
Specifies the language and optional locale of the content that is contained in an element. Valid values are 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.

> **Comment by Kristen J Eberlein on 29 September 2022**
>
> Do we also want to direct readers to the architectural topics about the `@xml:lang` attribute?

Metadata attributes

This group contains common metadata attributes: `@base`, `@importance`, `@props`, `@rev`, and `@status`. The `@base` and `@props` attributes can be specialized.

@base
Specifies metadata about the element. It is often used as a base for specialized attributes that have a simple syntax for values, but which are not conditional processing 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 conditional processing, although applications might use the value of the `@importance` attribute to highlight elements. For example, in steps of a task topic, the value of the `@importance` attribute indicates whether a step is optional or required.

> **Comment by Kristen J Eberlein on 29 September 2022**
>
> I think the phrase "to highlight elements" is a little off. Maybe "render generated text"? And how about adding "Processors often add text or images to ensure that readers of the generated content understand whether the step is optional or required." to the end of the example?

@props

Specifies metadata about the element. New attributes can be specialized from the `@props` attribute. This attribute supports conditional processing. 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 serving 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.

Comment by Kristen J Eberlein on 29 September 2022

I want to tweak this. How about the following? Also, neither definition describes what values are permitted.

Specifies metadata that identifies when the element was added or the content of the element was modified. The `@rev` attribute can be used for flagging. 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 the modification status of the 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%).

Table accessibility attributes

This group defines a set of attributes that promote table accessibility:

@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.

@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 Using the -dita-use-conref-target value for more information.

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".

Appendix C.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".

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.

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.

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 element. It is often used as a base for specialized attributes that have a simple syntax for values, but which are not conditional processing 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 Using the -dita-use-conref-target value for more information.

See STUB CONTENT (190) 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 (190) 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 (190) 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 (190) 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 Using the -dita-use-conref-target value 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 conditional processing, although applications might use the value of the `@importance` attribute to highlight elements. For example, in steps of a task topic, the value of the `@importance` attribute indicates whether a step is optional or required.

Comment by Kristen J Eberlein on 29 September 2022

I think the phrase "to highlight elements" is a little off. Maybe "render generated text"? And how about adding "Processors often add text or images to ensure that readers of the generated content understand whether the step is optional or required." to the end of the example?

@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".

@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."

@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 element. New attributes can be specialized from the `@props` attribute. This attribute supports conditional processing. 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 serving 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.

> **Comment by Kristen J Eberlein on 29 September 2022**
>
> I want to tweak this. How about the following? Also, neither definition describes what values are permitted.
>
> Specifies metadata that identifies when the element was added or the content of the element was modified. The `@rev` attribute can be used for flagging. 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 the modification status of the element. The following values are valid: "new", "changed", "deleted", "unchanged", and "-dita-use-conref-target".

@translate

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.

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.

@xml:lang

Specifies the language and optional locale of the content that is contained in an element. Valid values are 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.

Comment by Kristen J Eberlein on 29 September 2022

Do we also want to direct readers to the architectural topics about the `@xml:lang` attribute?

Appendix C.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.

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.

Table accessibility attributes

This group contains attributes that are defined on the `<stentry>` element and its specializations: `@headers` (184) and `@scope` (as defined on `<stentry>`) (189).

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`, `@impose-role`, `@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 Using the -dita-use-conref-target value 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.

@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.

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

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

@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.

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" ...

@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. This attribute is in the namespace `http://dita.oasis-open.org/architecture/2005/`. This attribute is specified in the topic and map modules, and it uses a default value of the current version of DITA. The current default is "2.0".

@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 Using the -dita-use-conref-target value 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".

@encoding (inclusion attributes)

Comment by Kristen J Eberlein on 29 April 2019

Can we replace "should" in the following definition?

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 Using the -dita-use-conref-target value 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 (190) 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 Using the -dita-use-conref-target value 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).

@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.

@href (link-relationship attributes)
Specifies a reference to a resource. See STUB CONTENT (190) for detailed information on supported values and processing implications.

@impose-role
Specifies whether this element will impose its role on elements in a referenced map. The attribute is ignored if the target of the reference is not a map or branch of a map. The following values are valid:

keeptarget
The role of the current reference is not imposed on the target of the reference. This is the default for the unspecialized `<topicref>` element and for many convenience elements such as `<keydef>`.

impose
The role of the current reference is imposed on the target of the reference. For example, if a specialized topic reference `<chapter>` uses this value and references a map, a topic reference that resolves in place of the `<chapter>` will be treated as if it were a chapter.

