Darwin Information Typing Architecture (DITA)
Version 1.3

OASIS Standard
TBD

 Specification URLs:

This Version:
http://docs.oasis-open.org/dita/v1.3/os/spec/DITA1.3-spec.html (Authoritative version)
http://docs.oasis-open.org/dita/v1.3/os/spec/DITA1.3-spec.pdf
http://docs.oasis-open.org/dita/v1.3/os/spec/DITA1.3-spec-chm.zip
http://docs.oasis-open.org/dita/v1.3/os/spec/DITA1.3-spec.zip

Previous Version:
http://docs.oasis-open.org/dita/v1.3/cs01/spec/DITA1.3-spec.html (Authoritative version)
http://docs.oasis-open.org/dita/v1.3/cs01/spec/DITA1.3-spec.pdf
http://docs.oasis-open.org/dita/v1.3/cs01/spec/DITA1.3-spec-chm.zip
http://docs.oasis-open.org/dita/v1.3/cs01/spec/DITA1.3-spec.zip

Latest Version:
http://docs.oasis-open.org/dita/v1.3/spec/DITA1.3-spec.html (Authoritative version)
http://docs.oasis-open.org/dita/v1.3/spec/DITA1.3-spec.pdf
http://docs.oasis-open.org/dita/v1.3/spec/DITA1.3-spec-chm.zip
http://docs.oasis-open.org/dita/v1.3/spec/DITA1.3-spec.zip

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

Chair:
Kristen James Eberlein, Individual Member

Editors:
Kristen James Eberlein, Individual Member
Robert D. Anderson, IBM

Related Work:
This specification replaces or supercedes:

OASIS DITA Version 1.1 Architectural Specification
OASIS DITA Version 1.1 Language Specification

This specification is related to:

OASIS DITA Version 1.3 DTDs (http://docs.oasis-open.org/dita/v1.3/os/DITA1.3-dtds.zip)
Declared XML Namespace:
None.

Abstract:
The Darwin Information Typing Architecture (DITA) 1.3 specification defines both a) a set of document types for authoring and organizing topic-oriented information; and b) a set of mechanisms for combining, extending, and constraining document types.

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 Version" or "Latest Approved Version" location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee’s web page at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dita.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/dita/ipr.php).

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dita.
# Table of Contents

## 1 Introduction to DITA 1.3

1.1 About the DITA specification ................................................................. 8
1.2 Terminology .......................................................................................... 8
1.3 Normative references ............................................................................ 9
1.4 Non-normative references .................................................................... 10
1.5 About the specification source ............................................................. 12
1.6 Formatting conventions in the XHTML version of the specification ...... 12

## 2 Architectural specification: Learning and training

2.1 Introduction to DITA ............................................................................ 15
   2.1.1 DITA terminology and notation .................................................... 15
   2.1.2 Basic concepts ............................................................................ 20
   2.1.3 File naming conventions ............................................................... 21
   2.1.4 Producing different deliverables from a single source .................. 23
2.2 DITA markup ....................................................................................... 23
   2.2.1 DITA topics ............................................................................... 24
   2.2.2 DITA maps ............................................................................... 30
   2.2.3 DITA metadata ......................................................................... 44
2.3 DITA processing .................................................................................. 56
   2.3.1 Module compatibility and the @domains attribute ...................... 57
   2.3.2 Navigation ............................................................................... 57
   2.3.3 DITA linking ............................................................................ 58
   2.3.4 Direct DITA addressing ............................................................... 63
   2.3.5 Indirect key-based addressing .................................................... 66
   2.3.6 Content inclusion (conref) .......................................................... 84
   2.3.7 Conditional processing (profiling) ............................................... 88
   2.3.8 Branch filtering ....................................................................... 95
   2.3.9 Chunking ............................................................................... 102
   2.3.10 Translation and localization ...................................................... 107
   2.3.11 Context hooks for help systems and other user assistance .......... 110
   2.3.12 Sorting ............................................................................... 111
2.4 Configuration, specialization, and constraints ...................................... 112
   2.4.1 Overview of DITA extension facilities .......................................... 112
   2.4.2 Configuration (Document type shells) ......................................... 118
   2.4.3 Specialization ........................................................................... 141
   2.4.4 Constraints ............................................................................. 178
2.5 Technical content specializations ......................................................... 192
   2.5.1 Overview of the DITA 1.3 Specification: Technical Content .......... 192
   2.5.2 Technical content: Document types ........................................... 193
3 Language reference: Learning and training ............................................................ 223

3.1 Element quick reference ........................................................................................... 223
3.1.1 Base DITA elements, A to Z ................................................................................. 223
3.1.2 Technical content elements, A to Z ....................................................................... 226
3.1.3 Learning and training elements, A to Z ................................................................. 232
3.1.4 All DITA elements, A to Z ..................................................................................... 235

3.2 Topic elements ............................................................................................................. 247
3.2.1 Basic topic elements .............................................................................................. 247
3.2.2 Body elements ........................................................................................................ 258
3.2.3 Table elements ....................................................................................................... 287
3.2.4 Related links elements ........................................................................................... 297

3.3 Map elements .............................................................................................................. 302
3.3.1 Basic map elements .............................................................................................. 302
3.3.2 Map group elements .............................................................................................. 313

3.4 Metadata elements ..................................................................................................... 320
3.4.1 Prolog (metadata) elements .................................................................................... 320
3.4.2 Indexing group elements ........................................................................................ 336
3.4.3 Delayed conref resolution elements ....................................................................... 343

3.5 Domain elements ........................................................................................................ 352
3.5.1 Hazard statement elements .................................................................................... 352
3.5.2 Highlighting elements ........................................................................................... 357
3.5.3 Utilities elements ................................................................................................... 360
3.5.4 DITAVAL reference domain .................................................................................. 366

3.6 Classification elements ............................................................................................... 371
3.6.1 Subject scheme maps ............................................................................................. 371
3.6.2 Classification domain elements .............................................................................. 395

3.7 Specialization elements ............................................................................................... 406
3.7.1 <boolean> ............................................................................................................. 406
3.7.2 <data> ....................................................................................................................... 407
3.7.3 <data-about> .......................................................................................................... 409
3.7.4 <foreign> ................................................................................................................... 410
3.7.5 <index-base> ............................................................................................................. 411
3.7.6 <itemgroup> ............................................................................................................. 411
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.7.7</td>
<td>&lt;no-topic-nesting&gt;</td>
<td>411</td>
</tr>
<tr>
<td>3.7.8</td>
<td>&lt;state&gt;</td>
<td>412</td>
</tr>
<tr>
<td>3.7.9</td>
<td>&lt;unknown&gt;</td>
<td>412</td>
</tr>
<tr>
<td>3.8</td>
<td>Legacy conversion elements</td>
<td>413</td>
</tr>
<tr>
<td>3.8.1</td>
<td>&lt;required-cleanup&gt;</td>
<td>413</td>
</tr>
<tr>
<td>3.9</td>
<td>DITAVAL elements</td>
<td>414</td>
</tr>
<tr>
<td>3.9.1</td>
<td>&lt;val&gt;</td>
<td>414</td>
</tr>
<tr>
<td>3.9.2</td>
<td>&lt;style-conflict&gt;</td>
<td>416</td>
</tr>
<tr>
<td>3.9.3</td>
<td>&lt;prop&gt;</td>
<td>417</td>
</tr>
<tr>
<td>3.9.4</td>
<td>&lt;revprop&gt;</td>
<td>418</td>
</tr>
<tr>
<td>3.9.5</td>
<td>&lt;startflag&gt;</td>
<td>420</td>
</tr>
<tr>
<td>3.9.6</td>
<td>&lt;endflag&gt;</td>
<td>420</td>
</tr>
<tr>
<td>3.9.7</td>
<td>&lt;alt-text&gt;</td>
<td>421</td>
</tr>
<tr>
<td>3.10</td>
<td>Technical content elements</td>
<td>421</td>
</tr>
<tr>
<td>3.10.1</td>
<td>Concept elements</td>
<td>421</td>
</tr>
<tr>
<td>3.10.2</td>
<td>Task elements</td>
<td>423</td>
</tr>
<tr>
<td>3.10.3</td>
<td>Reference elements</td>
<td>438</td>
</tr>
<tr>
<td>3.10.4</td>
<td>Troubleshooting elements</td>
<td>444</td>
</tr>
<tr>
<td>3.10.5</td>
<td>Glossary elements</td>
<td>447</td>
</tr>
<tr>
<td>3.10.6</td>
<td>Bookmap elements</td>
<td>460</td>
</tr>
<tr>
<td>3.10.7</td>
<td>Technical-content domains elements</td>
<td>488</td>
</tr>
<tr>
<td>3.11</td>
<td>Learning and training elements</td>
<td>549</td>
</tr>
<tr>
<td>3.11.1</td>
<td>Learning and training topic elements</td>
<td>549</td>
</tr>
<tr>
<td>3.11.2</td>
<td>Learning and training map types</td>
<td>557</td>
</tr>
<tr>
<td>3.11.3</td>
<td>Learning and training map domain elements</td>
<td>559</td>
</tr>
<tr>
<td>3.11.4</td>
<td>Learning2 domain</td>
<td>569</td>
</tr>
<tr>
<td>3.11.5</td>
<td>Learning and training interactions domain elements</td>
<td>589</td>
</tr>
<tr>
<td>3.11.6</td>
<td>Learning and training metadata elements</td>
<td>595</td>
</tr>
<tr>
<td>3.11.7</td>
<td>Other learning and training elements</td>
<td>605</td>
</tr>
<tr>
<td>3.12</td>
<td>Attributes</td>
<td>650</td>
</tr>
<tr>
<td>3.12.1</td>
<td>Universal attribute group</td>
<td>650</td>
</tr>
<tr>
<td>3.12.2</td>
<td>Architectural attribute group</td>
<td>654</td>
</tr>
<tr>
<td>3.12.3</td>
<td>Attributes common to many map elements</td>
<td>655</td>
</tr>
<tr>
<td>3.12.4</td>
<td>Complex-table attribute group</td>
<td>657</td>
</tr>
<tr>
<td>3.12.5</td>
<td>Data element attributes group</td>
<td>659</td>
</tr>
<tr>
<td>3.12.6</td>
<td>Date attributes group</td>
<td>659</td>
</tr>
<tr>
<td>3.12.7</td>
<td>Display attribute group</td>
<td>660</td>
</tr>
<tr>
<td>3.12.8</td>
<td>Link relationship attribute group</td>
<td>661</td>
</tr>
<tr>
<td>3.12.9</td>
<td>Other common attributes</td>
<td>661</td>
</tr>
<tr>
<td>3.12.10</td>
<td>Simpletable attribute group</td>
<td>662</td>
</tr>
<tr>
<td>3.12.11</td>
<td>Specialization attributes group</td>
<td>663</td>
</tr>
<tr>
<td>3.12.12</td>
<td>Topicref element attributes group</td>
<td>663</td>
</tr>
<tr>
<td>3.12.13</td>
<td>Complex attribute definitions</td>
<td>663</td>
</tr>
</tbody>
</table>
4 Conformance ................................................................. 679

5 Acknowledgments .......................................................... 682

6 Non-normative information ............................................. 683
  6.1 Changes from previous versions .................................... 683
  6.1.1 Changes from DITA 1.2 to DITA 1.3 .......................... 683
  6.1.2 Changes from DITA 1.1 to DITA 1.2 ......................... 685
  6.1.3 Changes from DITA 1.0 to DITA 1.1 ......................... 686
  6.2 Migrating from DITA 1.1 to 1.2 .................................... 687
  6.3 DITA MIME type ....................................................... 689
  6.4 Considerations for generalizing <foreign> elements ......... 689
  6.5 Summary of addressing elements ............................... 690
  6.6 Element-by-element recommendations for translators .... 693
  6.7 Implementation reference for DITA modules and document-type shells .................................. 719
    6.7.1 Domain modules .................................................. 719
    6.7.2 XML schema organization ..................................... 721
  6.8 Processing component modularization and naming conventions ............................................. 728
  6.9 Processing interoperability considerations .................... 728
  6.10 Specialization design, customization, and the limits of specialization ...................................... 730
2 Introduction to DITA 1.3

The Darwin Information Typing Architecture (DITA) specification defines a set of document types for authoring and organizing topic-oriented information, as well as a set of mechanisms for combining, extending, and constraining document types.

2.1 About the DITA specification

The DITA specification consists of a written specification and XML grammar files.

The DITA specification consists of the following components:

XML grammar files

The DITA markup for DITA vocabulary modules and DITA document types is available in several XML languages: RELAX NG, XML Document-Type Definitions (DTDs), and W3C XML Schema.

While the files should define the same DITA elements, the RELAX NG grammars are normative if there is a discrepancy. If there is a discrepancy between the written specification (this document) and the RELAX NG grammars, the written specification takes precedence.

DITA written specification

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

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

The DITA written specification is available in the following formats; the HTML version is authoritative:

- XHTML (available from the OASIS Web site)
- CHM
- PDF
- DITA source
- ZIP of XHTML (optimized for local use)

2.2 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 (http://www.ietf.org/rfc/rfc2119.txt).

MUST

This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.

MUST NOT

This phrase, or the phrase "SHALL NOT", means that the definition is an absolute prohibition of the specification.

SHOULD
This word, or the adjective "RECOMMENDED", means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

SHOULD NOT
This phrase, or the phrase "NOT RECOMMENDED", means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

MAY
This word, or the adjective "OPTIONAL", means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option must be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option must be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides).

2.3 Normative references

Normative references are references to external documents or resources to which implementers of DITA MUST comply.

[MathML 3.0]

[Namespaces in XML 1.0]

[Namespaces in XML 1.1]

[RELAX NG]

[RELAX NG Compact Syntax]

[RELAX NG DTD Compatibility]

[RFC 2119]

[RFC 3986]

[SVG 1.1]
2.4 Non-normative references

Non-normative references are references to external documents or resources that implementers of DITA might find useful.

**Comment by Kristen James Eberlein, 3 June 2010**
Contents of this topic based on suggestions made on the e-mail list by Eliot Kimber, Seth Park, and Bruce Nevin. This topic must be reviewed. Here also are suggestions that I did not include:

- IMS-QTI (Kimber)
- Information mapping (Nevin)
- SGML (Nevin) -- no ISO spec available without charge
- Standards reflected in Machine Industry specializations (Kimber)
- XNAL (Kimber)

**Comment by Kristen Eberlein, 15 February 2014**
From proposal 13112:

[DITA RNG2DTDXSD]
DITA Editors. Utilities for generating conforming DITA DTD and XSD document type shells and modules from DITA RELAX NG grammars. {Location to be determined, but likely a GitHub repository}.

[SCHEMATRON]

**Comment by Kristen Eberlein, 7 April 2014**
From review #1:
Eliot Kimber: "ISO 8601 is referenced from change-started-date and change-completed-date. It's not available online but we should still have a citation of the standard."

Kris Eberlein: "It's available, but only if you purchase it. In fact, I cannot even get the citation information without purchasing it. I've sent a note to customerservice@iso.org."

**[CSS 2.1]**

**[XHTML 1.0]**


**[XHTML 1.1]**


**[XPointer 1.0]**


**[XLIFF 1.2]**


**[XML Catalogs 1.1]**


**[xml:tm 1.0]**


**[XQuery 1.0]**


**[XSL 1.0]**


**[XSL 1.1]**


**[XSLT 1.0]**


**[XSLT 2.0]**


**[XTM 1.0]**

2.5 About the specification source

The DITA specification is authored as a DITA content set and published using the DITA Open Toolkit. It is an example of a complex document that is developed by a team of people and uses many DITA features, including key references (keyrefs) and content references (conrefs).

Comment by Kristen Eberlein, 5 March 2014
Jarno Elovirta made the following comment in the first review:

This is a colophon topic and is not introduction to DITA. This should be moved to "Non-normative information" and possibly renamed "Colophon".

As a result, I moved the topic to its current location in the "Introduction to DITA 1.3" section. Would it be better located in the appendix?

As a convenience for readers, the DITA specification is published in several packages which contain different combinations of required and optional modules. The ability to easily define, manage, and publish these packages is a result of using DITA for the specification source. The source files for the DITA specification are managed in a version control repository that is maintained by OASIS; they also can be downloaded from OASIS.

The specification editors used the following applications to work with the DITA source:

- <oXygen/> XML Editor
- DITAweb
- Antenna House XML Formatter

2.6 Formatting conventions in the XHTML version of the specification

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

Link previews

The DITA specification uses the content of the unique DITA `<shortdesc>` element to provide link previews for its readers. These link previews are visually highlighted by a border and a colored background. The link previews are not normative; they contain the content of the `<shortdesc>` element for the child topic, which is normatively rendered as the first paragraph of the child topic; the content is identical in both renditions. 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 XHTML version of the specification:
To ease readers in navigating from one topic to another, each XHTML file generated by a DITA topic contains the following navigation links at the bottom:

**Parent topic**
- Takes readers to the parent topic, which 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 XHTML version of the specification:

**Figure 2: Navigation links**
When readers hover over the navigation links, the short description of the DITA topic also is displayed.
3 Architectural specification: Learning and training

DITA 1.2 introduces the Learning and Training specialization, which is designed for developing instructional materials. It uses all of base DITA and some (but not all) parts of the Technical Content package.

Comment by Kristen Eberlein
I'm confused as to why Learning and Training is the overarching title at this point. What follows is an introduction to DITA in general, and not just Learning and Training specialization. rhamilton updated comment 14/3/2014 00:50:35 1

If you're crazy, then so am I. I see a similar effect on the LangRef section -- it has "Learning and training" in the title and there's verbiage there about L&T, not about the DITA lang ref. Is there a high-level problem with the publication at large, or with DITAweb's presentation of it? I don't see anything obviously awry in the DITA maps in SVN. tmagliery new comment 14/3/2014 23:35:16 1

Good point. This is an artifact of early packaging prototyping that keberlein new comment 24/3/2014 22:42:17 1

3.1 Introduction to DITA

The Darwin Information Typing Architecture (DITA) is an XML-based architecture for authoring, producing, and delivering topic-oriented, information-typed content that can be reused and single-sourced in a variety of ways. While DITA historically has been driven by the requirements of large-scale technical documentation authoring, management, and delivery, it is a standard that is applicable to any kind of publication or information that might be presented to readers, including interactive training and educational materials, standards, reports, business documents, trade books, travel and nature guides, and more.

DITA is designed for creating new document types and describing new information domains based on existing types and domains. The process for creating new types and domains is called specialization. Specialization enables the creation of very specific, targeted document-type definitions that still can share the common output transformations and design rules developed for more general types and domains; this is similar to how classes in an object-oriented system can inherit the methods of ancestor classes.

Because DITA topics are XML conforming, they can be readily viewed, edited, and validated using standard XML tools, although realizing the full potential of DITA requires using DITA-aware tools.

3.1.1 DITA terminology and notation

The DITA specification uses specific notation and terms to define the components of the DITA standard.

Notation

Comment by Kristen Eberlein , 24 March 2014
Stan Doherty made the following comment in review #1 concerning the "Cascading of attributes and metadata in a DITA map" topic; do we need to add the term cascading to our terminology that is explicated in this topic?

The explicit distinction here between containment hierarchy (mentioned several times elsewhere in the spec) and element type hierarchy is very nice . . . might be worth promoting to our list of key concepts.

The following conventions are used throughout the specification:

attribute types
Attribute names might be preceded by @ to distinguish them from elements or surrounding text, for example, the @props or the @class attribute.

element types
Element names might be delimited with angle brackets (< and >) to distinguish them from surrounding text, for example, the <keyword> and the <prolog> element.

In general, the unqualified use of the term map or topic can be interpreted to mean "a <map> element and any specialization of a <map> element " or "a <topic> element or any specialization of a <topic> element."

**Normative and non-normative information**

The DITA specification contains normative and non-normative information:

**Normative information**

Normative information is the formal portion of the specification that describes the rules and requirements that make up the DITA standard and which must be followed.

**Non-normative information**

Non-normative information includes descriptions that provide background, examples, and other useful information that are not formal requirements or rules that must be followed. The terms non-normative and informative are used interchangeably.

All information in the specification should be considered normative unless it is an example, an appendix, or is explicitly labeled as informative or non-normative. Appendices are always non-normative, unless explicitly stated otherwise. The DITA specification contains examples to help clarify or illustrate specific aspects of the specification. Because examples are specific rather than general, they may not illustrate all aspects or be the only way to accomplish or implement an aspect of the specification. Therefore all examples are non-normative, unless explicitly stated otherwise.

**Basic DITA terminology**

The following terminology is used to discuss basic DITA concepts:

**DITA attribute type**

An attribute type that is one of the following:

- One of the base attribute types that are defined by the DITA specification
- A specialization of either the @base or @props attribute

**DITA document**

An XML document that conforms to the requirements of this specification. A DITA document **MUST** have as its root element one of the following elements:

- <map> or a specialization of the <map> element
- <topic> or a specialization of the <topic> element
- <dita>, which cannot be specialized, but which allows documents with multiple sibling topics

**DITA document type**

A unique set of structural modules, domain modules, and constraint modules that taken together provide the XML element and attribute declarations that define the structure of DITA documents. DITA document types normally are implemented using DITA document-type shells.

---

**Comment by Kristen Eberlein, 27 January 2014**

We might need to edit the last sentence if we move to using RNG.

**DITA document-type shell**

A set of DTD, XSD, or RELAX NG declarations that implement a DITA document type by using the rules and design patterns that are included in the DITA specification. A DITA document-type shell includes and configures one or more structural modules, zero or more domain modules, and zero or more constraint modules. With the exception of the optional declarations for the <dita> element and its attributes, DITA document-type shells do not declare any element or attribute types directly.

**DITA element**

An XML element instance whose type is a DITA element type. DITA elements must exhibit a @class attribute that has a value that conforms to the rules for specialization hierarchy specifications.
DITA element type
An element type that is one of the following:

• One of the base element types that are defined by the DITA specification
• A specialization of one of the base element types

A DITA element type is declared in exactly one vocabulary module. DITA element types may only exhibit attributes that are DITA attribute types.

map branch
A map branch consists of an element in a map, along with any child elements and all resources that are referenced by the original element or its children.

Comment by robander, 21 April 2014
Wondering if we should add "Also referred to simply as 'branch' when used in a map context", or perhaps change the term to "map branch or branch" - I do like having it sorted with the other map terms, and using "map branch" does that while keeping our current alphabetical order.

map instance
An occurrence of a map type in a document.

map type
An element type that defines a set of relationships among topic instances. The map type provides the root element and, through the contained element types, the substructure for the map instances. The map substructure provides hierarchy, group, and matrix organization of references to topic instances.

structural type instance
An occurrence of a topic type or a map type in a document.

topic instance
An occurrence of a topic type in a document.

topic type
An element type that defines a complete unit of content. The topic type provides the root element for the topic and, through the contained element types, the substructure for the topic instances. The root element of the topic type is not necessarily the same as the root element of a document type; document types can nest multiple topic types and might also declare non-DITA wrapper elements as the root element for compatibility with other processes.

Specialization terminology
The following terminology is used to discuss DITA specialization:

base content model
The content model of a DITA element before specialization or the application of constraints or extensions.

base type
An element or attribute type that is not a specialization. All base types are defined by the DITA specification.

extension element
Within a vocabulary module, an element type that can be extended, replaced, or constrained for use in a DITA document type.

generalization
The process by which a specialized element is transformed into a less-specialized ancestor element or a specialized attribute is transformed into a less-specialized ancestor attribute. The original specialization-hierarchy information may be preserved in the generalized instance, thus allowing the original specialized type to be recreated from the generalized instance.

restricted content model
For a DITA element type, a content model that has been restricted from the base content model for the element type by one or more of the following mechanisms:
• Removing optional elements
• Requiring optional elements
• Ordering of unordered elements
• Restricting repeatable (but optional) elements from repeating

Content models can be restricted through the use of constraint modules or through specialization.

**selective domain extension**
An extension that replaces an extension element with element types that are defined in an domain module, thus making the base type unavailable in the DITA document-type shell that configures the extension.

**specialization**
(1) The act of defining new element or attribute types as a semantic refinement of existing element or attribute types
(2) An element or attribute type that is a specialization of a base type
(3) A process by which a generalized element is transformed into one of its more specialized element types or a generalized attribute is transformed into a more specialized attribute.

**specialization hierarchy**
The sequence of element or attribute types, from the most general to most specialized, from which a given element or attribute type is specialized. The specialization hierarchy for a DITA element is formally declared through its `@class` attribute.

**specialization parent**
For a given DITA element type, the most specialized of its ancestors in its specialization hierarchy.

**specialized attribute type**
An attribute type that is defined as a semantic refinement of another attribute type. The attribute type must be specialized from either the `@base` or `@props` attribute, and its range of permissible values must be a subset of or identical to the values allowed by the original attribute type.

**specialized element type**
An element type that is defined as a semantic refinement of an existing element type. The content allowed by the specialized element type must be a subset of or identical to the content allowed by the original element type. Within a DITA document, all specialized element types must be refinements of one of the base element types, with the exception of elements that are used in the context of `<foreign>` or `<unknown>` elements.

**structural type**
A topic type or map type.

**DITA modules**
The following terminology is used to discuss DITA modules:

**attribute domain module**
A domain module that defines exactly one specialization of either the `@base` or `@props` attribute.

**constraint module**
A set of declarations that imposes additional constraints onto the element or attribute types that are defined in a specific vocabulary module.

**domain module**
A set of element types or an attribute type that supports a specific subject or functional area. Element types or attribute types in a domain can be integrated with topic types or map types to enhance semantic support for particular kinds of content. For example, the structural type `<topic>` declares the `<keyword>` element; when integrated with a domain for describing user interfaces, new keyword specializations (such as `<wintitle>`) become available wherever `<keyword>` was allowed in the original structural type.

**element domain module**
A domain module that defines one or more element types for use within maps or topics.

**map module**
A structural module that defines a single map type.

**structural module**

A vocabulary module that defines exactly one top-level map type or topic type. Structural modules also can define specializations of elements from domain modules.

**topic module**

A structural module that defines a single top-level topic type.

**vocabulary module**

A uniquely-named unit of element type or attribute type declaration. There are two types of vocabulary modules: structural modules and domain modules. For a given map type, topic type, or domain, there is exactly one vocabulary module that defines it.

The following figure illustrates the relationship between a DTD-based DITA document, its DITA document-type shell, and the various vocabulary modules that it uses. A similar structure applies to DITA documents that use other XML grammars.

**Figure 3: Instances, modules, and declarations**

**Linking and addressing terms**

The following terminology is used to discuss linking and addressing terms:

**key name**

An identifier defined by a value of the @keys attribute. A key is bound to one or more of the following items:

- A resource addressed by the `<topicref>` element or a specialization of the `<topicref>` element
- An element contained within the `<topicmeta>` element of the `<topicref>` element or a specialization of the `<topicref>` element

**key definition**

A `<topicref>` element or specialization of a `<topicref>` element that specifies the @keys attribute and defines one or more key names.

**key scope**

A root map or section of a map hierarchy associated with a unique key space for purposes of key reference resolution.
key space
The set of unique key names that are defined for a given key scope. Within a given key scope, a key name has at most one definition, as determined by the key definition precedence rules.

key resolution context
The key scope that establishes the context for resolving key references. For a given key-resolution instance, there is at most one key scope that defines the effective key space, which in turn provides the effective key definition.

referenced element
An element that is referenced by another DITA element. See also referencing element.

Example
Consider the following code sample from a installation-reuse.dita topic. The <step> element that it contains is a referenced element; other DITA topics reference the <step> element by using the @conref attribute.

```xml
<step id="run-startcmd-script">
  <cmd>Run the startcmd script that is applicable to your operating-system environment.</cmd>
</step>
```

referencing element
An element that references another DITA element by specifying an addressing attribute. See also referenced element and addressing attribute

Example
The following <step> element is a referencing element. It uses the @conref attribute to reference a <step> element in the installation-reuse.dita topic.

```xml
<step conref="installation-reuse.dita#reuse/run-startcmd-script">
  <cmd/>
</step>
```

addressing attribute
An attribute, such as @conref, @conkeyref, @keyref, and @href, that specifies an address.

3.1.2 Basic concepts

DITA has been designed to satisfy requirements for information typing, semantic markup, modularity, reuse, interchange, and production of different deliverable forms from a single source. These topics provide an overview of the key DITA features and facilities that serve to satisfy these requirements.

DITA topics
In DITA, a topic is the basic unit of authoring and reuse. All DITA topics have the same basic structure: a title and, optionally, a body of content. Topics can be generic or more specialized; specialized topics represent more specific information types or semantic roles, for example, <concept>, <task>, <reference>, or <learningContent>. See DITA topics for more information.

DITA maps
DITA maps are documents that organize topics and other resources into structured collections of information. DITA maps specify hierarchy and the relationships among the topics; they also provide the contexts in which keys are defined and resolved. DITA maps SHOULD have .ditamap as the file extension. See DITA maps for more information.

Information typing
Information typing is the practice of identifying types of topics, such as concept, reference, and task, to clearly distinguish between different types of information. Topics that answer different reader questions (How ...? What is ...?) can be categorized with different information types. The base information types provided by DITA specializations (for example, technical content, machine industry, and learning and training) provide
starter sets of information types that can be adopted immediately by many technical and business-related organizations. See Information typing for more information.

**DITA linking**

DITA depends heavily on links. The purposes for which it provides links include defining the content and organization of publication structures (DITA maps), topic-to-topic navigation links and cross references, and reuse of content by reference. All DITA links use the same addressing mechanisms, either URI-based addresses or indirect addresses using keys and key references. See DITA linking for more information.

**DITA addressing**

DITA provides two addressing mechanisms. DITA addresses either are direct, URI-based addresses, or they are indirect key-based addresses. Within DITA documents, individual elements are addressed by unique identifiers specified on the @id attribute. DITA defines two fragment identifier syntaxes for addressing DITA elements, one for topics and elements within maps and another for non-topic elements within topics. See DITA addressing for more information.

**Content reuse**

The DITA @conref, @conkeyref, @conrefend, and related attributes provide a mechanism for reuse of content fragments within DITA topics or maps. See Content reuse for more information.

**Conditional processing**

Conditional processing, also known as profiling, is the filtering or flagging of information based on processing-time criteria. See Conditional processing for more information.

**Configuration**

A given DITA map or topic document is governed by a DITA document type that defines the set of structural modules (topic or map types), domain modules, and constraints modules that the map or topic can use. See Configuration for more information.

**Specialization**

The specialization feature of DITA allows for the creation of new element types and attributes that are explicitly and formally derived from existing types. The resulting specialization allows for the blind interchange of all conforming DITA content and a minimum level of common processing for all DITA content. It also allows specialization-aware processors to add specialization-specific processing to existing base processing. See Specialization for more information.

**Constraints**

Constraint modules define additional constraints for corresponding vocabulary modules in order to restrict content models or attribute lists for specific element types, remove extension elements from an integrated domain module, or replace base element types with domain-provided extension element types. Constraint modules do not and cannot change element semantics, only the details of how element types can be used in the context of a specific concrete document type. Because constraints can make optional elements required, documents that use the same vocabulary modules may still have incompatible constraints. Thus the use of constraints can affect the ability for content from one topic or map to be used directly in another topic or map. See Constraints for more information.

### 3.1.3 File naming conventions

DITA uses certain naming conventions and file extensions for topics, maps, modules, and document-type implementation files.

Files that contain DITA content SHOULD use the following naming conventions and file extensions:

**DITA topics**

- *.dita (preferred)
- *.xml
DITA maps
*.ditamap

Conditional processing profiles
*.ditaval

Files that define DITA document-type components **MUST** use the following naming conventions:

**Document-type shell files**
- `typename.dtd`
- `typename.xsd`
- `typename.rnc`
- `typename.rng`

Where `typename` is the name of the intended root topic or map type defined by the document type shell or, as needed, a name that clearly identifies both the intended root map or topic type and distinguishes the document type shell from other shells for the same root type.

For example, the OASIS-provided document-type shells for technical content include two different document-type shells for the task topic type: `task.dtd` and `generalTask.dtd`, where `task.dtd` includes the strict task body constraint module and `generalTask.dtd` does not.

**DTD structural module files**
- `typename.mod`
- `typename.ent`

**DTD domain module files**
- `typenameDomain.mod`
- `typenameDomain.ent`

**DTD constraint module files**
- `constraintnameConstraint.mod`

**Schema structural module files**
- `typenameMod.xsd`
- `typenameGrp.xsd`

**Schema domain module files**
- `typenameDomain.xsd`

**Schema constraint module files**
- `constraintnameConstraintMod.xsd`
- `constraintnameConstraintIntMod.xsd`

**RELAX NG structural module files**
- `typenameMod.rnc`
- `typenameMod.rng`

**RELAX NG domain module files**
- `typenameDomain.rnc`
- `typenameDomain.rng`

**RELAX NG constraint module files**
- `constraintnameConstraintMod.rnc`
3.1.4 Producing different deliverables from a single source

DITA is designed to produce multiple deliverable formats from a single set of DITA content. This means that many rendition details are specified neither in the DITA specification nor in the DITA content; the rendition details are defined and controlled by the processors.

Like many XML-based applications for human-readable documentation, DITA supports the separation of content from presentation. This is necessary when content is used in different contexts, since authors cannot predict how or where the material that they author will be used. The following features and mechanisms enable users to produce different deliverable formats from a single source:

**DITA maps**
Different DITA maps can be optimized for different delivery formats. For example, you might have a book map for printed output and another DITA map to generate online help; each map uses the same content set.

**Specialization**
The DITA specialization facility enables users to create the XML elements that are needed by a particular information set in order to provide appropriate rendition distinctions. In XML-based systems where the presentation details are defined as styles bound to elements, the more precise and detailed the markup, the easier it is to define presentation rendering. Because the use of specializations does not impede interchange or interoperability, DITA users can safely create the specializations that are demanded by their local delivery and rendition requirements, with a minimum of additional impact on the systems and business processes that depend on or use the content. While general XML practices suggest that element types should be semantic, specialization can be used to define element types that are purely presentational in nature. The highlighting domain is an example of such a specialization.

**Conditional processing**
Conditional processing makes it possible to have a topic or DITA map that contains delivery-specific content.

**Content referencing**
The conref mechanism makes it possible to construct delivery-specific maps or topics from a combination of generic components and delivery-context-specific components.

**Key referencing**
The keyref mechanism makes it possible to change variables for volatile content, redirect links, and reap the benefits of indirect addressing.

**@outputclass attribute**
The @outputclass attribute provides a mechanism whereby authors can indicate specific rendition intent where necessary. Note that the DITA specification does not define any values for the @outputclass attribute; the use of the @outputclass attribute is, by nature, processor specific.

While DITA is independent of any particular delivery format, it is a standard that supports the creation of human-readable content. As such, it defines some fundamental document components including paragraphs, lists, and table. When there is a reasonable expectation that such basic document components be rendered consistently, the DITA specification defines default or suggested renderings.

3.2 DITA markup

Topics and maps are the basic building blocks of the Darwin Information Typing Architecture (DITA). Metadata attributes and values can be added to DITA topics and maps, as well as to elements within topics, to allow for conditional publishing and content reuse.

DITA topics and maps are XML documents that conform to the XML specification. As such, they can be viewed, edited, validated, and processed with standard XML tools, although some DITA-specific features, such as
content reference, key reference, and specialization require DITA-specific processing for full implementation and validation.

### 3.2.1 DITA topics

DITA topics are the basic units of DITA content and the basic units of reuse. Each topic contains a single subject. Topics may be of specific specialized information types, such as task, concept, or reference, or may be generic, that is, without a specified information type.

#### 3.2.1.1 The topic as the basic unit of information

In DITA, a topic is the basic unit of authoring and reuse. All DITA topics have the same basic structure: a title and, optionally, a body of content. Topics can be generic or more specialized; specialized topics represent more specific information types or semantic roles, for example, `<concept>`, `<task>`, `<reference>`, or `<learningContent>`.

DITA topics consist of content units that can be as generic as sets of paragraphs and unordered lists or as specific as sets of instructional steps in a procedure or cautions to be considered before a procedure is performed. Content units in DITA are expressed using XML elements and can be conditionally processed using metadata attributes and values.

Classically, a DITA topic is a titled unit of information that can be understood in isolation and used in multiple contexts. It should be short enough to address a single subject or answer a single question but long enough to make sense on its own and be authored as a self-contained unit. However, DITA topics also can be less self-contained units of information, such as topics that contain only titles and short descriptions and serve primarily to organize subtopics or links or topics that are designed to be nested for the purposes of information management, authoring convenience, or interchange.

DITA topics are used as components of DITA maps. DITA maps enable topics to be organized in a hierarchy for publication. Large units of content, such as complex reference documents or book chapters, are created by nesting topic references in a DITA map. The same set of DITA topics can be used in any number of maps.

DITA topics also can be used and published individually; for example, one can represent an entire deliverable as a single DITA document that consists of a root topic and nested topics. This strategy can accommodate the migration of legacy content that is not topic-oriented; it also can accommodate information that is not meaningful outside the context of a parent topic. However, the power of DITA is most fully realized by storing each DITA topic in a separate XML document and using DITA maps to organize how topics are combined for delivery. This enables a clear separation between how topics are authored and stored and how topics are organized for delivery.

#### 3.2.1.2 The benefits of a topic-based architecture

Topics enable the development of usable and reusable content.

While DITA does not require the use of any particular writing practice, the DITA architecture is designed to support authoring, managing, and processing of content that is designed to be reused. Although DITA provides significant value even when reuse is not a primary requirement, the full value of DITA is realized when content is authored with reuse in mind. To develop topic-based information means creating units of standalone information that are meaningful with little or no surrounding context.

By organizing content into topics that are written to be reusable, authors can achieve several goals:

- Content is readable when accessed from an index or search, not just when read in sequence as part of an extended narrative. Since most readers do not read technical and business-related information from beginning to end, topic-oriented information design ensures that each unit of information can be read independently.
• Content can be organized differently for online and print delivery. Authors can create task flows and concept hierarchies for online delivery and create a print-oriented hierarchy to support a narrative content flow.

• Content can be reused in different collections. Since a topic is written to support random access (as by search), it should also be understandable when included as part of various product deliverables. Topics permit authors to refactor information as needed, including only the topics that apply to each unique scenario.

• Content is more manageable in topic form whether managed as individual files in a traditional file system or as objects in a content management system.

• Content authored in topics can be translated and updated more efficiently and less expensively than information authored in larger or more sequential units.

• Content authored in topics can be filtered more efficiently, encouraging the assembly and deployment of information subsets from shared information repositories.

Topics written for reuse should be small enough to provide opportunities for reuse but large enough to be coherently authored and read. Since each topic is written to address a single subject, authors can organize a set of topics logically and achieve an acceptable narrative content flow.

3.2.1.2.1 Disciplined, topic-oriented writing

Topic-oriented writing is a disciplined approach to writing that emphasizes modularity and reuse of concise units of information: topics. Well-designed DITA topics can be reused in many contexts, as long as writers are careful to avoid unnecessary transitional text.

Conciseness and appropriateness

Readers who are trying to learn or do something quickly appreciate information that is written in a structure that is easy to follow and contains only the information needed to complete that task or grasp a fact. Recipes, encyclopedia entries, car repair procedures—all serve up a uniquely focused unit of information. The topic contains everything required by the reader.