-dita-use-conref-target
See Using the -dita-use-conref-target value for more information.

See STUB CONTENT (190) 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 (190) for information on using this attribute.

For HDITA, the equivalent of `@keyref` is `@data-keyref`

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

@keys
Specifies one or more names for a resource. See STUB CONTENT (190) for information on using this attribute.

For HDITA, the equivalent of `@keys` is `@data-keys`

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

See STUB CONTENT (190) 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.

@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".

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 Using the -dita-use-conref-target value 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.

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

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

@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)

Specifies whether the referenced resource is processed normally or treated as a resource that is only included in order to resolve references, such as key or content references. The following values are valid:

normal
: Indicates that the resource 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.

resource-only
: Indicates that the resource should be used only for processing purposes. 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.

-dita-use-conref-target
: See Using the -dita-use-conref-target value 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.

@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 Using the -dita-use-conref-target value 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 Using the -dita-use-conref-target value 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 (190).

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 (190) for detailed information on supported values and processing implications.

@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 Using the -dita-use-conref-target value for more information.

@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.

@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 (190) 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.

@type (link-relationship attributes)
Describes the target of a reference. See STUB CONTENT (190) 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 Using the -dita-use-conref-target value 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. This namespace is declared as such in the RNG modules for `<topic>` and `<map>`, but it is specified as an attribute in the equivalent DTD-based modules. The value is fixed to "http://dita.oasis-open.org/architecture/2005/".

Appendix C.4 STUB CONTENT

STUB CONTENT
STUB CONTENT

STUB CONTENT
STUB CONTENT

STUB CONTENT
STUB CONTENT

STUB CONTENT
: STUB CONTENT

STUB CONTENT
: STUB CONTENT

STUB CONTENT
: STUB CONTENT

STUB CONTENT
: STUB CONTENT

STUB CONTENT
: STUB CONTENT

STUB CONTENT
: STUB CONTENT

STUB CONTENT
: STUB CONTENT

STUB CONTENT
: STUB CONTENT

STUB CONTENT

STUB CONTENT

STUB CONTENT

STUB CONTENT

STUB CONTENT

STUB CONTENT

STUB CONTENT

STUB CONTENT

STUB CONTENT

STUB CONTENT

STUB CONTENT

STUB CONTENT

STUB CONTENT

STUB CONTENT

STUB CONTENT (192)

STUB CONTENT (192)

```
STUB CONTENT
```

STUB CONTENT

STUB CONTENT

STUB CONTENT

STUB CONTENT

STUB CONTENT

STUB CONTENT

Appendix D Element-by-element recommendations for translators

This topic contains a list of all OASIS DITA elements that are available in the edition. It includes recommendations on whether the element contents are likely to be suitable for translation and whether the element has attributes whose values are likely to be suitable for translation. Examples of content that is not suitable for translation include code fragments and mailing addresses.

Notes on the tables below

- Note that an element might be a block element in one context and an inline element in another. In addition, specialized element types might be rendered in a way that varies from their specialization base. Accordingly, the distinctions presented in the tables are provided only as a guide to known behavior with the base DITA. For element specializations that are not distributed by OASIS, the suggested default is to fall back to the closest ancestor element that is part of the OASIS distribution.
- For all elements, the `@translate` attribute overrides the suggested defaults specified in the tables below.
- Certain block-level elements might appear in the middle of a translation segment. They are considered *subflow* elements in regard to translation. When located in the middle of a translation segment, these element should not be translated as part of that segment. Whenever possible, such elements should be placed only at sentence boundaries in order to aid translation. The subflow elements in base DITA are `<draft-comment>`, `<fn>`, `<idex-see>`, `<index-see-also>`, `<indexterm>`, and `<required-cleanup>`
- The `<keyword>` element (as well as specializations of `<keyword>`) is an inline, phrase-like element when it appears in the body of a document. It can also appear in the `<keywords>` element in `<topicmeta>` (for maps) or in the `<prolog>` (for topic). When it appears in the `<keywords>` element, each `<keyword>` represents an individual segment. In that location, `<keyword>` is considered a subflow element.