Locational independence

A well-designed topic is reusable in other contexts to the extent that it is context free, meaning that it can be inserted into a new document without revision of its content. A context-free topic avoids transitional text. Phrases like “As we considered earlier ...” or “Now that you have completed the initial step ...” make little sense if a topic is reused in a new context in which the relationships are different or no longer exist. A well-designed topic reads appropriately in any new context because the text does not refer the reader outside the topic.

Navigational independence

Most print publications or web pages are a mixture of content and navigation. Internal links lead a reader through a sequence of choices as he or she navigates through a website. DITA supports the separation of navigation from content by assembling independent topics into DITA maps. Nonetheless, writers may want to provide links within a topic to additional topics or external resources. DITA does not prohibit such linking within individual topics. The DITA relationship table enables links between topics and to external content. Since it is defined in the DITA map, it is managed independently of the topic content.

Links in the content are best used for cross-references within a topic. Links from within a topic to additional topics or external resources should be avoided because they limit the reusability of the topic. To link from a term or keyword to its definition, use the DITA keyref facility to avoid creating topic-to-topic dependencies that are difficult to maintain. See Key-based addressing.

3.2.1.2.2 Transitional text solutions

Topic orientation does not mean that transitions cannot be authored in DITA. The key to providing both locational independence and intentional sequences is the adroit use of DITA markup features.

The @print attribute on the <topicref> element includes the value, "printonly", which may be used to indicate that the topic be omitted when the DITA map is transformed to a format for which transitions are unnecessary.
Consequently, a topic designated as "printonly" may be written in any style that serves as a transitional topic in the flow of printed information.

Comment by Kristen Eberlein, 27 January 2014
Since we are deprecating @print, should we revise this topic?

You can also use conditional text to insert transitional sequences into a map so that you can include or exclude the content of short descriptions or paragraphs at the end of a topic. However, if you share conditionally marked topics with other business partners or teams, you must instruct them on the proper runtime settings to enable the conditions to be used the way you intended.

DITA does not preclude authoring transitional text; it does provide an environment that allows you to tag and manage transitional elements apart from surrounding, topic-encapsulated information.

3.2.1.3 Information typing

Information typing is the practice of identifying types of topics, such as concept, reference, and task, to clearly distinguish between different types of information. Topics that answer different reader questions (How ...? What is ...?) can be categorized with different information types. The base information types provided by DITA specializations (for example, technical content, machine industry, and learning and training) provide starter sets of information types that can be adopted immediately by many technical and business-related organizations.

Information typing has a long history of use in the technical documentation field to improve information quality. It is based on extensive research and experience, including Robert Horn's Information Mapping and Hughes Aircraft's STOP (Sequential Thematic Organization of Proposals) technique. Note that many DITA topic types are not necessarily closely connected with traditional Information Mapping.

Information typing is a practice designed to keep documentation focused and modular, thus making it clearer to readers, easier to search and navigate, and more suitable for reuse. Classifying information by type helps authors perform the following tasks:

- Develop new information more consistently
- Ensure that the correct structure is used for closely related kinds of information (retrieval-oriented structures like tables for reference information and simple sequences of steps for task information)
- Avoid mixing content types, thereby losing reader focus
- Separate supporting concept and reference information from tasks, so that users can read the supporting information if needed and ignore if it is not needed
- Eliminate unimportant or redundant detail
- Identify common and reusable subject matter

DITA currently defines a small set of well-established information types that reflects common practices in certain business domains, for example, technical communication and instruction and assessment. However, the set of possible information types is unbounded. Through the mechanism of specialization, new information types can be defined as specializations of the base topic type (<topic>) or as refinements of existing topics types, for example, <concept>, <task>, <reference>, or <learningContent>.

You need not use any of the currently-defined information types. However, where a currently defined information type matches the information type of your content, the currently defined information type should be used, either directly, or as a base for specialization. For example, information that is procedural in nature should use the task information type or a specialization of task. Consistent use of established information types helps ensure smooth interchange and interoperability of DITA content.
3.2.1.4 Generic topics

The element type `<topic>` is the base topic type from which all other topic types are specialized. All topics have the same basic structure.

For authors, typed content is preferred to support consistency in writing and presentation to readers. The generic topic type should only be used if authors are not trained in information typing or when a specialized topic type is inappropriate. The OASIS DITA standard provides several specialized topic types, including concept, task, and reference that are critical for technical content development.

For those pursuing specialization, new specialized topic types should be specialized from appropriate ancestors to meet authoring and output requirements.

3.2.1.5 Topic structure

All topics have the same basic structure, regardless of topic type: title, description or abstract, prolog, body, related links, and nested topics.

All DITA topics must have an XML identifier (the `@id` attribute) and a title. The basic topic structure consists of the following parts, some of which are optional:

**Topic element**

The topic element holds the required `@id` attribute and contains all other elements.

**Title**

The title contains the subject of the topic.

**Alternate titles**

Titles specifically for use in navigation or search. When not provided, the base title is used for all contexts.

**Short description or abstract**

A short description of the topic or a longer abstract with an embedded short description. The short description may be used both in topic content (as the first paragraph), in generated summaries that include the topic, and in links to the topic. Alternatively, the abstract lets you create more complex introductory content and uses an embedded short description element to define the part of the abstract that is suitable for summaries and link previews.

While short descriptions aren't required, they can make a dramatic difference to the usability of an information set and should generally be provided for all topics.

**Prolog**

The prolog is the container for topic metadata, such as change history, audience, product, and so on.

**Body**

The topic body contains the topic content: paragraphs, lists, sections, and other content that the information type permits.

**Related links**

Related links connect to other topics. When an author creates a link as part of a topic, the topic becomes dependent on the other topic being available. To reduce dependencies between topics and thereby increase the reusability of each topic, authors may use DITA maps to define and manage links between topics, instead of embedding links directly in each related topic.

**Nested topics**

Topics can be defined inside other topics. However, nesting requires special care because it can result in complex documents that are less usable and less reusable. Nesting may be appropriate for information that is first converted from desktop publishing or word processing files or for topics that are unusable independent from their parent or sibling topics.
The rules for topic nesting can be configured in a document-type shells. For example, the standard DITA configuration for concept topics only allows nested concept topics. However, local configuration of the concept topic type could allow other topic types to nest or disallow topic nesting entirely. In addition, the @chunk attribute enables topics to be equally re-usable regardless of whether they are separate or nested. The standard DITA configuration for ditabase document-type documents allows unrestricted topic nesting and may be used for holding sets of otherwise unrelated topics that hold re-usable content. It may also be used to convert DITA topics from non-DITA legacy source without first determining how individual topics should be organized into separate XML documents.

3.2.1.6 Topic content

The content of all topics, regardless of topic type, is built on the same common structures.

**Topic body**

The topic body contains all content except for that contained in the title or the short description/abstract. The topic body can be specialized to impose constraints appropriate for the specific topic type even when titles and prolog are generic, or the topic body can be generic while the topic title and prolog are specialized.

**Sections and examples**

The body of a topic might contain divisions, such as sections and examples. They might contain block-level elements like titles and paragraphs and phrase-level elements like API names or text. It is recommend that sections have titles, whether they are entered directly into the `<title>` element or rendered using a fixed or default title.

Either body divisions or unitted sections or examples may be used to delimit arbitrary structures within a topic body. However, body divisions may nest, but sections and examples cannot contain sections.

**<sectiondiv>**

The `<sectiondiv>` element enables the arbitrary grouping of content within a section for the purpose of content reuse. The `<sectiondiv>` element does not include a title. Content that requires a title should use `<section>` or `<example>`.

**<bodydiv>**

The `<bodydiv>` element enables the arbitrary grouping of content within the body of a topic for the purpose of content reuse. The `<bodydiv>` element does not include a title. Content that requires a title should use `<section>` or `<example>`.

**div**

The `<div>` element enables the arbitrary grouping of content within a topic. The `<div>` element does not include a title. Content that requires a title should use `<section>` or `<example>` or, possibly, `<fig>`.

**Block-level elements**

Paragraphs, lists, and tables are types of "block" elements. As a class of content, they can contain other blocks, phrases, or text, though the rules vary for each structure.

**Phrases and keywords**

Block-level elements can contain markup to label parts of a paragraph or parts of a sentence as having special semantic meaning or presentation characteristics, such as `<uicontrol>` or `<b>`. Phrases can usually contain other phrases and keywords as well as text. Keywords can only contain text.

**Images**

Images can be inserted to display photographs, illustrations, screen captures, diagrams, and more. At the phrase level, they can display trademark characters, icons, toolbar buttons, and the so forth.

**Multimedia**
With the `<object>` element, multimedia information may be added to display, for example, diagrams that can be rotated and expanded. With the `<foreign>` element, media may be included within topic content, for example, SVG graphics, MathML equations, and so on.

Comment by Kristen Eberlein, 27 January 2014
Does this need to be rewritten now that we are shipping domains for MathML and SVG?

3.2.1.7 Topic domains: Base DITA

A DITA vocabulary domain defines a set of elements that are associated with a particular subject area or authoring requirement. DITA incorporates the following domains into the base DITA content: highlighting, hazard statement, indexing, and utilities. Other domains are incorporated into the Technical Content and Learning and Training packages.

The elements in a domain are defined in a domain module. A domain module can be integrated with a topic type to make the domain elements available within the topic type. The following domains are provided as part of base DITA.

Table 1: DITA topic domains: Base DITA

<table>
<thead>
<tr>
<th>Domain</th>
<th>Description</th>
<th>Short name</th>
<th>Module name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hazard statements</td>
<td>For providing detailed information about safety hazards</td>
<td>hazard-d</td>
<td>hazardstatementDomain.mod (DTD)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>hazardstatementDomain.xsd (Schema)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>hazardstatementDomain.rng (RELAX NG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>hazardstatementDomain.rnc (RELAX NG Compact)</td>
</tr>
<tr>
<td>Highlighting</td>
<td>For highlighting when the appropriate semantic element does not exist yet</td>
<td>hi-d</td>
<td>highlightDomain.mod (DTD)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>highlightDomain.xsd (Schema)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>highlightDomain.rng (RELAX NG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>highlightDomain.rnc (RELAX NG Compact)</td>
</tr>
<tr>
<td>Indexing</td>
<td>For extended indexing functions such as see and see-also</td>
<td>indexing-d</td>
<td>indexingDomain.mod (DTD)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>indexingDomain.xsd (Schema)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>indexingDomain.rng (RELAX NG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>indexingDomain.rnc (RELAX NG Compact)</td>
</tr>
<tr>
<td>Utilities</td>
<td>For providing imagemaps, sort keys, and other useful structures</td>
<td>ut-d</td>
<td>utilitiesDomain.mod (DTD)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>utilitiesDomain.xsd (Schema)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>utilitiesDomain.rng (RELAX NG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>utilitiesDomain.rnc (RELAX NG Compact)</td>
</tr>
</tbody>
</table>
3.2.2 DITA maps

This topic collection contains information about DITA maps and the purposes that they serve. It also includes high-level information about DITA map elements, attributes, and metadata.

3.2.2.1 Definition of DITA maps

DITA maps are documents that organize topics and other resources into structured collections of information. DITA maps specify hierarchy and the relationships among the topics; they also provide the contexts in which keys are defined and resolved. DITA maps SHOULD have .ditamap as the file extension.

Maps draw on a rich set of existing best practices and standards for defining information models, such as hierarchical task analysis. They also support the definition of non-hierarchical relationships, such as matrices and groups, which provide a set of capabilities that has similarities to Resource Description Framework (RDF) and ISO topic maps.

DITA maps use <topicref> elements (or specializations of the <topicref> element) to reference DITA topics, DITA maps, and non-DITA resources, for example, HTML and TXT files. The <topicref> elements can be nested or grouped to create relationships between the referenced topics, maps, and non-DITA files; the <topicref> elements can be organized into hierarchies in order to represent a specific order of navigation or presentation.

DITA maps impose an architecture on a set of topics. Information architects can use DITA maps to specify what DITA topics are needed to support a given set of user goals and requirements; the sequential order of the topics; and the relationships that exist among those topics. Because DITA maps provide this context for topics, the topics themselves can be relatively context-free; they can be used and reused in multiple different contexts.

DITA maps often represent a single deliverable, for example, a specific Web site, a printed publication, or the online help for a product. DITA maps also can be subcomponents for a single deliverable, for example, a DITA map might contain the content for a chapter in a printed publication or the troubleshooting information for an online help system. The DITA specification provides specialized map types; book maps represent printed publications, subject scheme maps represent taxonomic or ontological classifications, and learning maps represent formal units of instruction and assessment. However, these map types are only a starter set of map types reflecting well-defined requirements.

DITA maps establish relationships through the nesting of <topicref> elements and the application of the @collection-type attribute. Relationship tables may also be used to associate topics with each other based on membership in the same row; for example, task topics can be associated with supporting concept and reference topics by placing each group in cells of the same row. During processing, these relationships can be rendered in different ways, although they typically result in lists of "Related topics" or "For more information" links. Like many aspects of DITA, the details about how such linking relationships are presented is determined by the DITA processor.

DITA maps also define keys and organize the contexts in which key references are resolved. A <topicref> or <map> element, or any specialization, may be used to create a new context for key definition and resolution. A <topicref> element, or specialized <topicref> such as <keydef>, may also be used to define a key which binds that key name to a specified resource for references in the same key scope.

Comment by jtv

DITA maps also define keys and organize the key scopes in which key references are resolved. A <topicref> or <map> element, or any specialization, may be used to create a new scope for key definition and resolution. A <topicref> element, or specialized <topicref> such as <keydef>, may also be used to define a key which binds that key name to a specified resource for references in the same key scope.

Eberlein: Jim basically suggested replacing the term "context" with "Key scopes" or "scope."
3.2.2.2 Purpose of DITA maps

DITA maps enable the scalable reuse of content across multiple contexts. They can be used by information architects, writers, and publishers to plan, develop, and deliver content.

DITA maps support the following uses:

**Defining an information architecture**
Maps can be used to define the topics that are required for a particular audience, even before the topics themselves exist. DITA maps can aggregate multiple topics for a single deliverable.

**Defining what topics to build for a particular output**
Maps reference topics that are included in output processing. Information architects, authors, and publishers can use maps to specify a set of topics that are processed at the same time, instead of processing each topic individually. In this way, a DITA map can serve as a manifest or bill of materials.

**Defining navigation**
Maps can define the online navigation or table of contents for a deliverable.

**Defining related links**
Maps define relationships among the topics they reference. These relationships are defined by the nesting of elements in the DITA map, relationship tables, and the use of elements on which the `@collection-type` attribute is set. On output, these relationships might be expressed as related links or the hierarchy of a table of contents (TOC).

**Defining an authoring context**
The DITA map can define the authoring framework, providing a starting point for authoring new topics and integrating existing ones.

**Defining keys and key scopes**
Maps can define keys, which provide an indirect addressing mechanism that enhances portability of content. The keys are defined by `<topicref>` elements or specialization of `<topicref>` elements, such as `<keydef>`. The `<keydef>` element is a convenience element; it is a specialized type of a `<topicref>` element with the following attributes:

- A required `@keys` attribute
- A `@processing-role` attribute with a default value of "resource-only".

Maps also define the context or contexts for resolving key-based references, such as elements that specify the `@keyref` or `@conkeyref` attribute. Elements within a map structure that specify a `@keyscope` attribute create a new context for key reference resolution. Key references within such elements are resolved against the set of effective key definitions for that scope.

Specialized maps can provide additional semantics beyond those of organization, linking, and indirection. For example, the subjectScheme map specialization adds the semantics of taxonomy and ontology definition.

3.2.2.3 DITA map elements

A DITA map describes the relationships among a set of DITA topics. The DITA map and map group elements organize topics into hierarchies, groups, and relationships; they also define keys.

A DITA map is composed of the following elements:
The `<map>` element is the root element of the DITA map.

The `<topicref>` elements are the basic elements of a map. A `<topicref>` element can reference a DITA topic, a DITA map, or a non-DITA resource. A `<topicref>` element also can have a title, short description, and the same kind of prolog-level metadata that is available in topics.

The `<topicref>` elements can be nested to create a hierarchy, which can be used to define a table of contents (TOC) for print output, online navigation, and parent/child links. Hierarchies can be annotated using the `@collection-type` attribute to define a particular type of relationship, such as a set of choices, a sequence, or a family. These collection types can affect link generation, and they may be interpreted differently for different outputs.

Relationship tables are defined with the `<reltable>` element. Relationship tables can be used to define relationships between DITA topics or between DITA topics and non-DITA resources. In a relationship table, the columns define common attributes, metadata, or information types (for example, task or troubleshooting) for the resources that are referenced in that column. The rows define relationships between the resources in different cells of the same row.

The `<relrow>`, `<relcell>`, `<relheader>`, and `<relcolspec>` elements are used to define the components of the relationship table. Relationships defined in the relationship table also can be further refined by using the `@collection-type` attribute.

The `<topicgroup>` element defines a group or collection outside of a hierarchy or relationship table. It is a convenience element that is equivalent to a `<topicref>` element without an `@href` attribute or navigation title. Groups can be combined with hierarchies and relationship tables, for example, by including a `<topicgroup>` element within a set of siblings in a hierarchy or within a table cell. The `<topicref>` elements so grouped can then share inherited attributes and linking relationships with no effect on the navigation or table of contents.

Most map-level elements, including the map itself, can contain metadata inside the `<topicmeta>` element. Metadata typically is applied to an element and its descendants.

The `<ux-window>` element enables authors to define windowing information for the display of output topics that are appropriate to the delivery platform. Window management is important in user assistance and help system outputs, as well as for other hypertext and electronic delivery modes.

The `<topichead>` element provides a navigation title; it is a convenience element that is equivalent to a `<topicref>` element with a navigation title but no `@href` attribute.

The `<anchor>` element provides an integration point that another map can reference in order to insert its navigation into the current navigation tree. For those familiar with Eclipse help systems, this serves the same purpose as the `<navref>` element in that system. It might not be supported for all output formats.

The `<navref>` element represents a pointer to another map which should be preserved as a transcluding link rather than resolved. Output formats that support such linking will integrate the referenced resource when displaying the referencing map to an end user.
<keydef>
Enables authors to define keys. This element is a convenience element; it is a specialization of <topicref> that sets the default value of the @processing-role attribute to "resource-only". Setting the @processing-role attribute to resource-only ensures that the resource referenced by the key definition is not directly included in the navigation that is defined by the map.
</keydef>

<mapref>
Enables authors to reference an entire DITA map, including hierarchy and relationship tables. This element is a convenience element; it is a specialization of <topicref> that sets the default value of the @format attribute to "ditamap". The <mapref> element represents a reference from a parent map to a subordinate map.
</mapref>

<topicset>
Enables authors to define a branch of navigation in a DITA map so that it can be referenced from another DITA map.
</topicset>

<topicsetref>
Enables authors to reference a navigation branch that is defined in another DITA map.
</topicsetref>

<anchorref>
Enables authors to define a map fragment that is pushed to the location defined by an anchor.
</anchorref>

3.2.2.4 DITA map attributes

DITA maps have unique attributes that are designed to control the way that relationships are interpreted for different output purposes. In addition, DITA maps share many metadata and linking attributes with DITA topics.

Attributes unique to DITA maps

DITA maps often encode structures that are specific to a particular medium or output, for example, Web pages or a PDF document. Attributes, such as @print and @toc, are designed to help processors interpret the DITA map for each kind of output. These attributes are not available in DITA topics; individual topics, once separated from the high-level structures and dependencies associated with a particular kind of output, should be entirely reusable regardless of the intended output format. The @collection-type and @linking attributes affect how related links are generated for topics that are referenced in the DITA map.

@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. Where the @collection-type attribute is available on elements that cannot directly contain elements (such as <reltable> or <topicref>), the behavior of the attribute is reserved for future use.

@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 should participate 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.

```
@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.