Explanation of column headers

The following list explains the headers for the columns:

Element name
The name of the element.

Specialization base
The element from which the current element is specialized. This column only appears in tables for the domain elements.

Same behavior as specialization base?
Indicates whether the element has the same behaviors in regard to translation as its specialization base. The behaviors are whether the element is formatted as a single block or as an inline element, whether the element represents a complete translatable segment, and whether the element contains translatable content. This column only appears in tables for the domain elements.

Block/inline translation
Indicates whether the element represents a complete translatable segment.

Translatable content?
Whether the element contains one or both of the following:

- Text content that can be translated
- Child elements that contain content that can be translated

Notes

This column contains any additional information, including the following items.This column only appears in tables when it is needed.

- Whether the element has any attributes with values that might need translation
- If specializations of the element might need translation,
- If the element is a "subflow" element for the purposes of translation

Bookmap elements

The following table contains information about the bookmap specialization. There are no translatable attributes.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<abbrevlist>`	`<topicref>`	yes	block	yes
`<amendments>`	`<topicref>`	yes	block	yes
`<appendix>`	`<topicref>`	yes	block	yes
`<approved>`	`<data>`	yes	block	no
`<backmatter>`	`<topicref>`	yes	block	yes
`<bibliolist>`	`<topicref>`	yes	block	yes
`<bookabstract>`	`<topicref>`	yes	block	yes
`<bookchangehistory>`	`<data>`	yes	block	no
`<bookevent>`	`<data>`	yes	block	no
`<bookeventtype>`	`<data>`	yes	block	no
`<bookid>`	`<data>`	yes	block	no
`<booklibrary>`	`<ph>`	yes	inline	yes
`<booklist>`	`<topicref>`	yes	block	yes
`<booklists>`	`<topicref>`	yes	block	yes
`<bookmap>`	`<map>`	no	block	yes
`<bookmeta>`	`<topicmeta>`	yes	block	yes
`<booknumber>`	`<data>`	yes	block	no
`<bookowner>`	`<data>`	yes	block	no
`<bookpartno>`	`<data>`	yes	block	no
`<bookrestriction>`	`<data>`	yes	block	no
`<bookrights>`	`<data>`	yes	block	no
`<booktitle>`	`<title>`	yes	block	yes
`<booktitlealt>`	`<ph>`	yes	inline	yes

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<chapter>`	`<topicref>`	yes	block	yes
`<colophon>`	`<topicref>`	yes	block	yes
`<completed>`	`<ph>`	no	inline	no
`<copyrfirst>`	`<data>`	yes	block	no
`<copyrlast>`	`<data>`	yes	block	no
`<day>`	`<ph>`	no	inline	no
`<dedication>`	`<topicref>`	yes	block	yes
`<draftintro>`	`<topicref>`	yes	block	yes
`<edited>`	`<data>`	yes	block	no
`<edition>`	`<data>`	yes	block	no
`<figurelist>`	`<topicref>`	yes	block	yes
`<frontmatter>`	`<topicref>`	yes	block	yes
`<glossarylist>`	`<topicref>`	yes	block	yes
`<indexlist>`	`<topicref>`	yes	block	yes
`<isbn>`	`<data>`	yes	block	no
`<mainbooktitle>`	`<ph>`	yes	inline	yes
`<maintainer>`	`<data>`	yes	block	no
`<month>`	`<ph>`	no	inline	no
`<notices>`	`<topicref>`	yes	block	yes
`<organization>`	`<data>`	yes	block	no
`<part>`	`<topicref>`	yes	block	yes
`<person>`	`<data>`	yes	block	no
`<preface>`	`<topicref>`	yes	block	yes
`<printlocation>`	`<data>`	yes	block	no
`<published>`	`<data>`	yes	block	no
`<publisherinformation>`	`<publisher>`	yes	block	yes
`<publishtype>`	`<data>`	yes	block	no
`<reviewed>`	`<data>`	yes	block	no
`<revisionid>`	`<ph>`	no	inline	no
`<started>`	`<ph>`	no	inline	no
`<summary>`	`<ph>`	yes	inline	yes
`<tablelist>`	`<topicref>`	yes	block	yes
`<tested>`	`<data>`	yes	block	no

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<toc>`	`<topicref>`	yes	block	yes
`<trademarklist>`	`<topicref>`	yes	block	yes
`<volume>`	`<data>`	yes	block	no
`<year>`	`<ph>`	no	inline	no

Concept elements

The following table contains information about the concept specialization. There are no translatable attributes.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<conbody>`	`<body>`	yes	block	yes
`<conbodydiv>`	`<bodydiv>`	yes	block	yes
`<concept>`	`<topic>`	yes	block	yes

Glossary entry elements

The following table contains information about the glossary entry specialization.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?	Notes
`<glossAcronym>`	`<title>`	yes	block	yes	
`<glossAlt>`	`<section>`	yes	block	yes	
`<glossbody>`	`<body>`, `<conbody>`	yes	block	yes	
`<glossdef>`	`<abstract>`	yes	block	yes	
`<glossentry>`	`<topic>`, `<concept>`	yes	block	yes	
`<glossSurfaceForm>`	`<p>`	yes	block	yes	
`<glossSymbol>`	`<image>`	yes	block when `@placement`= break, otherwise inline	yes	
`<glossSynonym>`	`<title>`	yes	block	yes	
`<glossterm>`	`<title>`	yes	block	yes	
`<glossUsage>`	`<note>`	yes	block	yes	`@othertype` can specify translatable content.

Glossary group elements

The following table contains information about the glossary group specialization. There are no translatable attributes.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<glossgroup>`	`<topic>`	yes	block	yes

Reference elements

The following table contains information about the reference specialization. There are no translatable attributes.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<propdesc>`	`<stentry>`	yes	block	yes
`<propdeschd>`	`<stentry>`	yes	block	yes
`<properties>`	`<simpletable>`	yes	block	yes
`<property>`	`<strow>`	yes	block	yes
`<prophead>`	`<sthead>`	yes	block	yes
`<proptype>`	`<stentry>`	yes	block	yes
`<proptypehd>`	`<stentry>`	yes	block	yes
`<propvalue>`	`<stentry>`	yes	block	yes
`<propvaluehd>`	`<stentry>`	yes	block	yes
`<refbody>`	`<body>`	yes	block	yes
`<refbodydiv>`	`<bodydiv>`	yes	block	yes
`<reference>`	`<topic>`	yes	block	yes
`<refsyn>`	`<section>`	yes	block	yes

Task elements

The following table contains information about the task specialization. There are no translatable attributes.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<chdesc>`	`<stentry>`	yes	block	yes
`<chdeschd>`	`<stentry>`	yes	block	yes
`<chhead>`	`<sthead>`	yes	block	yes
`<choice>`	`<li>`	yes	block	yes
`<choices>`	`<ul>`	yes	block	yes

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<choicetable>`	`<simpletable>`	yes	block	yes
`<choption>`	`<stentry>`	yes	block	yes
`<choptionhd>`	`<stentry>`	yes	block	yes
`<chrow>`	`<strow>`	yes	block	yes
`<cmd>`	`<ph>`	no	block	yes
`<context>`	`<section>`	yes	block	yes
`<info>`	`<div>`	no	block	yes
`<postreq>`	`<section>`	yes	block	yes
`<prereq>`	`<section>`	yes	block	yes
`<result>`	`<section>`	yes	block	yes
`<step>`	`<li>`	yes	block	yes
`<stepresult>`	`<div>`	no	block	yes
`<steps>`	`<ol>`	yes	block	yes
`<steps-informal>`	`<section>`	yes	block	yes
`<steps-unordered>`	`<ul>`	yes	block	yes
`<stepsection>`	`<li>`	yes	block	yes
`<steptroubleshooting>`	`<div>`	yes	block	yes
`<stepxmp>`	`<div>`	no	block	yes
`<task>`	`<topic>`	yes	block	yes
`<taskbody>`	`<body>`	yes	block	yes
`<tasktroubleshooting>`	`<section>`	yes	block	yes
`<tutorialinfo>`	`<div>`	no	block	yes

Troubleshooting elements

The following table contains information about the troubleshooting specialization. There are no translatable attributes.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<cause>`	`<section>`	yes	block	yes
`<condition>`	`<section>`	yes	block	yes
`<remedy>`	`<section>`	yes	block	yes
`<responsibleParty>`	`<p>`	yes	block	yes
`<troublebody>`	`<body>`	yes	block	yes
`<troubleshooting>`	`<topic>`	yes	block	yes