```
@navtitle
```

- Specifies a navigation title. This is a shorter version of the title that is used in the navigation only. By default, the `@navtitle` attribute is ignored; it serves only to help the DITA map author keep track of the title of the topic.

  **Note:** The `@navtitle` attribute is deprecated in favor of the `<navtitle>` element. When both a `<navtitle>` element and a `@navtitle` attribute are specified, the `<navtitle>` element should be used.

```
@locktitle
```

- If `@locktitle` is set to "yes", the `<navtitle>` element or `@navtitle` attribute is used if it is present. Otherwise, the `<navtitle>` element or `@navtitle` attribute is ignored and the navigation title is retrieved from the referenced file.

  **Note:** The `@navtitle` attribute is deprecated in favor of the `<navtitle>` element. When both a `<navtitle>` element and a `@navtitle` attribute are specified, the `<navtitle>` element should be used.

```
@print
```

- Specifies whether the topic should be included in printed output.

  **Note:** Beginning with DITA 1.3, the `@print` attribute is deprecated. It is replaced with a conditional processing attribute: `@deliveryTarget`. See `@deliveryTarget` for more details.

```
@search
```

- Specifies whether the topic should be included in search indexes.

```
@chunk
```

- Specifies that the processor generates an interim set of DITA topics that are used as the input for the final processing. This can produce the following output results:
  - Multi-topic files are transformed into smaller files, for example, individual HTML files for each DITA topic.
  - Individual DITA topics are combined into a single file.
Specifying a value for the @chunk attribute on a <map> element establishes chunking behavior that applies to the entire map, unless overridden by @chunk attributes that are set on more specific elements in the DITA map. For a detailed description of the @chunk attribute and its usage, see Chunking (see page 102).

@copy-to
In most situations, specifies whether a duplicate version of the topic is created when it is transformed. This duplicate version can be either literal or virtual. The value of the @copy-to attribute specifies the uniform resource identifier (URI) by which the topic can be referenced by a @conref attribute, <topicref> element, or <xref> element. The duplication is a convenience for output processors that use the URI of the topic to generate the base address of the output. The @keys and @keyref attributes provide an alternative mechanism; they enable references to topics in specific-use contexts without making copies.

The @copy-to attribute also can be used to specify the name of a new chunk when topics are being chunked; it also can be used to determine the name of the stub topic that is generated from a topicref <>element that contains a title but does not specify a target. In both of those cases, no duplicate version of the topic is generated.

For information on how the @copy-to attribute can be used with the @chunk attribute, see Chunking (see page 102).

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

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

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

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

@cascade
Specifies whether the default rules for the cascading of metadata attributes in a DITA map apply. In addition to the following specified values, processors also MAY define additional values.

cascade="merge"
The metadata attributes cascade; the values of the metadata attributes are additive. This is the processing default for the @cascade attribute and was the only defined behavior for DITA 1.2 and earlier.

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

For more information, see How the cascade attribute functions (see page 37).

@keys
Specifies one or more key names.

@keyscope
Defines a new scope for key definition and resolution, and gives the scope one or more names, separated by white space. Key definitions within a scope can only be addressed directly by key references within the same scope. The key space defined by a key scope also inherits all of the key definitions from its parent scope, and if a key is defined in both the parent and child scopes, the definition from the parent scope takes precedence. Scoped key definitions are also added to the parent scope's key space, prepended by the scope name.
followed by a period. If a scope has more than one name, then each key within the scope will be contributed to the parent scope once for each prefix.

**Attributes shared by DITA maps and DITA topics**

The following metadata and reuse attributes are used by both DITA maps and DITA topics:

- @product, @platform, @audience, @otherprops, @rev, @status, @importance
- @dir, @xml:lang, @translate
- @id, @conref, @conrefend, @conkeyref, @conaction
- @props and any attribute specialized from @props
- @search

DITA maps also use many of the following attributes that are used with `<link>` or `<xref>` elements in DITA topics:

- @format
- @href
- @keyref
- @scope
- @type
- @query

When new attributes are specialized from @props or @base as a domain, they can be incorporated into both map and topic structural types.

**How the collection-type and linking attributes work in a relationship table**

The following example illustrates how linkage is defined in a DITA map:

```xml
<topicref href="A.dita" collection-type="sequence">
  <topicref href="A1.dita"/>
  <topicref href="A2.dita"/>
</topicref>
<reltab>
  <relrow>
    <relcell><topicref href="A.dita"/></relcell>
    <relcell><topicref href="B.dita"/></relcell>
  </relrow>
</reltab>
```

*Figure 4: Simple linking example*

When the output is generated, the topics contain the following linkage:

**A**
- Links to A1, A2 as children
- Links to B as related

**A1**
- Links to A as a parent
- Links to A2 as next in the sequence

**A2**
- Links to A as a parent
- Links to A1 as previous in the sequence

**B**
- Links to A as related
The following example illustrates how setting the `@linking` attribute can change the default behavior:

```xml
<topicref href="A.dita" collection-type="sequence">
  <topicref href="B.dita" linking="none"/>
  <topicref href="A1.dita"/>
  <topicref href="A2.dita"/>
</topicref>
<reltable>
  <relrow>
    <relcell><topicref href="A.dita"/></relcell>
    <relcell linking="sourceonly"><topicref href="B.dita"/></relcell>
  </relrow>
</reltable>
```

**Figure 5: Linking example with the `@linking` attribute**

When the output is generated, the topics contain the following linkage:

A
- Links to A1, A2 as children
- Does not link to B as a child or related topic

A1
- Links to A as a parent
- Links to A2 as next in the sequence
- Does not link to B as previous in the sequence

A2
- Links to A as a parent
- Links to A1 as previous in the sequence

B
- Links to A as a related topic

**How the `@cascade` attribute functions**

The following example illustrates how the `@cascade` attribute can be used to fine tune how the values for the `@platform` attribute apply to topics referenced in a DITA map. Here a DITA map contains a collection of topics that apply to Windows, Linux, and Macintosh OS; it also contains a topic that is only applicable to users running the application on Linux.

```xml
<map product="PuffinTracker" platform="win linux mac" cascade="nomerge">
  <title>Puffin Tracking Software</title>
  <topicref href="intro.dita" navtitle="Introduction"/>
  <topicref href="setup.dita" navtitle="Setting up the product"/>
  <topicref href="linux-instructions.dita" navtitle="Linux instructions" platform="linux" cascade="nomerge"/>
</map>
```

The values of the `@platform` attribute set at the map level cascade throughout the map and apply to the "Introduction" and "Setting up the product" topics. However, since the value of the `@cascade` attribute is set to "nomerge", the value of the `@platform` attribute for the "Linux instructions" topic does not merge with the values that cascade from above in the DITA map. The effective value of the `@platform` attribute for `linux-instructions.dita` is "linux".

The same results are produced by the following mark-up:

```xml
<map product="PuffinTracker" platform="win linux mac">
  <title>Puffin Tracking Software</title>
  <topicref href="intro.dita" navtitle="Introduction"/>
  <topicref href="setup.dita" navtitle="Setting up the product"/>
  <topicref href="linux-instructions.dita" navtitle="Linux instructions" platform="linux" cascade="nomerge"/>
</map>
```
3.2.2.5 Subject scheme maps and their usage

Subject scheme maps can be used to define controlled values and taxonomies. The controlled values and taxonomic subjects can be bound to attributes and element and attribute pairs.

3.2.2.5.1 Subject scheme maps

A subject scheme map enables adopters to create custom controlled values and to manage metadata attribute values without having to write a DITA specialization. Subject scheme maps use key definition to define a collection of controlled values rather than a collection of topics.

A controlled value is a keyword that can be used as a value for an attribute. For example, the @audience attribute can take a value that identifies the users that are associated with a particular product. Typical values for a medical-equipment product line might include therapist, oncologist, physicist, and radiologist. In a subject scheme map, an information architect can define a list of these @audience values.

Authoring tools SHOULD use these lists of controlled values to provide lists from which authors can select values when they specify attribute values. If controlled values for a metadata attribute are defined using the subject scheme map, tools MAY give an organization a list of readable labels, a hierarchy of values to simplify selection, and a shared definition of the value.

Controlled values can be used to classify content for filtering and flagging at build time. They also can be used for retrieval and traversal of the content at run time in conjunction with information viewing application that provide such functionality.

Comment by jelovirta
What is the interaction between key scopes and <subjectdef>? Should subject scheme maps be processed with the same rules for key scopes as normal keys?

Validating metadata attributes against a subject scheme

3.2.2.5.2 Controlled values for attributes

Subject scheme maps can define controlled values for DITA attributes.

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

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

- The <navtitle> element can provide a more readable value name.
- The <shortdesc> can provide a definition.

In addition, the <subjectdef> element can use an @href attribute to refer to a more detailed definition of the subject.

An editor MAY support drilling down to the subject definition topic for a detailed explanation of the subject. Tools MAY produce a help file, PDF, or other readable catalog to help authors better understand the controlled values.

An enumeration of controlled values can be defined with hierarchical levels by nesting subject definitions. This has an impact on how processors perform filtering and flagging. Processors SHOULD be aware of the hierarchy defined by nesting and process accordingly.

Comment by Kristen Eberlein, 7 October 2014
Need more precise working for above normative statement.

Example: Controlled values that provide additional information about the subject

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

```
<subjectdef keys="terminology" href="terminology.dita">
  <subjectdef keys="rf2119" href="rfc-2119.dita">
    <topicmeta>
      <navtitle>RFC-2119 terminology</navtitle>
      <shortdesc>The normative terminology that the DITA TC uses for the DITA specification</shortdesc>
    </topicmeta>
  </subjectdef>
</subjectdef>
```

Example: A hierarchy of controlled values and conditional processing

The following example illustrates a set of controlled values that contains a hierarchy.

```
<subjectScheme>
  <subjectdef keys="users">
    <subjectdef keys="therapist">
      <subjectdef keys="novice"/>
      <subjectdef keys="expert"/>
    </subjectdef>
    <subjectdef keys="oncologist"/>
    <subjectdef keys="physicist"/>
    <subjectdef keys="radiologist"/>
  </subjectdef>
</subjectScheme>
```

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

- If "therapist" is excluded and neither "novice" and "expert" are explicitly included, processors automatically should exclude both "novice" and "expert."
- If "novice" is included and "therapist" is not explicitly excluded, processors automatically should include "therapist" since it applies to "novice".
- If "therapist" is flagged and "novice" is not explicitly flagged, processors automatically should flag "novice" since it is a type of therapist.

3.2.2.5.3 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 has an impact on the expected behavior for processors and authoring tools.

If a subject scheme is used and an enumeration is bound, tools SHOULD validate controlled values for attributes against the subject scheme map. For authoring tools, this prevents users from entering misspelled or undefined values. For processors, processors SHOULD check that all values listed for an attribute in a DITAVAL file are bound to the attribute by the scheme before filtering or flagging.

Comment by Kristen Eberlein, 7 October 2014

The above paragraph originally contained the injunction "As with all key definitions and references, the reference must appear in the highest map that makes use of the controlled values."

The `<enumerationdef>` element binds the set of controlled values to an attribute. An enumeration can specify an empty category without children. In this case, no value is valid for the attribute. An enumeration also can specify an optional default value by using the `<defaultSubject>`.
Example: Binding a list of controlled values to the @audience attribute

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

```xml
<subjectScheme>
  <!-- Define types of users -->
  <subjectdef keys="users">
    <subjectdef keys="therapist"/>
    <subjectdef keys="oncologist"/>
    <subjectdef keys="physicist"/>
    <subjectdef keys="radiologist"/>
  </subjectdef>

  <!-- Bind the "users" subject to the @audience attribute. This restricts the @audience attribute to the following values: therapist, oncologist, physicist, radiologist -->
  <enumerationdef>
    <attributedef name="audience"/>
    <subjectdef keyref="users"/>
  </enumerationdef>
</subjectScheme>
```

Note that "users" is not a valid value for the @audience attribute.

3.2.2.5.4 Scaling a list of controlled values to define a taxonomy

Optional classification elements make it possible to create a taxonomy from a list of controlled values. A taxonomy differs from a controlled values list primarily in the degree of precision with which the metadata values are defined. A set of controlled values lists sometimes is regarded as the simplest form of taxonomy. Regardless of whether the goal is a simple list of controlled values or a taxonomy:

- The same core elements are used: <subjectScheme>, <subjectdef>, and <subjectref>.
- A category and its subjects can have a binding that enumerates the values of a metadata attribute.

Beyond the core elements and the attribute binding elements, sophisticated taxonomies can take advantage of some optional elements. These optional elements make it possible to specify more precise relationships among subjects. The <hasNarrower>, <hasPart>, <hasKind>, <hasInstance>, and <hasRelated> elements specify the kind of relationship in a hierarchy between a container subject and its contained subjects.

While users who have access to sophisticated processing tools benefit from defining taxonomies with this level of precision, other users can safely ignore this advanced markup and define taxonomies with hierarchies of subjectdef elements that are not precise about the kind of relationship between the subjects.

Example: A taxonomy defined using subject scheme elements

The following example defines San Francisco as both an instance of a city and a geographic part of California.

```xml
<subjectScheme>
  <hasInstance>
    <subjectdef keys="city" navtitle="City">
      <subjectdef keys="la" navtitle="Los Angeles"/>
      <subjectdef keys="nyc" navtitle="New York City"/>
      <subjectdef keys="sf" navtitle="San Francisco"/>
    </subjectdef>
    <subjectdef keys="state" navtitle="State">
      <subjectdef keys="ca" navtitle="California"/>
    </subjectdef>
  </hasInstance>
</subjectScheme>
```
Sophisticated tools can use this scheme to associate content about San Francisco with related content about other California places or with related content about other cities (depending on the interests of the current user).

The scheme can also define relationships between subjects that are not hierarchical. For instance, cities sometimes have "sister city" relationships. The example scheme could add a `<subjectRelTable>` element to define these associative relationships, with a row for each sister-city pair and the two cities in different columns in the row.

### 3.2.2.6 Examples of DITA maps

This section of the specification contains examples of DITA maps. The examples illustrate the wide variety of purposes that DITA maps serve.

**Comment by Kristen Eberlein**, 7 October 2014

The children topic were relocated from the "DITA map elements" topic. They each need a short description and more contextual information that clarifies the importance of the example, that is, why we think the example should be in the DITA specification.

#### 3.2.2.6.1 Example: Simple DITA map with a relationship table

**Comment by Kristen James Eberlein**, 21 April 2009

This section contains content untouched from the 1.1 arch spec. Is this example helpful and appropriate? The reltable topic in the lang spec 1.2 contains a different and more developed example of a reltable, using the bat topics. Do we want to ensure that we conref and use the same material in both specifications?

The following example contains the markup for a simple relationship table:

```xml
<map>
  <!-- ... -->
  <reltable>
    <relheader>
      <relcolspec type="concept"/>
      <relcolspec type="task"/>
      <relcolspec type="reference"/>
    </relheader>
    <relrow>
      <relcell>
        <topicref href="A.dita"/>
      </relcell>
      <relcell>
        <topicref href="B.dita"/>
      </relcell>
      <relcell>
        <topicref href="C1.dita"/>
      </relcell>
      <relcell>
        <topicref href="C2.dita"/>
      </relcell>
    </relrow>
  </reltable>
</map>
```
A DITA-aware tool might represent the relationship table graphically:

<table>
<thead>
<tr>
<th>type=&quot;concept&quot;</th>
<th>type=&quot;task&quot;</th>
<th>type=&quot;reference&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C1, C2</td>
</tr>
</tbody>
</table>

When the output is generated, the topics contain the following linkage:

- **A**
  - Links to B, C1, and C2
- **B**
  - Links to A, C1, and C2
- **C1, C2**
  - Links to A and B

### 3.2.2.6.2 Example: Simple DITA map that references a subordinate map

The following code sample illustrates how a DITA map can reference another DITA map:

```xml
<map>
  <title>DITA work at OASIS</title>
  <topicref href="oasis-dita-technical-committees.dita">
    <topicref href="dita_technical_committee.dita"/>
    <topicref href="dita_adoption_technical_committee.dita"/>
  </topicref>
  <mapref href="oasis-processes.ditamap"/>
</map>
```

The map also could be tagged in the following way:

```xml
<map>
  <title>DITA work at OASIS</title>
  <topicref href="oasis-dita-technical-committees.dita">
    <topicref href="dita_technical_committee.dita"/>
    <topicref href="dita_adoption.Technical.Committee.dita"/>
  </topicref>
  <topicref href="oasis-processes.ditamap" format="ditamap"/>
</map>
```

With either of the above examples, during processing, the map is resolved in the following way:

```xml
<map>
  <title>DITA work at OASIS</title>
  <topicref href="oasis-dita-technical-committees.dita">
    <topicref href="dita_technical_committee.dita"/>
    <topicref href="dita_adoption.Technical.Committee.dita"/>
  </topicref>
  <topicref href="oasis-processes.ditamap" format="ditamap"/>
  <!-- Contents of the oasis-processes.ditamap file -->
</map>
```

### 3.2.2.6.3 Example: Simple DITA map that defines keys

The following example illustrates how keys can be defined:

```xml
<map>
  <keydef keys="dita-tc" href="dita_technical_committee.dita"/>
  <keydef keys="dita-adoption" href="dita_adoption.Technical.Committee.dita"/>
</map>
```
The map also could be tagged in either of the following ways:

### <topicref> element with @processing-role attribute set to "resource-only"

```xml
<map>
  <topicref keys="dita-tc" href="dita_technical_committee.dita" processing-role="resource-only"/>
  <topicref keys="dita-adoption" href="dita_adoption_technical_committee.dita" processing-role="resource-only"/>
</map>
```

### <topicref> element with @toc, @linking, and @search attributes set to "no"

```xml
<map>
  <topicref keys="dita-tc" href="dita_technical_committee.dita" toc="no" linking="no" search="no"/>
  <topicref keys="dita-adoption" href="dita_adoption_technical_committee.dita" toc="no" linking="no" search="no"/>
</map>
```

**Comment by Kristen James Eberlein, 24 August 2009**

Example added based on review #1 comment from Elliot Kimber. Exactly what do we want to communicate in this example?

- Best practices for key definitions: Using a separate map, defining keys at beginning of map, etc.?
- First key encountered is used?
- That `<keydef>` is equivalent to `<topicref processing-role="resource-only">` or setting @toc, @linking, and @search attributes to "no"?

Should the example include more information about why the keys are defined and how they would be resolved?

### 3.2.2.6.4 Example: DITA maps that use `<anchor>` and `<anchorref>` elements

In this example, the `map1.ditamap` contains an `<anchor>` element with an @id attribute set to "a1".

```xml
<map>
  <title>MyComponent tasks</title>
  <topicref navtitle="Start here" href="start.dita" toc="yes">
    <navref mapref="othermap2.ditamap"/>
    <navref mapref="othermap3.ditamap"/>
    <anchor id="a1"/>
  </topicref>
</map>
```

**Figure 6: DITA map that contains an anchor**

The @id on an `<anchor>` element can be referenced by the @anchorref attribute on another map's `<map>` element. For example, the map to be integrated at that spot could be defined as follows.