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<troubleSolution>`	`<bodydiv>`	yes	block	yes

Abbreviated form domain (abbrev-d)

The following table contains information about the abbreviated form domain. There are no translatable attributes in this domain.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<abbreviated-form>`	`<term>`	yes	n/a (empty element)	n/a (empty element)

Equation domain (equation-d)

The following table contains information about the equation domain. There are no translatable attributes in this domain.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<equation-inline>`	`<ph>`	yes	inline	yes
`<equation-block>`	`<div>`	yes	block	yes
`<equation-number>`	`<ph>`	yes	inline	yes
`<equation-figure>`	`<fig>`	yes	block	yes

Glossary reference domain (glossref-d)

The following table contains information about the glossary reference domain. There are no translatable attributes in this domain.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<glossref>`	`<topicref>`	yes	block	yes

Hardware domain (hw-d)

The following table contains information about the hardware domain. There are no translatable attributes in this domain.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<hwcontrol>`	`<ph>`	yes	inline	yes
`<partno>`	`<ph>`	yes	inline	yes

Markup domain (markup-d)

The following table contains information about the markup domain. There are no translatable attributes in this domain.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<markupname>`	`<keyword>`	no	inline	no

MathML domain (mathml-d)

The following table contains information about the MathML domain. There are no translatable attributes in this domain.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<mathml>`	`<foreign>`	yes	Depends on math content	Follow rules established for MathML standard
`<mathmlref>`	`<include>`	yes	n/a (empty element)	n/a (empty element)

Programming domain (pr-d)

The following table contains information about the programming domain. There are no translatable attributes in this domain.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<apiname>`	`<keyword>`	yes	inline	yes
`<codeblock>`	`<pre>`	yes	block	yes
`<codeph>`	`<ph>`	yes	inline	yes
`<coderef>`	`<include>`	yes	n/a (empty element)	n/a (empty element)
`<option>`	`<keyword>`	yes	inline	yes
`<parml>`	`<dl>`	yes	block	yes
`<parmname>`	`<keyword>`	yes	inline	yes
`<pd>`	`<dd>`	yes	block	yes
`<plentry>`	`<dlentry>`	yes	block	yes
`<pt>`	`<dt>`	yes	block	yes

Release management domain (relmgmt-d)

The following table contains information about the release management domain. There are no translatable attributes in this domain.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<change-historylist>`	`<metadata>`	no	n/a	no
`<change-completed>`	`<data>`	yes	n/a	no
`<change-item>`	`<data>`	yes	n/a	no
`<change-person>`	`<data>`	yes	n/a	no
`<change-organization>`	`<data>`	yes	n/a	no
`<change-revisionid>`	`<data>`	yes	n/a	no
`<change-request-reference>`	`<data>`	yes	n/a	no
`<change-request-system>`	`<data>`	yes	n/a	no
`<change-request-id>`	`<data>`	yes	n/a	no
`<change-started>`	`<data>`	yes	n/a	no
`<change-summary>`	`<data>`	yes	n/a	no

Software domain (sw-d)

The following table contains information about the software domain. There are no translatable attributes in this domain.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<cmdname>`	`<keyword>`	yes	inline	yes
`<filepath>`	`<ph>`	yes	inline	yes
`<msgblock>`	`<pre>`	yes	block	yes
`<msgnum>`	`<keyword>`	yes	inline	yes
`<msgph>`	`<ph>`	yes	inline	yes
`<systemoutput>`	`<ph>`	yes	inline	yes
`<userinput>`	`<ph>`	yes	inline	yes
`<varname>`	`<keyword>`	yes	inline	yes

SVG domain (svg-d)

The following table contains information about the SVG domain. There are no translatable attributes in this domain.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<svg-container>`	`<foreign>`	yes	Depends on SVG content	Follow rules established for SVG standard
`<svgref>`	`<include>`	yes	n/a (empty element)	n/a (empty element)

Syntax diagram domain (syntaxdiagram-d)