```xml
<map anchorref="map1.ditamap#a1">
  <title>This map is pulled into the MyComponent task map</title>
</map>
```

**Figure 7: DITA map that references an anchor**
3.2.3 DITA metadata

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

3.2.3.1 Metadata elements

The metadata elements, many of which map to Dublin core metadata, are available in topics and DITA maps. This design enables authors and information architects to use identical metadata markup in both topics and maps.

The `<metadata>` element is a wrapper element that contains many of the metadata elements. In topics, the `<metadata>` element is available in the `<prolog>` element. In maps, the `<metadata>` element is available in the `<topicmeta>` element.

In DITA maps, the metadata elements also are available directly in the `<topicmeta>` element. However, authors should use the metadata elements that are available in the `<metadata>` element, as it better supports reuse between topics and maps. Collections of metadata can be shared between DITA maps and topics by using the conref or keyref mechanism.

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

Note: Not all metadata elements are available in the `<metadata>` element. However, they are available in either the topic `<prolog>` element or the map `<topicmeta>` element.

Related Links

Dublin Core Metadata Initiative (DCMI)

3.2.3.2 Metadata attributes

Certain attributes are common across most DITA elements. These attributes support content referencing, conditional processing, application of metadata, and globalization and localization.

3.2.3.2.1 Conditional processing attributes

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

Metadata attributes typically are used for the following purposes:

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

Typically `@audience`, `@platform`, `@product`, `@otherprops`, `@props`, `@deliveryTarget`, and specializations of the `@props` attributes are used for filtering; the same attributes plus the `@rev` attribute are used for flagging. The `@status` and `@importance` attributes, as well as custom attributes specialized from `@base`, are used for application-specific behavior, such as identifying metadata to aid in search and retrieval.

Filtering and flagging attributes

The following conditional-processing attributes are available on most elements:

`@product`
The product that is the subject of the discussion.

@platform
The platform on which the product is deployed.

@audience
The intended audience of the content.

@deliveryTarget
The intended delivery target of the content, for example "html", "pdf", or "epub". This attribute is a replacement for the now deprecated @print attribute.

The @deliveryTarget attribute is defined in the deliveryTargetAttDomain, which is integrated into all OASIS-provided document-type shells. If this domain is not integrated into a given document-type shell, the @deliveryTarget attribute will not be available.

@rev
The revision or draft number of the current document. (This is used only for flagging.)

@otherprops
Other properties that do not require semantic identification.

@props
A generic conditional processing attribute that can be specialized to create new semantic conditional-processing attributes.

In general, a conditional processing attribute provides a list of one or more values separated with whitespace, or it contains one or more groups that subcategorize the attribute. For instance, audience="administrator programmer" qualifies the content as applying to administrators and programmers. Using groups, product="database(A B) appserver(C)" qualifies the content as applying to databases A and B, as well as to application server C.

Other metadata attributes

Other attributes are still considered metadata on an element, but they are not designed for filtering or flagging.

@importance
The degree of priority of the content. This attribute takes a single value from an enumeration.

@status
The current state of the content. This attribute takes a single value from an enumeration.

@base
A generic attribute that has no specific purpose, but is intended to act as the basis for specialized attributes that have a simple value syntax like the conditional processing attributes (one or more alphanumeric values separated by whitespace).

@outputclass
Provides a label on one or more element instances, typically to specify a role or other semantic distinction. As the @outputclass attribute does not provide a formal type declaration or the structural consistency of specialization, it should be used sparingly, usually only as a temporary measure while a specialization is developed. For example, <uicontrol> elements that define button labels could be distinguished by adding an @outputclass attribute:

```xml
<uicontrol outputclass="button">Cancel</uicontrol>
```

The value of the @outputclass attribute can be used to trigger XSLT or CSS rules, while providing a mapping to be used for future migration to a more specialized set of user interface elements.

Related Links

4.12.1.2 Metadata attribute group (see page 651)
The metadata attribute group includes common metadata attributes, several of which support conditional processing (filtering and flagging) or the creation of new attribute domain specializations.
4.9 DITAVAL elements (see page 414)

A conditional processing profile (DITAVAL file) is used to identify which values are to be used for conditional processing during a particular output, build, or some other purpose. The profile should have an extension of .ditaval.

3.3.7 Conditional processing (profiling) (see page 88)

Conditional processing, also known as profiling, is the filtering or flagging of information based on processing-time criteria.

3.2.3.2.2 Translation and localization attributes

DITA elements have several attributes that support localization and translation.

@xml:lang

Identifies the language of the content, using the standard language and country codes. For instance, French Canadian is identified by the value fr-ca. The @xml:lang attribute asserts that all content and attribute values within the element bearing the attribute are in the specified language, except for contained elements that declare a different language.

@translate

Determines whether the element requires translation. A default value may be inferred from the element type. For example, <apiname> may be untranslated by default, whereas <p> may be translated by default.

@dir

Determines the direction in which the content should be rendered.

3.2.3.2.3 Architectural attributes

The architectural attributes specify the version of DITA that the content supports, identify the DITA domains that are in use by the content, and provide essential information about the specializations that are in use by the content.

The architectural attributes should not be marked up in the source DITA map and topic instances. Instead, the values of the architectural attributes are handled by the processor when the content is processed, preferably through defaults set in the XML grammar. This practice ensures that the DITA content instances do not specify invalid values for the architectural attributes.

The architectural attributes are as follows:

@class

This attribute identifies the specialization modules for the element type as well as its ancestors. Every DITA element (except the <dita> element that is used as the root of a database document) has a @class attribute.

@domains

This attribute identifies the domain specialization modules that are used in a map or topic and, for each domain module, its module dependencies. The root element of every topic and map has a @domains attribute.

@DITAArchVersion

This attribute identifies the version of the DITA architecture that is used by the XML grammar. The root element of every topic and map has a @DITAArchVersion attribute. The attribute is declared in a DITA namespace to allow namespace-sensitive tools to detect DITA markup.

To make the document instance usable in the absence of an XML grammar, a normalization process can set the architectural attributes in the document instance.
3.2.3.3 Metadata in maps and topics

Topic metadata can be specified in a DITA map as well as in the topics that the map references. By default, metadata in the map supplements or overrides metadata that is specified at the topic level, unless the @lockmeta attribute of the <topicmeta> element is set to "no".

Where metadata about topics can be specified

Information about topics can be specified as metadata on the map, as attributes on the <topicref> element, or as metadata attributes or elements in the topic itself:

DITA map: Metadata elements

At the map level, properties can be set by using metadata elements. They can be set for an individual topic, for a set of topics, or globally for the entire document. The metadata elements are authored within a <topicmeta> element, which associates metadata with the parent element and its children. Because the topics in a branch of the hierarchy typically have some common subjects or properties, this is a convenient mechanism to define properties for a set of topics. For example, the <topicmeta> element in a <relcolspec> can associate metadata with all the topics that are referenced in the <reltable> column.

A map can override or supplement everything about a topic except its primary title and body content. All the metadata elements that are available in a topic also are available in a map. In addition, a map may provide alternate titles and a short description. The alternate titles can override their equivalents in the topic. The short description in the map may override the short description in the topic if the <topicref> element specifies a @copy-to attribute.

DITA map: Attributes of the <topicref> element

At the map level, properties can be set as attributes of the <topicref> element.

DITA topic

Within a topic, authors can either set metadata attributes on the root element or add metadata elements in the <prolog> element.

How metadata set at both the map and topic level intersects

In a topic, the metadata elements apply to the entire topic. In a map, they supplement or override any metadata that is provided in the referenced topics. When the same metadata element or attribute is specified in both a map and a topic, by default the value in the map takes precedence; the assumption here is that the author of the map has more knowledge of the reusing context than the author of the topic. The @lockmeta attribute on the <topicmeta> element controls whether map-specified values override values in the referenced topic.

The <navtitle> element is an exception to the rule of how metadata specified by the <topicmeta> element is propagated. The content of the <navtitle> element is used as a navigation title only if the @locktitle attribute of the parent <topicref> element is set to "yes".

Associating attribute-based metadata with element-based metadata

Comment by Kristen James Eberlein , 14 July 2009
Do we need examples that illustrate what is discussed earlier in this topic? The extant example really illustrates an edge case, rather than the basic principles.

Comment by Kristen James Eberlein , 17 December 2009
Discussion from review #3:

• Bruce Nevin: I agree with Kris that other examples are needed.
• Kris Eberlein: If we want examples, people are going to need to pony up and develop them. This might need to be an item that is deferred to DITA 1.3.

At the topic level, the content of the prolog metadata elements can provide more information about the values that are used for attributes on the elements in the body of the DITA topic. However, prolog metadata and
attribute metadata also can be used and expressed independently. The coordination shown here is possible but is not required.

In the preceding example, the attribute value AdminNovice is associated with the `<audience>` element with the same name, which gives authors and processes more information about the audience in question: in this case, that the "AdminNovice" audience consists of administrators who are customizing and who are new at it.

### 3.2.3.4 Cascading of attributes and metadata in a DITA map

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

#### Attributes and metadata that cascade

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

**Attributes**

- `@audience`, `@platform`, `@product`, `@otherprops`, `@rev`
- `@props` and any attribute specialized from `@props`
- `@linking`, `@toc`, `@print`, `@search`
- `@format`, `@scope`, `@type`
- `@xml:lang`, `@dir`, `@translate`
- `@processing-role`
- `@cascade`

**Metadata elements**

- `<author>`, `<source>`, `<publisher>`, `<copyright>`, `<critdates>`, `<permissions>`
- `<audience>`, `<category>`, `<prodinfo>`, `<othermeta>`

Cascading is additive for attributes and metadata elements that accept multiple values, except when the `@cascade` attribute is set to avoid adding values. For attributes that take a single value, the closest value defined on a containing element takes effect. In a relationship table, row-level metadata is considered more specific than column-level metadata, as shown in the following containment hierarchy:

- `<map>` (most general)
  - `<topicref>` container (more specific)
    - `<topicref>` (most specific)
  - `<reltable>` (more specific)
    - `<relcolspec>` (more specific)
      - `<relrow>` (more specific)
      - `<topicref>` (most specific)
Merging of cascading attributes

The @cascade attribute can be used to modify the additive nature of attribute cascading (though it does turn off cascading altogether). The attribute has two predefined values: "merge" and "nomerge".

**cascade="merge"**

The metadata attributes cascade; the values of the metadata attributes are additive. This is the processing default for the @cascade attribute and was the only defined behavior for DITA 1.2 and earlier.

**cascade="nomerge"**

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

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

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

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

**Examples of the @cascade attribute in use**

Consider the following code examples:

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

**Figure 8: Map A**

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

**Figure 9: Map B**

For map A, the values for the attribute are merged, and the effective value of the @audience attribute for `topic.dita` is "a b c". For map B, the values for the attribute are not additive, and the effective value of the @audience attribute for `topic.dita` is "c".
In the following example, merging is active at the map level but turned off below:

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

**Figure 10: Map C**

In map C, the reference to `one.dita` has effective merged values of "a b" for `@platform` and "x y" for `@product`.

The reference to `two.dita` turns off merging, so the explicit `@product` value of "z" is used (it does not merge with ancestor values). The `@platform` attribute is not present, so the already-merged value of "a b" continues to cascade and the effective value of `@platform` on this reference.

### Order for processing cascading attributes in a map

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

For example, in the case of `<topicref toc="yes">`, applications **MUST** stop at 2 (see page 50) in the list; a value is specified for `@toc` in the document instance, so `@toc` values from containing elements will not cascade to that specific `<topicref>` element. The `toc="yes"` setting on that `<topicref>` element will cascade to contained elements, provided those elements reach 5 (see page 50) below when evaluating the `@toc` attribute.

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

1. The `@conref` and `@keyref` attributes are evaluated.
2. The explicit values specified in the document instance are evaluated. For example, a `<topicref>` element with the `@toc` attribute set to "no" will use that value.
3. The default or fixed attribute values are evaluated. For example, the `@toc` attribute on the `<reltable>` element has a default value of "no".
4. The default values that are supplied by a controlled values file are evaluated.
5. The attributes cascade.
6. The processing-supplied default values are applied.
7. After the attributes are resolved within the map, they cascade to referenced maps.

**Note:** The processing-supplied default values do not cascade to other maps. For example, most processors will supply a default value of `toc="yes"` when no `@toc` attribute is specified. However, a processor-supplied default of `toc="yes"` **MUST** not override a value of `toc="no"` that is set on a referenced map. If the `toc="yes"` value is explicitly specified, is given as a default through a DTD, XSD, RNG, or controlled values file, or cascades from a containing element in the map, it **MUST** override a `toc="no"` setting on the referenced map. See Map-to-map cascading behaviors (see page 51) for more details.

8. Repeat steps 1 (see page 50) to 4 (see page 50) for each referenced map.
9. The attributes cascade within each referenced map.
10. The processing-supplied default values are applied within each referenced map.
11. Repeat the process for maps referenced within the referenced maps.
Example of metadata elements cascading in a DITA map
The following code sample illustrates how an information architect can apply certain metadata to all the DITA topics in a map:

```xml
<map title="DITA maps" xml:lang="en-us">
  <topicmeta>
    <author>Kristen James Eberlein</author>
    <copyright>
      <copyryear year="2009"/>
      <copyrholder>OASIS</copyrholder>
    </copyright>
  </topicmeta>
  <topicref href="dita_maps.dita" navtitle="DITA maps">
    <topicref href="definition_ditamaps.dita" navtitle="Definition of DITA maps"></topicref>
    <topicref href="purpose_ditamaps.dita" navtitle="Purpose of DITA maps"></topicref>
  </topicref>
</map>
```

The author and copyright information cascades to each of the DITA topics referenced in the DITA map. When the DITA map is processed to XHTML, for example, each XHTML file contains the metadata information.

### 3.2.3.5 Map-to-map cascading behaviors

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

#### 3.2.3.5.1 Cascading of attributes from map to map

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

The following attributes cascade from map to map:

- `@audience`, `@platform`, `@product`, `@otherprops`, `@rev`
- `@props` and any attribute specialized from `@props`
- `@linking`, `@toc`, `@print`, `@search`
- `@type`
- `@translate`
- `@processing-role`
- `@cascade`

Comment by jelovirta

This list of attributes that cascade should be a complete list that requires no filtering. As it stands now and in DITA 1.2 spec, it’s a list of attributes followed by a second list of attributes that do not apply. When reader wants to get a complete list of attributes that cascade, they have to read the whole section, then take the first list and remove four attributes from it.

I suggest the attribute list be modified to remove the four non-cascading attributes and note elements be added to explain the rational why some attributes do now cascade.

Note that the above list excludes the following attributes:

- `@format`
The @format attribute must be set to "ditamap" in order to reference a map or a branch of a map, so it cannot cascade through to the referenced map.

@xml:lang and @dir

Cascading behavior for @xml:lang is defined in The xml:lang attribute (see page 107). The @dir attribute work the same way.

@scope

The value of the @scope attribute describes the map itself, rather than the content. When the @scope attribute is set to "external", it indicates that the referenced map itself is external and unavailable, so the value cannot cascade into that referenced map.

The @class attribute is used to determine the processing roles that cascade from map to map. See Cascading of roles from map to map (see page 53) for more information.

As with values that cascade within a map, the cascading is additive if the attribute permits multiple values (such as @audience). When the attribute only permits one value, the cascading value overrides the top-level element.

Example of attributes cascading between maps

Comment by robander, 3 April 2014

Stan commented in another topic, but I think the example would be post appropriate somewhere in here:

Question (or perhaps a candidate for a clarifying example) -- if I set cascade="merge" on a parent/root map and cascade="nomerge" on a child map, would the subordinate cascade="nomerge" setting be ignored?? Processing-supplied inheritance gets mentioned in the "list," but not cascading behavior/inheritance.

I'm not sure where best to include it, storing the comment here for the moment.

For example, assume the following references in test.ditamap:

```xml
<map>
  <topicref href="a.ditamap" format="ditamap" toc="no"/>
  <mapref href="b.ditamap" audience="developer"/>
  <topicref href="c.ditamap#branch1" format="ditamap" print="no"/>
  <mapref href="c.ditamap#branch2" platform="myPlatform"/>
</map>
```

• The map a.ditamap is treated as if toc="no" is specified on the root <map> element. This means that the topics that are referenced by a.ditamap do not appear in the navigation generated by test.ditamap (except for branches within the map that explicitly set toc="yes").

• The map b.ditamap is treated as if audience="developer" is set on the root <map> element. If the @audience attribute is already set on the root <map> element within b.ditamap, the value "developer" is added to any existing values.

• The element with id="branch1" within the map c.ditamap is treated as if print="no" is specified on that element. This means that the topics within the branch with id="branch1" do not appear in the printed output generated by test.ditamap (except for nested branches within that branch that explicitly set print="yes").

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

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

For a complete list of which elements cascade within a map, see the column "Does it cascade to child `<topicref>` elements?" in the topic Reconciling topic and map metadata (see page 54).

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

For example, consider the following code examples:

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

Figure 11: test-2.ditamap

```xml
<map>
  <topicmeta>
    <audience type="writer"/>
  </topicmeta>
  <topicref href="b-1.dita"/>
  <topicref href="b-2.dita"/>
</map>
```

Figure 12: b.ditamap

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

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

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

3.2.3.5.3 Cascading of roles from map to map

When specialized `<topicref>` elements (such as `<chapter>` or `<mapref>`) reference a map, they supply a role for the referenced content. Generally, the `@class` attribute of the referencing element cascades to the top-level `<topicref>` elements in the referenced map, but there are situations where this does not happen.

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

Unless otherwise instructed, a specialized <topicref> element that references a map supplies a role for the referenced content. This means that, in effect, the @class attribute of the referencing element cascades to top-level topicref elements in the referenced map. In situations where this should not happen - such as all elements from the mapgroup domain - the non-default behavior should be clearly specified.

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

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

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

**Example of cascading roles between maps**

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

**Referenced map contains a single top-level <topicref> element**

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

**Referenced map contains multiple top-level <topicref> elements**

Each top-level <topicref> element is processed as if it were a <chapter> element (the referencing element).

**Referenced map contains a single <appendix> element**

The <appendix> element is processed as it were a <chapter> element.

**Referenced map contains a single <part> element, with nested <chapter> elements.**

The <part> element is processed as it were a chapter element. Nested <chapter> elements might not be understandable by processors; applications MAY recover as described above.

**<chapter> element references a single <topicref> element rather than a map**

The referenced <topicref> element is processed as if it were a <chapter> element.

### 3.2.3.6 Reconciling topic and map metadata

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

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

**How does it apply to the topic?**
This column describes how the metadata specified within the `<topicmeta>` element interacts with the metadata specified in the topic. In most cases, the properties are additive. For example, when the `<audience>` element is set to "user" at the map level, the value "user" is added during processing to any audience metadata that is specified within the topic.

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

This column indicates whether the specified metadata value cascades to nested `<topicref>` elements. For example, when an `<audience>` element is set to "user" at the map level, all child `<topicref>` elements implicitly have an `<audience>` element set to "user" also. Elements which can apply only to the specific `<topicref>` element, such as `<linktext>`, do not cascade.

**What is the purpose when specified on the `<map>` element?**

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

### Table 2: Topicmeta elements and their properties

<table>
<thead>
<tr>
<th>Element</th>
<th>How does it apply to the topic?</th>
<th>Does it cascade to child <code>&lt;topicref&gt;</code> elements?</th>
<th>What is the purpose when set on the <code>&lt;map&gt;</code> element?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;audience&gt;</code></td>
<td>Add to the topic</td>
<td>Yes</td>
<td>Specify an audience for the entire map</td>
</tr>
<tr>
<td><code>&lt;author&gt;</code></td>
<td>Add to the topic</td>
<td>Yes</td>
<td>Specify an author for the entire map</td>
</tr>
<tr>
<td><code>&lt;category&gt;</code></td>
<td>Add to the topic</td>
<td>Yes</td>
<td>Specify a category for the entire map</td>
</tr>
<tr>
<td><code>&lt;copyright&gt;</code></td>
<td>Add to the topic</td>
<td>Yes</td>
<td>Specify a copyright for the entire map</td>
</tr>
<tr>
<td><code>&lt;critdates&gt;</code></td>
<td>Add to the topic</td>
<td>Yes</td>
<td>Specify critical dates for the entire map</td>
</tr>
<tr>
<td><code>&lt;data&gt;</code></td>
<td>Add to the topic</td>
<td>No, unless specialized for a purpose that cascades</td>
<td>No stated purpose, until the element is specified</td>
</tr>
<tr>
<td><code>&lt;data-about&gt;</code></td>
<td>Add the property to the specified target</td>
<td>No, unless specialized for a purpose that cascades</td>
<td>No stated purpose, until the element is specified</td>
</tr>
<tr>
<td><code>&lt;foreign&gt;</code></td>
<td>Add to the topic</td>
<td>No, unless specialized for a purpose that cascades</td>
<td>No stated purpose, until the element is specified</td>
</tr>
<tr>
<td><code>&lt;keywords&gt;</code></td>
<td>Add to the topic</td>
<td>No</td>
<td>No stated purpose</td>
</tr>
<tr>
<td><code>&lt;linktext&gt;</code></td>
<td>Not added to the topic; applies only to links created based on this occurrence in the map</td>
<td>No</td>
<td>No stated purpose</td>
</tr>
<tr>
<td><code>&lt;metadata&gt;</code></td>
<td>Add to the topic</td>
<td>Yes</td>
<td>Specify metadata for the entire map</td>
</tr>
<tr>
<td><code>&lt;navtitle&gt;</code></td>
<td>Not added to the topic; applies only to navigation that is created based on this occurrence in the map</td>
<td>No</td>
<td>No stated purpose</td>
</tr>
<tr>
<td>Element</td>
<td>How does it apply to the topic?</td>
<td>Does it cascade to child &lt;topicref&gt; elements?</td>
<td>What is the purpose when set on the &lt;map&gt; element?</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>map. The @locktitle attribute of the parent &lt;topicref&gt; element must be set to “yes” in order for the navigation title to be used.</td>
<td>No</td>
<td>Define metadata for the entire map</td>
<td></td>
</tr>
<tr>
<td>&lt;othermeta&gt;</td>
<td>Add to the topic</td>
<td>No</td>
<td>Specify permissions for the entire map</td>
</tr>
<tr>
<td>&lt;permissions&gt;</td>
<td>Add to the topic</td>
<td>Yes</td>
<td>Specify product info for the entire map</td>
</tr>
<tr>
<td>&lt;prodinfo&gt;</td>
<td>Add to the topic</td>
<td>Yes</td>
<td>Specify a publisher for the map</td>
</tr>
<tr>
<td>&lt;publisher&gt;</td>
<td>Add to the topic</td>
<td>Yes</td>
<td>Specify a resource ID for the map</td>
</tr>
<tr>
<td>&lt;resourceid&gt;</td>
<td>Add to the topic</td>
<td>No</td>
<td>No stated purpose</td>
</tr>
<tr>
<td>&lt;searchtitle&gt;</td>
<td>Replace the one in the topic. If multiple &lt;searchtitle&gt; elements are specified for a single target, processors may choose to issue a warning.</td>
<td>No</td>
<td>Provide a description of the map</td>
</tr>
<tr>
<td>&lt;shortdesc&gt;</td>
<td>Only added to the topic when the &lt;topicref&gt; element specifies a @copy-to attribute. Otherwise, it applies only to links created based on this occurrence in the map.</td>
<td>No</td>
<td>Provide a description of the map</td>
</tr>
<tr>
<td>&lt;source&gt;</td>
<td>Add to the topic</td>
<td>No</td>
<td>Specify a source for the map</td>
</tr>
<tr>
<td>&lt;unknown&gt;</td>
<td>Add to the topic</td>
<td>No, unless specialized for a purpose that cascades</td>
<td>No stated purpose, until the element is specified</td>
</tr>
</tbody>
</table>

3.3 DITA processing

Several DITA processing behaviors are driven by attributes, including setting the set of vocabulary and constraint modules on which a DITA document depends, navigation, linking, content reuse (using direct or
indirect addressing), conditional processing, chunking, and printing. In addition, translation of DITA content is expedited through the use of the @dir, @translate, and @xml:lang attributes, and the <index-sort-as> element.

### 3.3.1 Module compatibility and the @domains attribute

A DITA document declares, through the @domains attribute on map and topic elements, the set of vocabulary and constraint modules on which it depends.

Comment by Kristen Eberlein , 16 February 2014

Comment from Michael Priestley, 14 January: "Doesn't mention constraints at all. Should be updated to include mention of constraints, as well as intended behavior of loose validation by default, with processor option to turn on strict validation."

The @domains attribute serves two primary purposes:

- To indicate to DITA processors the specific features that they should or must provide in order to completely process the document.
- To determine the validity of elements that are copied from one DITA document to another. This copying might occur as the result of a content reference (conref) or key reference (keyref), or it might occur in the context of an author editing a DITA document.

A processor can examine the value of the @domains attribute and compare the set of modules listed to the set of modules for which it provides direct support. It then can take appropriate action if it does not provide support for a given module, for example, issuing a warning before applying fallback processing.

When copying content from one DITA document to another, processors need to determine if the data being copied (the copy source) requires modules that are not required by the document into which the data is to be copied (the copy target). Such a copy operation is always safe if the copy source requires a subset of the modules that are required by the copy target. Such a copy is unsafe if the copy source requires modules that are not required by the copy target.

When a copy operation is unsafe, processors may compare the copy source to the copy target to determine if the copy source satisfies the constraints of the copy target. If the copy source meets the copy target constraints, the copy operation can proceed. Processors SHOULD issue a warning that the copy was allowed but the constraints are not compatible. If the copy source does not meet the constraints of the copy target, processors may apply generalization until the generalized result either satisfies the copy target constraints or no further generalization can be performed. If the copy operation can be performed following generalization, the processor SHOULD issue a warning that the constraints are not compatible and generalization had to be performed in order to complete the copy operation.

### 3.3.2 Navigation

DITA includes markup that processors can use to generate reader navigation to or across DITA topics. Such navigation behaviors include table of contents (TOCs) and indexes.

#### 3.3.2.1 Table of contents

Processors can generate a table of contents (TOC) based on the hierarchy of the elements in the DITA map. By default, each <topicref> element in the map represents a node in the TOC. These topic references define a navigation tree.

When a map contains a topic reference to a map (often called a map reference), processors should integrate the navigation tree of the referenced map with the navigation tree of the referencing map at the point of reference. In this way, a deliverable can be compiled from multiple DITA maps.

Comment by Kristen Eberlein , 29 January 2014
Can we simplify or clarify the following phrase: "processors should integrate the navigation tree of the referenced map with the navigation tree of the referencing map at the point of reference."

**Note:** If a `<topicref>` element that references a map contains child `<topicref>` elements, the processing behavior regarding the child `<topicref>` elements is undefined.

By default, processors obtain the text for each TOC node from the title of the referenced topic. If the `@locktitle` attribute on the `<topicref>` element is set to "yes", the node text is taken from the `@navtitle` attribute or the `<navtitle>` element; it must not be read from the title of the referenced topic. If a `<topicref>` element contains both a `<navtitle>` child element and a `@navtitle` attribute, the `@locktitle` attribute applies to both `<navtitle>` and `@navtitle` and, when set to "yes", the value of the `<navtitle>` element must be used.

**Comment by Kristen Eberlein , 29 January 2014**
I'm a little at a loss by the final sentence. I think it could be simplified as "If a `<topicref>` element contains both a `<navtitle>` child element and a `@navtitle` attribute, and the `@locktitle` attribute is set to "yes", the value of the `<navtitle>` element must be used."

A TOC node is generated for every `<topicref>` element that references a topic or specifies a navigation title, except in the following cases:

- The `@processing-role` attribute that is specified on the `<topicref>` element or an ancestor element is set to "resource-only".
- The `@print` attribute is specified on the `<topicref>` element or an ancestor element, and the current processing does not match the value set by the `@print` attribute. For example, `print="printonly"` and the output format is XHTML-based, or `print="no"` and the output format is PDF.
- Conditional processing is used to filter out the node or an ancestor node.
- There is no information from which a TOC entry can be constructed; there is no `@href` attribute or `@navtitle` attribute or `<navtitle>` element.
- The node is a `<topicgroup>` element.

To suppress a `<topicref>` element from appearing in the TOC, set the `@toc` attribute to "no". The value of the `@toc` attribute cascades to child `<topicref>` elements, so if `@toc` is set to "no" on a particular `<topicref>`, all children of the `<topicref>` element are also excluded from the TOC. If a child `<topicref>` overrides the cascading operation by specifying `toc="yes"`, then the node that specifies `toc="yes"` appears in the TOC (minus the intermediate nodes that set `@toc` to "no").

### 3.3.2.2 Indexes

An index can be generated from index entries that occur in topic bodies, topic prologs, or DITA maps.

For more information, see indexterm (see page 336).

**Comment by Kristen Eberlein , 8 February 2014**
We need more information here, at the least an overview of the indexing domain. There is content in the related Language Reference topics that should examined; we should decided whether some of that content should be migrated to here or repeated here.

### 3.3.3 DITA linking

DITA depends heavily on links. The purposes for which it provides links include defining the content and organization of publication structures (DITA maps), topic-to-topic navigation links and cross references, and reuse of content by reference. All DITA links use the same addressing mechanisms, either URI-based addresses or indirect addresses using keys and key references.

**Comment by Kristen Eberlein , 6 October 2014**
This topic is WAY too long. The content needs to be moved into a child topic.
At its most general, a link establishes a relationship among two or more objects. In DITA, relationships are among DITA elements and either other DITA elements or non-DITA resources, such as Web pages. Relationships may be explicitly typed in some cases (relationship tables and subject scheme maps for example) but are not always associated with a specific relationship type.

**Note:** For example, a `<keyword>` element that uses a key reference to link to the definition of the keyword can be considered to be establishing a "mention-of" relationship from the `<keyword>` element to the definition and a "definition-of" relationship from the definition to the `<keyword>` element. But those link types are not formally defined either in the DITA definition of `<keyword>` or in the markup for the `<keyword>` element itself. While DITA enables the formal definition of typed relationships for some types of link elements, it does not require that all links be formally typed and does not provide a general mechanism for associating explicit link types with links.

In the abstract, link relationships may be explicit, defined directly by some type of markup in the source data, or implicit, implied by properties of the content that a processor uses to infer relationships (for example, matching the content of a `<keyword>` element to the title of a topic of a specific topic type). DITA formally defines only explicit links, although processors MAY implement implicit links.

A link may establish either a navigation relationship or a use-by-reference relationship (e.g., content references). Navigation relationships are used primarily to enable navigation from one element to another, although they may also be used for other purposes, such as classification, or association of metadata. Use-by-reference relationships establish the effective structure and content of the information set.

An element that establishes one or more such relationships is a "link-defining element". Some element types, such as `<link>` and `<xref>`, are always link-defining elements. Other element types become link-defining elements when they use specific link-defining attributes.

Almost any element may become a use-by-reference link by using the `@conref` or `@conkeyref` attribute to establish a content reference (conref) relationship to another element or set of elements (see Use by Reference). Elements such as `<term>` and `<keyword>` may become navigation links by using the `@keyref` attribute to establish a relationship to another DITA element or non-DITA resource.

In general, elements within topics that take both the `@href` and `@keyref` attributes always act as elements that define a navigation link, while elements that take `@keyref` but not `@href` act as elements that define a navigation link only when they specify `@keyref`.

A given link-defining element may establish more than one relationship. For example, an element may establish both a content reference link and a navigation link. A single row in a relationship table may establish a number of distinct relationships among the topics referenced in the different cells of the relationship table. A topic reference within a hierarchy of topic references establishes not only a use-by-reference relationship from the map to the topic, but also hierarchical relationships from the referenced topic to other topics in the navigation hierarchy (parents, siblings, and children).

DITA defines two forms of addresses for use in defining links, direct URI-based addresses and indirect key references. In all cases, the nature of the relationships established is independent of the form of address used. For example, a cross reference that uses a key reference to address the target of the cross reference is functionally equivalent to having addressed the same target by URI reference, in that the final processing result should be the same in both cases. However, the two forms of address have different practical and intermediate processing implications. See DITA addressing.

Links from maps to other maps, topics, or non-DITA resources establish explicit dependencies from the map containing the links to the associated resources. Links from maps to maps create a "map tree". The set of dependencies for a root map is the union of the dependencies of all the maps in the map tree.

Links from a topic to other topics, maps, or non-DITA resources establish explicit dependencies from the topic containing the links to the associated resources, and implicit dependencies from any maps that use that linking topic to its dependencies.
For the purposes of determining the set of dependencies for a given map tree, processors MAY ignore any implicit dependencies created by links within topics that are not also established by explicit dependencies in the map tree. In the case where a map includes a topic that includes a topic-to-topic link, where the linked topic is not explicitly included in the map, and the processor considers only dependencies that are explicitly defined in the map, the processor may fail to resolve the topic-to-topic link. This case can be avoided by using a resource-only topic reference in the map tree to establish the dependency explicitly. If the resource-only topicref also defines a key, the link within the topic can then be changed to use a key reference (@keyref or @conkeyref) instead of a URI reference (@href, @conref). See Key-based addressing.

Navigation links have an associated "scope" indicating the closeness of the relationship of the linking element to associated resources. See The scope attribute (see page 677).

Most navigation links have an associated "link text", which is the text used to render the link so that it can be used. For all elements that allow or require link text, the link text may be specified as part of the linking element or, if unspecified, should be taken from the referenced resource. The details for how the link text for a given element should or may be generated are defined for that element type and may also be determined entirely by a rendition processor.

In the specific case of cross references created using <xref> and related links using <link>, the potential set of rules for constructing link text is essentially unbounded. Processors MAY, for example, define conventions for the value of @outputclass by which authors can indicate the details of how the link text should be constructed, or they may provide appropriate configuration options for controlling or customizing the construction of link text in cross references.

### 3.3.3.1 Links within maps

DITA maps serve primarily to define a navigation hierarchy of topics and non-DITA resources. Through relationship tables, maps may also define arbitrary topic-to-topic relationships such as "related links". Maps may also link to topics or non-DITA resources to establish dependency relationships without binding the linked resource into the navigation tree.

By default, the topic references within a map but not within a relationship table establish a navigation tree rooted at the root map within a map tree. A topic reference contributes to the navigation tree when it specifies a navigation title or references a topic or non-DITA resource. The @collection-type attribute of the <topicref> element determines the relationships established between the topicref and its parent, sibling, and child topicrefs, as well as among its child topicrefs.

A <topicref> or <navref> element that references a map does not bind the map to the navigation tree but acts as a form of use-by-reference link to the direct subelements of <map> and the relationship tables of the referenced map.

Maps may also contain relationship tables (<reltable>). Relationship tables establish navigation links among sets of topics and non-DITA resources. A given relationship table defines one or more links of a specific relationship type. See reltable (see page 306). A map may include any number of relationship tables. Within a map tree, the effective set of relationship tables is the union of all the relationship tables in all the maps in the map tree.

Topic references that specify a @processing-role value of "resource-only" establish dependencies from the map to the associated resource but do not bind the resource to the navigation tree. Resource-only topic references are typically used for key definitions where the key is not intended to represent a specific navigation tree location and for topics that hold elements used only for content reference or that otherwise should not be reflected in the navigation tree.

Topic references in the navigation tree can further control whether or not they are included in tables of contents using the @toc attribute. A topic reference that specifies "no" for the @toc attribute and is not a resource-only topic reference still contributes to the navigation tree. In particular, any relationships determined by the value of the @collection-type attribute are created.
Topic references in the navigation tree can use the @linking attribute to control how links created by the effective @collection-type value apply to the topic reference’s associated resource. See Attributes common to many map elements (see page 655).

Within maps, subordinate maps may be linked in either of two ways:

- `<topicref>` with a @format value of "ditamap" (this type of map reference is sometimes referred to as a "mapref")
- `<navref>`

The `<navref>` element links to an otherwise independent map and indicates that the integration of that map’s navigation structure into the larger navigation tree is deferred so that it can be performed as a final step in any delivery of the rendered content. Maps referenced by `<navref>` do not contribute to the key space of the map tree from which they are referenced. The map referenced by `<navref>` need not be available for processing at the time the referencing map is processed.

### 3.3.3.2 Links between maps

When using scoped keys, one map can refer to keys defined in a different root map, creating peer map-to-map links. These map-to-map links enable the production of deliverables with working links to other deliverables (cross-deliverable links).

Comment by Kristen Eberlein, 5 October 2014

I suspect that this content needs to be reworked to conform to the new key terminology and the coverage of keys in the "Indirect key-based addressing" section.

When a key scope is defined on a topic reference to a DITA map and the value of @scope is "peer", then the referenced map is a peer map with respect to the map making the reference. The implication of a scope of "peer" in this case is that the target map is managed along with the referencing map such that the author or processor of the first map likely has access to the referenced map as well.

Keys defined by the peer map belong to the key scope declared on the topicref element that references that map. Such keys can be referenced from content in the referencing map using the standard scope-qualified key names used for any other key scope (that is, one of the scope names, followed by a period, followed by the key name). However, references to keys defined in peer maps are processed differently from references to keys defined in the local map structure.

For example, given the map documents `map-a.dita` and `map-b.dita`, Map A can treat Map B as a peer root map using the key definition:

```xml
<map>
  <title>Map A</title>
  <topicref
    scope="peer"
    format="ditamap"
    keyscope="map-b"
    href="../map-b/map-b.ditamap"
    processing-role="resource-only"
  />
</map>
```

In this example, `map-b.ditamap` is not a submap of Map A; it is a peer map.

Conforming DITA processors are not required to support the resolution of or delivery of key references to peer maps. Note, however, that the general implication of peer scope is that the peer resources are all available in the same processing or management context. This means that processors always have the potential to resolve peer key references when the peer resources are in fact managed together. There may, of course, be performance, scale, and user interface challenges in implementing such systems, but the fundamental ability to resolve any given reference is ensured when the source files are physically accessible. That is, given a correctly-specified
reference to a peer map, if the map itself is accessible, it must then be possible to process that map in order to resolve keys defined by it, if it is also possible to resolve keys in the map making the peer map reference.

Note also the inverse implication: if the peer resource is not available, then it is impossible to resolve the key reference. For that reason, processors should provide appropriate messages when a peer resource cannot be resolved. Depending on how a set of DITA resources is authored, managed, and processed, peer resources may be resolvable or not at different points in the content life cycle.

The referenced peer map may specify @keyscope on its root element. In that case, the @keyscope on the peer map is ignored for the purpose of resolving scoped key references from the referencing map. This avoids the need for processors to have access to or examine the peer map before being able to determine that a given key reference is or is not in a peer scope.

For example, given this map reference in map Map A:

```
<mapref
  keyscope="scope-b"
  scope="peer"
  href="map-b.ditamap"
/>
```

Where map-b.ditamap is:

```
<map keyscope="scopeB">
  ...
</map>
```

For references from the context of Map A, key references of the form "scope-b.somekey" will be resolved to keys defined in the global scope of map B, but key references of the form "scopeB.somekey" will not. The presence or absence of a @keyscope attribute on Map B's <map> element has no effect in this case. Another way to view this is that a key reference to the scope "scope-b.somekey" is equivalent to the unscoped reference "somekey" processed in the context of Map B as the root map. In both cases, the presence of @keyscope on Map B's <map> element has no effect, in the first case (peer key reference) because it is explicitly ignored, in the second because the key reference will necessarily be within the scope "scope-b" and therefore does not need to be scope qualified.

For local-scope map references, the @keyscope attribute values on the map reference and the map element are merged, meaning that both the scopes named on the map reference and on the map will be resolvable for key references processed in the context of the referencing parent root map. The referenced map does not create another level of scope hierarchy when referenced as a sub map.

Note: This rule ensures that the same key reference (e.g., "scope-b.somekey") will resolve to the same resource whether the referenced map is referenced as a local map (submap) or a peer map.

### 3.3.3.3 Links within topics

A topic may contain several types of links.

- Content reference links from any element in the topic that allows @conref or @conkeyref.
- Related information links, within a <related-links> element following the topic body. The related links are usually rendered at the end of the topic.
- Image links created using <image>. Image elements may use <longdesc> to link to the long description for the image as a supplement to the <alt> element.
- Object links created using <object>. Object elements may use <longdesc> to link to the long description for the object as a supplement to the <alt> element.
- Navigation links created using <xref>. For output media that support hyperlinking, the <xref> should result in a hyperlink.
• Navigation links created using @keyref on elements that allow @keyref but not @href (e.g., <ph>, <cite>, <keyword>, and <term>).
• Metadata associations using <data-about> in contexts where <data> is allowed.
• Navigation links from long quotes to the source of the quote using <longquoteref>.

Links to resources outside a topic's containing XML document that use direct URI-based addresses establish unconditional topic-to-resource dependencies. Such dependencies can impede reuse in two ways:
• The linking topic cannot be used in a given map unless the dependent resource is also used.
• The linked resource cannot be dynamically changed based on the map context in which the linking topic is used.

These issues can be avoided by using key-based addressing. Because keys are defined in maps, each map that uses the linking topic can bind the key to the most appropriate resource.

3.3.4 Direct DITA addressing

DITA provides two addressing mechanisms. DITA addresses either are direct, URI-based addresses, or they are indirect key-based addresses. Within DITA documents, individual elements are addressed by unique identifiers specified on the @id attribute. DITA defines two fragment identifier syntaxes for addressing DITA elements, one for topics and elements within maps and another for non-topic elements within topics.

3.3.4.1 ID attribute

The @id attribute assigns an identifier to DITA elements so that the elements can be referenced.

The @id attribute is available for most elements. The @id attribute is required on some elements. For an element to be referenced, it must have an @id attribute with a valid value, although entire maps and the first topic, only topic, or all direct-child topics (depending on processing context) in a topic-containing document can be referenced without using an ID. The requirements for the @id attribute differ depending on whether it is used on a topic element, a map element, or an element within a topic or map.

All values for the @id attribute must be XML name tokens.

The @id attributes for topic and map elements are true XML IDs; therefore, they must be unique with respect to other XML IDs within the XML document that contains the topic or map element. The @id attribute for most other elements within topics and maps are not declared to be XML IDs; this means that XML parsers do not require that the values of those attributes be unique.

Within documents that contain multiple topics, the values of the @id attribute for all non-topic elements that have the same nearest-ancestor- topic element should be unique with respect to each other. The values of the @id attribute for non-topic elements can be the same as non-topic elements with different nearest-ancestor-topic elements. Therefore, within a single DITA document that contains multiple peer or nested topics, the values of the @id attribute of the non-topic elements need only to be unique within each topic.

Within a map document, the values of the @id attributes for all elements should be unique. When two elements within a map have the same value for the @id attribute, processors MUST resolve references to that ID to the first element with the given ID value in document order.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attribute type</th>
<th>Unique within</th>
<th>Required?</th>
<th>Value type</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;map&gt;</td>
<td>ID</td>
<td>document</td>
<td>No</td>
<td>XML non-colonized name token</td>
</tr>
<tr>
<td>&lt;topic&gt;</td>
<td>ID</td>
<td>document</td>
<td>Yes</td>
<td>XML non-colonized name token</td>
</tr>
</tbody>
</table>
## 3.3.4.2 URI-based (direct) addressing

Content reference and link relationships can be established from DITA elements using URI references to point directly to targets.

URI references address "resources" and, optionally, subcomponents of those resources. In the context of DITA, a resource is a DITA document (map, topic, or DITA base document) or a non-DITA resource (e.g., a Web page, a PDF document, etc.). For DITA resources, fragment identifiers can be used with the URI to address individual elements. The fragment identifier is the part of the URI that starts with a number sign ("#"), e.g., "#topicId/elementId". URI references may also include a query component, introduced with "?". DITA processors MAY ignore queries on URI references to DITA resources.

**Note:** URI references that are URLs must conform to the rules for URLs and URIs. In particular, Windows paths with backslashes are not valid URLs.

### URIs and DITA fragment identifiers

DITA uses URI references in @href, @conref, or other attributes for all direct addressing of resources.

For addressing DITA elements within maps and topics or individual topics within documents containing multiple topics, URI references must include the appropriate DITA-defined fragment identifier. URI references may be relative or absolute. A relative URI reference may consist of just a fragment identifier. Such a reference is a reference to the document that contains the reference.

When addressing a DITA topic element, URI references may include a fragment identifier that includes the ID of the topic element (filename.dita#topicId or #topicId). When addressing the DITA topic element that contains the URI reference, the URI reference may include the same topic fragment identifier of '.' (\#). When addressing a non-topic element within a DITA topic, a URI reference must use a fragment identifier that contains the ID of the ancestor topic element of the non-topic element being referenced, a slash ("\"), and the ID of the non-topic element (filename.dita#topicId/elementId or #topicId/elementId). When addressing a non-topic element within the topic that contains the URI reference the URI reference may include the same topic fragment identifier of '.' (\#/elementId).

This addressing model makes it possible to reliably address elements whose id attribute values are unique within a single DITA topic but which may not be unique within a larger XML document that contains multiple DITA topics. (See **ID attribute** (see page 63) for more information on ID attributes.)

When addressing a DITA map element, URI references may include a fragment identifier that includes the ID of the map element (filename.ditamap#mapId or #mapId). The same topic URI reference fragment identifier of '.' may not be used in DITA map elements.

If a target DITA element is within the same XML document as the element making the reference, the URI reference can consist of only the fragment identifier (including the "#" (number sign) character).

### Addressing non-DITA targets via URI

All resources, regardless of type, are directly addressed by URI references from DITA elements. Any fragment identifier used must conform to the fragment identifier requirements that defined for the target media type.
**Addressing DITA topics via URI**

Topics can always be addressed by a URI reference whose fragment identifier consists of the topic's ID. For the purposes of linking, a reference to a topic-containing document addresses the first topic within that document in document order. For the purposes of rendering, a reference to a topic-containing document addresses the root element of the document.

**Note:** For example, given a document whose root element is a topic, a URI reference (with no fragment identifier) addressing that document implicitly references the topic element. Given a `<dita>` document containing multiple topics, a URI reference addressing the `<dita>` document implicitly addresses the first child topic of the `<dita>` element for purposes of linking (for example, from a cross reference element) but addresses the `<dita>` element for the purposes of rendering (implying that all the topics contained by the `<dita>` element will be rendered in the result).

**Addressing non-topic DITA elements via URI**

To address non-topic elements within topics via URI, a topicID/elementID fragment identifier must be used.

To address elements within a DITA map via URI, an elementID fragment identifier must be used. The linking element must specify a value of "ditamap" for the `@format` attribute.

**URI reference syntax examples**

The following table shows the URI syntax for common use cases.

<table>
<thead>
<tr>
<th>Use case</th>
<th>Sample syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target a table in a topic at a network location</td>
<td>&quot;<a href="http://example.com/file.dita#topicID/tableID">http://example.com/file.dita#topicID/tableID</a>&quot;</td>
</tr>
<tr>
<td>Target a section in a topic on a local file system</td>
<td>&quot;directory/file.dita#topicID/sectionID&quot;</td>
</tr>
<tr>
<td>Target a figure contained in the same XML document</td>
<td>&quot;#topicID/figureID&quot;</td>
</tr>
<tr>
<td>Target a figure contained in the same topic of an XML document</td>
<td>&quot;#./figureID&quot;</td>
</tr>
<tr>
<td>Target an element within an map</td>
<td>&quot;<a href="http://example.com/map.ditamap#elementID">http://example.com/map.ditamap#elementID</a>&quot; (and a value of &quot;ditamap&quot; for the <code>@format</code> attribute)</td>
</tr>
<tr>
<td>Target a map element within the same map document</td>
<td>&quot;#elementID&quot; (and a value of &quot;ditamap&quot; for the <code>@format</code> attribute)</td>
</tr>
<tr>
<td>Reference an external Web site</td>
<td>&quot;<a href="http://www.somesite.com%22,%22http://www.somesite.com#somefragment">http://www.somesite.com&quot;,&quot;http://www.somesite.com#somefragment</a>&quot; or any other valid URI</td>
</tr>
<tr>
<td>Reference an element within a local map</td>
<td>&quot;filename.ditamap#elementid&quot; (and a value of &quot;ditamap&quot; for the <code>@format</code> attribute)</td>
</tr>
<tr>
<td>Reference a local map</td>
<td>&quot;filename.ditamap&quot; (and a value of &quot;ditamap&quot; for the <code>@format</code> attribute)</td>
</tr>
<tr>
<td>Reference a local topic</td>
<td>Reference a local topic &quot;filename.dita&quot; or &quot;path/ filename.dita&quot;</td>
</tr>
<tr>
<td>Reference a specific topic in a local document</td>
<td>&quot;filename.dita#topicid&quot; or &quot;path/ filename.dita#topicid&quot;</td>
</tr>
<tr>
<td>Reference a specific topic in the same file</td>
<td>&quot;#topicid&quot;</td>
</tr>
</tbody>
</table>
3.3.5 Indirect key-based addressing

DITA allows references to be made indirectly by referencing a name for a link target, instead of referencing the location of that target. This name is called a key. The DITA key-reference mechanism provides a layer of abstraction so that resource locations can be defined globally at the DITA map level instead of locally in each topic.

When using DITA topics in the context of different maps, it is often necessary to have relationships resolve to different resources. For example, a content reference to a `<keyword>` element that contains a product name might need to resolve to a different `<keyword>` element when used in a different product-specific map. The DITA key-reference mechanism provides an indirect addressing mechanism that separates references (topicrefs, conrefs, cross references, etc.) from the direct address of the target. (A direct address is the address specified on the element that references the key, for example, through the `@href` or `@conref` attribute.) Linking elements can refer to key names; the key names then are bound to specific resources by maps. Different maps can bind the same key names to different resources. The binding of key names to resources is determined at processing time based on the current set of key definitions for the map context, rather than from a static binding that is created when a topic or map is authored.

3.3.5.1 Core concepts for working with keys

The concepts described below are critical for a full understanding of keys and key processing.

Core definitions related to keys

key
A name for a resource.

key definition
A `<topicref>` element or specialization that binds one or more key names to resources.

key reference
An attribute that references a key, such as `@keyref` or `@conkeyref`.

Comment by chris.nitchie
Look for places where we talk about elements as key references to make sure we're being consistent.

key space
A list of key definitions that are used to resolve key references.

effective key definition
The definition for a key within a key space that is used to resolve references to that key. A key might have multiple definitions within a key space, but only one of those definitions is effective.

key scope
A map or section of a map that defines its own key space and serves as the resolution context for its key references.

Key definitions
A key definition binds the key to a number of resources:

- Any content referenced directly by the `@href` attribute or indirectly by the `@keyref` attribute. References to the key definition are considered references to this content.
• The contents of any `<topicmeta>` child element. Those contents can be used to populate the content of elements that reference the definition.

If a key definition does not contain a `<topicmeta>` element, and does not reference a resource by `@href` or `@keyref`, it is nonetheless a valid key definition. References to the key definition are considered resolvable, but no linking or content population occurs.

**Key scopes**

All key definitions and key references exist within a key scope. Within a map hierarchy, key scopes are bounded by

• The root map.
• The root element of submaps whose root elements specify the `@keyscope` attribute.
• Any `<topicref>` elements or `<topicref>` specializations that specify the `@keyscope` attribute.

The `@keyscope` attribute declares a name or names for the scope. If `@keyscope` is specified both on the root element in a map and on a `<topicref>` element that references that map, only one scope is introduced; its names are the union of the names that are specified by both `@keyscope` attributes.

**Key spaces**

A key scope is associated with exactly one key space. That key space contains all key definitions that are located directly within the scope; it may also contain definitions that exist in other scopes. Specifically, a key scope's key space is comprised of the following key definitions, in order of precedence:

1. All key definitions from the parent key scope's key space, if any.
2. Key definitions within the scope-defining element, including those defined in directly-addressed, locally-scoped submaps, but excluding those defined in child scopes.
3. The key definitions from child scopes, with each key prepended by the child scope name followed by a period. If a child scope has multiple names, the keys in that scope are addressable from the parent scope using any of the scope names as a prefix.

**Note:** Because of rules 1 and 3, the key space associated with a child scope includes the scope-qualified copies of its own keys that are inherited from the parent scope's key space, as well as those from other "sibling" scopes.

The key space associated with a key scope is used to resolve all key references that occur immediately within that scope. Key references in child scopes are resolved using the key spaces that are associated with those child scopes.

**Effective key definitions**

A key space can contain many definitions for a given key, but only one definition is effective for the purpose of resolving key references.

When a key has a definition in a parent scope's key space, that definition is ceffective. Otherwise, a key definition is effective if it is first in a breadth-first traversal of the locally-scoped submaps beneath the scope-defining element. Put another way, a key definition is effective if it is the first definition in the shallowest map for that key. This allows higher-level map authors to override keys defined in referenced submaps.

**Note:** A key definition that specifies more than one key name in its `@keys` attribute may be the effective definition for some of its keys but not for others.
Within a key scope, keys do not have to be defined before they are referenced. The key space is effective for the entire scope, so the order of key definitions and key references relative to one another is not significant. This means that all key spaces for a root map must be determined before any key reference processing can be performed, and that maps referenced solely by key reference have no bearing on key space contents.

For purposes of key definition precedence, the scope-qualified key definitions from a child scope are considered to occur at the location of the scope-defining element within the parent scope.

For purposes of key definition precedence, the scope-qualified key definitions from a child scope are considered to occur at the location of the scope-defining element within the parent scope.

Comment by chris.nitchie, 7/25/14
This is controversial and needs more thought.

map
  sub-map scope="scopeA"
  sub-map scope="scopeB"
    keydef="MYKEY"
  sub-map scope="scopeA.scopeB"
    keydef MYKEY
  key scopeA.scopeB.MYKEY=?

Per traditional rules, the answer is the second definition; per this sentence, it's the first, violating existing stuff.

3.3.5.2 Using keys to address DITA elements

For topic references, image references, and navigation link relationships (<link>, <xref>, and elements that take the @keyref but not the @href attribute), resources can be addressed by key using the @keyref attribute. For content reference relationships, resources can be addressed by key using the @conkeyref attribute.

Syntax

For references to topics, maps, and non-DITA resources, the value of the @keyref attribute is simply a key name (for example, keyref="topic-key").

For references to non-topic elements within topics and non-topicref elements within maps, the value of the @keyref attribute is a key name, a slash (/), and the ID of the target element: keyref="topic-key/some-element-id".

If both @keyref and @href attributes are specified on an element, the @href value MUST be used as a fallback address when the key name is undefined. If both @conkeyref and @conref attributes are specified on an element, the @conref value MUST be used as a fallback address when the key name is undefined.

Comment by robander, September 16, 2014
After discussing with Chris Nitchie, I've removed 2 copies of a clause from the previous paragraph, highlighted below in the original version:

If both @keyref and @href attributes are specified on an element, the @href value MUST be used as a fallback address when the key name is undefined, and SHOULD be used as a fallback address when the key name is defined but the key reference cannot be resolved to a resource. If both @conkeyref and @conref attributes are specified on an element, the @conref value MUST be used as a fallback address when the key name is undefined, and SHOULD be used as a fallback address when the key name is defined but the key reference cannot be resolved to a resource.

In our reading, the portion about a key being defined but not resolved conflicts with other language about processing, which for example states that key definitions without a link target or with linking="none" result in no link. That language is clear and conflicts with the removed language here, which suggested that given a valid key definition with no link target, processors should use the local @href as fallback.

The language could also have been interpreted as "If the resource referenced by a key definition does not exist, @href SHOULD be used as fallback", which does not seem like something we can or should enforce. If the key definition points to example.com, but that server is down, an application would seemingly have to 1) know this, 2) assume it does not exist, and 3) switch to using @href.
Example

For example, consider this topic in the document file.dita:

```xml
<topic id="topicid">
  <title>Example referenced topic</title>
  <body>
    <p id="para-01">Some content.</p>
  </body>
</topic>
```

and this key definition:

```xml
<map>
  <topicref keys="myexample" href="file.dita"/>
</map>
```

A keyref of the form "myexample/para-01 resolves to the <p> element in the topic. The key reference would be equivalent, in the context of this map, to the URI reference file.dita#topicid/para-01.

A key reference to a <topicref> element where the linking element specifies a format value of "ditamap" addresses the <topicref> element itself as though the <topicref> element had been addressed by ID. In particular, a <topicref> with a key reference to another <topicref> and a format value of "ditamap" is a use of the map branch rooted at the referenced <topicref>.

### 3.3.5.3 Processing key references

Key references can resolve as links, as text, or as both. Within a map, they can also be used to create or supplement information on a topic reference. This topic covers information that is common to all key processing, regardless of how the key is used.

#### Keys and conditional processing

The effective key definitions for a key space might be affected by conditional processing (filtering). Processors must perform conditional processing before determining the effective definition for a key reference. However, processors might determine key space contents before filtering. Consequently, different processors might produce different effective bindings for the same map when there are key definitions that might be filtered out based on their filtering attributes.

#### Reusing a topic in multiple key scopes

If a topic that contains key references is reused in multiple key scopes within a given root map such that its references resolve differently at each use context, processors MUST produce multiple copies of the source topic in resolved output for each distinct set of effective key definitions that are referenced by the topic. In such cases, authors can use the @copy-to attribute to control output generation.

#### Keys in peer root maps

For references to keys in peer root maps, if the peer map is unavailable for processing or the key reference cannot otherwise be resolved, the key reference is processed as though there was no key reference. If the peer map is available, the key reference is processed normally.

#### Error conditions

It is an error for an empty element to have a key reference with an undefined key and have no @href attribute for fallback. In this case, an implementation MAY give an error message, and MAY recover from this error condition by leaving the key reference element empty.
3.3.5.4 Processing key references on <topicref> elements

While <topicref> elements are used to define keys, they also can reference keys defined elsewhere. This topic explains how to evaluate key references on <topicref> and its specializations.

For topic references that use the @keyref attribute, the effective value of the <topicref> element is determined in the following way:

Comment by Kristen Eberlein
Might want to change the <ul> to <dl>, with appropriate headers.

- The effective resource bound to the <topicref> element is determined by resolving all intermediate key references. Each key reference is resolved either to a resource addressed directly by URI reference in an @href attribute, or to no resource. Processors MAY impose reasonable limits on the number of intermediate key references that they will resolve. Processors SHOULD support at least three levels of key references.

  Note: This rule applies to all topic references, including those that define keys. The effective bound resource for a key definition that uses the @keyref attribute cannot be determined until the key space has been constructed.

The attributes that are common to a key definition element and a key reference element using that key, other than the @keys, @processing-role, and @id attributes, are combined as for content references, including the special processing for the @xml:lang, @dir, and @translate attributes. There is no special processing associated with either the @locktitle or the @lockmeta attributes when attributes are combined.

- Content from a key reference element and a key-defining element is combined following the rules for combining metadata between maps and other maps and between maps and topics. The @lockmeta attribute is honored when metadata content is combined.

- The combined attributes and content cascade from one map to another or from a map to a topic, but this is controlled by existing rules for cascading, which are not affected by the use of key references.

If, in addition to the @keys attribute, a key definition specifies a @keyref attribute that can be resolved after the key resolution context for the key definition has been determined, the resources bound to the referenced key definition take precedence.

Comment by robander
NEEDS EXAMPLE OF THIS PARAGRAPH

3.3.5.5 Processing key references to create or redirect links

Keys can be used to create or redirect links and cross references. This topic explains how to evaluate key references on links and cross references to determine a link target.

When a key definition is bound to a resource that is addressed by @href or @keyref and does not specify "none" for the @linking attribute, all references to that key definition become navigation links to the bound resource.

When a key definition is not bound to a resource or specifies "none" for the @linking attribute, references to that key do not become navigation links.

When a key definition has no @href value and no @keyref value, references to that key will not result in a link, even if they do contain an @href attribute of their own. If the key definition also does not contain a <topicmeta> subelement, empty elements that refer to the key (such as <link keyref="a"/> or <xref keyref="a" href="fallback.dita"/> are removed.

If a referencing element specifies a key reference to an undefined key, it is processed as if there were no key reference, and the value of the @href attribute is used as the reference. If the @href attribute is not specified either, the element is not treated as a navigation link.
3.3.5.6 Processing key references to generate text or link text

Key references can be used to pull text from the key definition. This topic explains how to generate text from a key definition, regardless of whether the key reference also results in a link.

Note: The processing described in this topic is unrelated to the @conkeyref attribute. In that case @conkeyref is used to determine the target of a @conref attribute, after which the normal @conref rules apply.

Empty elements that include a key reference with a defined key might get their effective content from the key definition. Empty elements are defined as elements which meet the following criteria:

- Have no text content
- Have no sub-elements

Comment by robander, September 16 2014
There is a recognized edge case that could confuse processors. What if my specialization has a required kid. For validity, I must include that kid. (Same issue tends to come up with conref on something like <step> or <table>). For example, if I have an image with required <alt> child, to be valid it must be: <myImage keyref="a"><alt/></myImage>. This can never be empty, thus can never pull the alt-text along with @href.

Not sure what the alternative is though. Originally considered text: "Have no sub-elements that contain text or result in text". But as Chris pointed out this is troublesome - how can a processor know if a child element will result in text? Will <myData/> result in text? What if it results in text for PDF but not HTML - then does my effective content get pulled from the key for one and not the other? What if you have <keyword> child with @conref, but @conref doesn't resolve or pulls nothing? In that case again we have something that may be empty now but resolve later. So, for now using the much simpler "no sub-elements" rule, while recognizing that the edge case exists.

- Have no attributes that would be used as text content (such as @alt on the <image> element)

When a key definition has a child <topicmeta> element, content from that <topicmeta> element is used to determine effective content. Effective content from the key definition becomes the element content, with the following exceptions:

- For empty <image> elements, effective content is used as alternate text, equivalent to creating an <alt> sub-element to hold that content.
- For empty <link> elements, effective content is used as link text, equivalent to creating an <linktext> sub-element to hold that content.
- For empty <link> and <xref> elements, a key definition can be used to provide a short description in addition to the normal effective content. If the key definition includes <shortdesc> inside of <topicmeta>, that <shortdesc> should be used to provide effective content for a <desc> sub-element.

Comment by robander, 29 July 2014
Original DITA 1.2 text for this was: "For <link> tags with a @keyref attribute, the contents of the <shortdesc> tag in the key-defining element should provide the <desc> contents." This restatement expands that to <xref> which I believe was the intent. It also tries to stay consistent by using the term "effective content". Hopefully it is as clear / clearer.

- The <longdescref> and <longquoteref> elements are empty elements with no effective content. Key definitions are not used to set effective text for these elements.
- The <indextermref> element is not completely defined, so determining effective content for this element is also left undefined.
- The <abbreviated-form> is an empty element with special rules that determine effective content.

Effective text content is determined using the following set of rules:
1. For the `<abbreviated-form>` element, see the rules described in `<abbreviated-form>`

2. If an element does not allow the `@href` attribute, content is taken from the first `<keyword>` or `<term>` element inside of `<keywords>` inside of the `<topicmeta>`. For example, given the following key definition, empty `<keyword>`, `<term>`, and `<dt>` elements with the attribute `keyref="noreferrer"` would all result in the matching content "first":