The following table contains information about the syntax diagram domain. There are no translatable attributes in this domain.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<delim>`	`<ph>`	yes	inline	yes
`<fragment>`	`<figgroup>`	yes	block	yes
`<fragref>`	`<xref>`	yes	inline	yes
`<groupchoice>`	`<figgroup>`	yes	block	yes
`<groupcomp>`	`<figgroup>`	yes	block	yes
`<groupseq>`	`<figgroup>`	yes	block	yes
`<kwd>`	`<keyword>`	yes	inline	yes
`<oper>`	`<ph>`	yes	inline	yes
`<repsep>`	`<ph>`	yes	inline	yes
`<sep>`	`<ph>`	yes	inline	yes
`<synblk>`	`<figgroup>`	yes	block	yes
`<synnote>`	`<fn>`	yes	block	yes
`<synnoteref>`	`<xref>`	yes	inline	yes
`<synph>`	`<ph>`	yes	inline	yes
`<syntaxdiagram>`	`<fig>`	yes	block	yes
`<var>`	`<ph>`	yes	inline	yes

User interface domain (ui-d)

The following table contains information about the user interface domain. There are no translatable attributes in this domain.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<menucascade>`	`<ph>`	yes	inline	yes
`<screen>`	`<pre>`	yes	block	yes
`<shortcut>`	`<keyword>`	yes	inline	yes
`<uicontrol>`	`<ph>`	yes	inline	yes

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<wintitle>`	`<keyword>`	yes	inline	yes

XML mention domain (xml-d)

The following table contains information about the XML mention domain. There are no translatable attributes in this domain.

Element name	Specialization base	Same behavior as specialization base?	Block/inline (translation)	Translatable content?
`<numcharref>`	`<keyword>`, `<markupname>`	no	inline	no
`<parameterentity>`	`<keyword>`, `<markupname>`	no	inline	no
`<textentity>`	`<keyword>`, `<markupname>`	no	inline	no
`<xmlatt>`	`<keyword>`, `<markupname>`	no	inline	no
`<xmlelement>`	`<keyword>`, `<markupname>`	no	inline	no
`<xmlnsname>`	`<keyword>`, `<markupname>`	no	inline	no
`<xmlpi>`	`<keyword>`, `<markupname>`	no	inline	no

Appendix E Revision history

The following table contains information about revisions to this document.

Revision	Date	Editor	Description of changes
01	19 May 2019	Kristen James Eberlein	Generated working draft #01. Contains initial TOC for consideration.
02	24 May 2019	Kristen James Eberlein	Generated working draft #02. Contains reworked task element topics.
03	27 August 2017	Kristen James Eberlein	Generated working draft #03. Contains updated DITA source for the following proposals: • #85: Several glossentry elements should allow `<sub>` and `<sup>` • #106 Allow steps to nest Contains edited element-reference topics for the following domains: • Markup • Programming (partial) • Software • User interface • XML mention
06	17 October 2022	Kristen James Eberlein	Contains content for review A: Task elements
07	08 November 2022	Kristen James Eberlein	Contains content revised as a result of review A comments, plus content prepped for review B
08	09 January 2023	Kristen James Eberlein	Content prepped for review B.
09	24 January 2023	Kristen James Eberlein	Contains content revised as result of review C comments, plus content prepped for review D.
10	06 February 2023	Kristen James Eberlein	Contains content prepped for review E.
11	20 February 2023	Kristen James Eberlein	Contains the following content: • Content revised as a result of review E.1 • Updated navigation structure concerning placement of `<abbreviated-form>`, `<glossgroup>`, and `<glossref>` element-reference topics
12	28 February 2023	Kristen James Eberlein	Contains the following content:

Revision	Date	Editor	Description of changes
			• Content revised as a result of review E.2 • Content prepped for review F: "Syntax diagram domain"
13	07 March 2023	Kristen James Eberlein	Contains content prepped for review G: "xNAL domain"
14	21 March 2023	Kristen James Eberlein	Contains content prepped for review H: "Release management domain"
15	03 April 2023	Kristen James Eberlein	Contains content prepped for review I: "Equation, MathML, and SVG domains"
16	01 November 2023	Kristen James Eberlein	Contains the following content • Content revised as part of review K • Draft comments from Dawn Stevens in glossary content
18	26 March 2024	Kristen James Eberlein	Contains content updated as part of review of "Glossary entry elements and `<abbreviated-form>`
19	14 May 2024	Kristen James Eberlein	Contains content updated for the 2nd review of "Glossentry elements"