```xml
<keydef keys="noreferrer">
  <topicmeta>
  <keywords><keyword>first</keyword><keyword>second</keyword><term>third</term></keywords>
  </topicmeta>
</keydef>
```

3. For elements that do allow `@href`, elements from within `<topicmeta>` that are legal within the element using `@keyref` are considered matching text. For example, the `<xref>` element allows `@href`, and also allows both `<keyword>` and `<term>` as children. Using the code sample from the previous item, an empty `<xref>` with `keyref="test"` would use all three of these elements as text content; after processing, the result would be equivalent to:

```xml
<xref keyref="test"><keyword>first</keyword><keyword>second</keyword><term>third</term></xref>
```

4. Otherwise, if `<linktext>` is specified inside of `<topicmeta>`, the contents of `<linktext>` are used as the effective content.

Comment by robander, 29 July 2014

I'd really like to add a note here to call out the fact that this might be the best way to specify your replacement text - it's the only way to ensure that all elements referencing this key pick up the same text, regardless of whether they have `@href` or become links. I'm not sure if that extra text is warranted or is really a "best practice" in the view of the TC.

5. Otherwise, if the element with the key reference results in a link, normal link text determination rules apply as they would for `<xref>` (for example, using the `<navtitle>` or falling back to the URI of the link target).

Comment by robander, 29 July 2014

I think from close review that we should have a rule in here that is specific to elements allowed inside of `<topicmeta>` -- that is, author, source, data, and data-about. If those exist as a child of `<topicmeta>` in the key definition, then they are used to provide the content for the same elements with `@keyref`. I think (but am not sure) that this is the correct interpretation of the following line in the DITA 1.2 spec: "When a key definition has a `<topicmeta>` subelement, elements that refer to that key and that are empty may get their effective content from the first matching subelement of the `<topicmeta>` subelement of the key-defining topicref." I've left it out until others confirm.

When the effective content for a key reference element results in invalid tagging, that tagging should be generalized. For example, `<linktext>` in the key definition may use a domain specialization of `<keyword>` that is not valid in the key reference context, in which case the specialized element should be generalized to `<keyword>`. If the generalized content is also not valid, a text equivalent should be used instead. For example, `<linktext>` may include `<ph>` or a specialized `<ph>` in the key definition, but neither of those are valid as the effective content for a `<keyword>`. In that case, the text content of the `<ph>` should be used.

Comment by robander, 29 July 2014

This is a change from DITA 1.2, but one that is needed to avoid nonsensical results. The 1.2 language was: "Elements within `<linktext>` that do not match the content model of the key reference directly or after generalization should be skipped." In this case, the following key definition would give varying results:

```xml
<keydef keys="ph">
  <topicmeta>
  <linktext>This is <b>IMPORTANT</b></linktext>
  </topicmeta>
</keydef>
```
3.3.5.7 Examples of keys

This section of the specification contains examples and scenarios. They illustrate a wide variety of ways that keys can be used.

3.3.5.7.1 Examples: Key definition

Either <topicref> or <keydef> elements can be used to define keys.

In the following example, a <topicref> element is used to define a key; the <topicref> element also contributes to the navigation structure.

```
<map>
  <!--... -->
  <topicref keys="apple-definition" href="apple-gloss-en-US.dita" />
  <!--... -->
</map>
```

The `apple-gloss-en-US.dita` topic is processed as if the @keys attribute is not present.

In the following example, a <keydef> element is used to define a key.

```
<map>
  <!--... -->
  <keydef keys="apple-definition" href="apple-gloss-en-US.dita"/>
  <!--... -->
</map>
```

Because the <keydef> element sets the default value of the @processing-role attribute to "resource-only", the key definition does not contribute to the map navigation structure; it only serves as a key definition for "apple-definition".

3.3.5.7.2 Examples: Key definitions for variable text

Key definitions can be used to store variable text, such as product names and user-interface labels. Depending on the key definition, the rendered output might have a link to a related resource.

Comment by Kristen Eberlein

Robert Anderson and I have talked about whether "variable text" is the correct phrase to use. Another option we considered was "generated text," but I think that has too many connotations about text that is generated by style sheets.

This example overlaps significantly with the original "Swap out variable content" example. What that example showed that is not covered here is:

- Using a <linktext> element to hold the variable text
- Authors using different key definitions to "swap out" content

In the following example, a "product-name" key is defined. The key definition contains a child <keyword> element nested within a <keydef> element.

```
<map>
  <keydef keys="product-name">
    <topicmeta>
    <keywords>
```
A topic can reference the "product-name" key using the following markup:

```xml
<topic id="topicid">
  <p><keyword keyref="product-name"/> is a product designed to ...</p>
</topic>
```

When processed, the output contains the text "Thing-O-Matic is a product designed to ... ".

In the following example, the key definition contains both a reference to a resource and variable text.

```xml
<map>
  <keydef keys="product-name" href="thing-o-matic.dita">
    <topicmeta>
      <keywords>
        <keyword>Thing-O-Matic</keyword>
      </keywords>
    </topicmeta>
  </keydef>
</map>
```

When processed using the key reference from the first example, the output contains the "Thing-O-Matic is a product designed to ... " text. The phrase "Thing-O-Matic" also is a link to the thing-o-matic.dita topic.

### 3.3.5.7.3 Example: Scoped key definitions for variable text

Scoped key definition can be used for variable text. This enables you to use the same DITA topic multiple times in a DITA map, and in each instance the variable text can resolve differently.

The Acme Tractor Company produces two models of tractor: X and Y. Their product manual contains sets of instructions for each model. While most maintenance procedures are different for each model, the instructions for changing the oil are identical for both model X and model Y. The company policies call for including the specific model number in each topic, so a generic topic that can be used for both models is not permitted.

1. The authoring team references the model information in the changing-the-oil.dita topic by using the following mark-up:

   ```xml
   <keyword keyref="model"/>
   ```

2. The information architect examines the root map for the manual, and decides how to define key scopes. Originally, the map looks like the following:

   ```xml
   <map>
     <!-- Model X: Maintenance procedures -->
     <topicref href="model-x-procedures.dita">
       <topicref href="model-x/replacing-a-tire.dita"/>
       <topicref href="model-x/adding-fluid.dita"/>
     </topicref>
   </map>
   <map>
     <!-- Model Y: Maintenance procedures -->
     <topicref href="model-y-procedures.dita">
       <topicref href="model-y/replacing-a-tire.dita"/>
       <topicref href="model-y/adding-fluid.dita"/>
     </topicref>
   </map>
   ```

3. The information architect wraps each set of procedures in a <topicgroup> element and sets the @keyscope attribute.

   ```xml
   <map>
     <!-- Model X: Maintenance procedures -->
     <topicgroup keyscope="model-x">
       <topicref href="model-x-procedures.dita">
         <topicref href="model-x/replacing-a-tire.dita"/>
       </topicref>
     </topicgroup>
   </map>
   ```
This defines the key scope for set of procedures.

4. The information architect then adds key definitions to each set of procedures, as well as a reference to the `changing-the-oil.dita` topic.

When the DITA map is processed, the `changing-the-oil.dita` topic is rendered twice. The model variable is rendered differently in each instance, using the text as specified in the scoped key definition.

### 3.3.5.7.4 Example: Duplicate key definitions within a single map

A DITA map might contain duplicate key definitions. How processors find the effective key definition depends on document order and whether conditional processing occurs.

In the following example, a map contains two definitions for the key "load-toner":

In this example, only the first key definition -- in document order -- of the "load-toner" key is effective. All references to the key within the scope of the map resolve to the topic `model-1235-load-toner-proc.dita`.

In the following example, a map contains two definitions for the "file-chooser-dialog" key; each key definition specifies a different value for the @platform attribute.
In this case, the effective key definition is determined not only by the order in which the definitions occur, but also by whether the active value of the @platform attribute is "osx" or "windows7". Both key definitions are potentially effective because they have distinct values for the conditional attribute. Note that if no active value is specified for the @platform attribute at processing time, then both of the key definitions are present and so the first one in document order is the effective definition.

If the DITAVAL settings are defined so that both "osx" and "windows" values for the @platform attribute are excluded, then neither definition is effective and the key is undefined. That case can be avoided by specifying an unconditional key definition after any conditional key definitions, for example:

```xml
<map>
    <!--... -->
    <keydef keys="file-chooser-dialog" href="file-chooser-osx.dita" platform="osx"/>
    <keydef keys="file-chooser-dialog" href="file-chooser-win7.dita" platform="windows7"/>
    <keydef keys="file-chooser-dialog" href="file-chooser-generic.dita"/>
    <!--... -->
</map>
```

If the above map is processed with both "osx" and "windows" values for the @platform attribute excluded, then the effective key definition for "file-chooser-dialog" is the file-chooser-generic.dita resource.

### 3.3.5.7.5 Example: Duplicate key definitions across multiple maps

A root map might reference submaps, each of which might contain duplicate key definitions. Which key definition is effective depends on the document order.

In the following example, a root map contains a key definition for the "toner-specs" and references to two submaps.

```xml
<map>
    <!--... -->
    <keydef keys="toner-specs" href="toner-type-a-specs.dita"/>
    <mapref href="submap-01.ditamap"/>
    <mapref href="submap-02.ditamap"/>
</map>
```

The first submap, submap-01.ditamap, contains key definitions for the "toner-specs" and "toner-handling":

```xml
<map>
    <keydef keys="toner-specs" href="toner-type-b-specs.dita"/>
    <keydef keys="toner-handling" href="toner-type-b-handling.dita"/>
</map>
```

The second submap, submap-02.ditamap, contains key definitions for the "toner-specs", "toner-handling", and "toner-disposal":

```xml
<map>
    <keydef keys="toner-specs" href="toner-type-c-specs.dita"/>
    <keydef keys="toner-handling" href="toner-type-c-handling.dita"/>
    <keydef keys="toner-disposal" href="toner-type-c-disposal.dita"/>
</map>
```

For this example, the effective key definitions are listed in the following table.

<table>
<thead>
<tr>
<th>Key</th>
<th>Bound resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>toner-specs</td>
<td>toner-type-a-specs.dita</td>
</tr>
<tr>
<td>toner-handling</td>
<td>toner-type-b-handling.dita</td>
</tr>
</tbody>
</table>
The key definition for "toner-specs" in the root map is effective, because it is the first encountered in a breadth-first traversal of the map tree. The key definition for "toner-handling" in submap-01.ditamap is effective, because submap-01 is included before submap-02 and so comes first in the map tree. The key definition for "toner-disposal" is effective because it is the only definition of the key in the map tree.

### 3.3.5.7.6 Example: Scoped key references

You can address scoped keys definitions from outside the key scope where the key definitions are specified.

```xml
<map xml:lang="en">
  <title>Examples of scoped key references</title>
  <!-- Key scope #1 -->
  <topicgroup keyscope="scope-1">
    <keydef keys="key-1" href="topic-1.dita"/>
    <topicref keyref="key-1"/>
    <topicref keyref="scope-1.key-1"/>
    <topicref keyref="scope-2.key-1"/>
  </topicgroup>
  <!-- Key scope #2 -->
  <topicgroup keyscope="scope-2">
    <keydef keys="key-1" href="topic-2.dita"/>
    <topicref keyref="key-1"/>
    <topicref keyref="scope-1.key-1"/>
    <topicref keyref="scope-2.key-1"/>
  </topicgroup>
  <topicref keyref="key-1"/>
  <topicref keyref="scope-1.key-1"/>
  <topicref keyref="scope-2.key-1"/>
</map>
```

For this example, the effective key definitions are listed in the following tables.

<table>
<thead>
<tr>
<th>Key reference</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>key-1</td>
<td>topic-1.dita</td>
</tr>
<tr>
<td>scope-1.key-1</td>
<td>topic-1.dita</td>
</tr>
<tr>
<td>scope-2.key-1</td>
<td>topic-2.dita</td>
</tr>
</tbody>
</table>

**Table 4: Effective key definitions for scope #2**

<table>
<thead>
<tr>
<th>Key reference</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>key-1</td>
<td>topic-2.dita</td>
</tr>
<tr>
<td>scope-1.key-1</td>
<td>topic-1.dita</td>
</tr>
<tr>
<td>scope-2.key-1</td>
<td>topic-2.dita</td>
</tr>
</tbody>
</table>
Table 5: Effective key definitions for root map

<table>
<thead>
<tr>
<th>Key reference</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>key-1</td>
<td>Undefined</td>
</tr>
<tr>
<td>scope-1.key-1</td>
<td>topic-1.dita</td>
</tr>
<tr>
<td>scope-2.key-1</td>
<td>topic-2.dita</td>
</tr>
</tbody>
</table>

Comment by Kristen Eberlein

The comments in the original code block includes the following: "Keys defined in a scope become a part of the parent scope, prefixed with the scope name and a period."

Do they really become part of the parent scope? Or is it just that they can be addressed from within the parent scope?

We've use the terminology "parent scope" and "child scope". Do we need "peer scope"?

3.3.5.7.7 Example: Key definitions in nested key scopes

A root map might contain nested key scopes, each of which might contain duplicate key definitions. Which key definition is effective depends on key-scope precedence rules.

Consider the following DITA map:

```xml
<map>
  <title>Root map</title>
  <!-- Root scope -->
  <keydef keys="a"/>

  <!-- Key scope A -->
  <topicgroup keyscope="A">
    <keydef keys="b"/>

    <!-- Key scope A-1 -->
    <topicgroup keyscope="A-1">
      <keydef keys="c"/>
    </topicgroup>

    <!-- Key scope A-2 -->
    <topicgroup keyscope="A-2">
      <keydef keys="d"/>
    </topicgroup>
  </topicgroup>

  <!-- Key scope B -->
  <topicgroup keyscope="B">
    <keydef keys="e"/>

    <!-- Key scope B-1 -->
    <topicgroup keyscope="B-1">
      <keydef keys="f"/>
    </topicgroup>

    <!-- Key scope B-2 -->
    <topicgroup keyscope="B-2">
      <keydef keys="g"/>
    </topicgroup>
  </topicgroup>
</map>
```

The key scopes in this map form a tree structure.
Each box in the diagram represents a key scope; the name of the key scope is indicated in bold with upper-case letters. Below the name of the key scope, the key definitions that are present in the scope are listed. Different typographic conventions are used to indicate where the key definition occurs:

**No styling**

The key definition occurs in the immediate key scope and is not overridden by a key definition in a parent scope. For example, key "a" in the root map.

**Parentheses**

The key definition occurs in a child scope. For example, keys "A-1.c" and "A-2.d" in key scope A.

**Brackets**

The key definition occurs in the immediate key scope, but it is override by a key definition in a parent scope. For example, key "a" in key scope B.

Arrows points from child to parent scopes.

Assume that each key scope contains numerous key references. For this example, effective key definitions for certain key scopes are listed in the following tables.

**Table 6: Key scope A-2**

<table>
<thead>
<tr>
<th>Key reference</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>&quot;a&quot;, defined in the root map</td>
</tr>
<tr>
<td>d</td>
<td>&quot;d&quot;, as defined in the immediate key scope</td>
</tr>
<tr>
<td>A-2.d</td>
<td>&quot;d&quot;, as defined in the immediate key scope</td>
</tr>
<tr>
<td>c</td>
<td>Undefined</td>
</tr>
<tr>
<td>A-1.c</td>
<td>&quot;A-1.c&quot;, as defined in key scope A-1. This key name is available because it exists in the parent scope, key scope A.</td>
</tr>
<tr>
<td>A.A-1.c</td>
<td>&quot;A-1.c&quot;, as defined in key scope A-1. This key name is available because it exists in the root key scope.</td>
</tr>
</tbody>
</table>
Table 7: Key scope B

<table>
<thead>
<tr>
<th>Key reference</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>&quot;e&quot;, defined in the immediate key scope</td>
</tr>
<tr>
<td>a</td>
<td>&quot;a&quot;, as defined in the root key scope. (While a key definition for &quot;a&quot; exists in the immediate key scope, it is overriden by the key definition that occurs in the parent key scope.)</td>
</tr>
<tr>
<td>B.a</td>
<td>&quot;a&quot;, as defined in the immediate key scope. However, since a key definition for &quot;a&quot; exists in the parent key scope, the effective value for &quot;a&quot; in the immediate key scope still is overriden by the key definition that occurs in the parent key scope.</td>
</tr>
<tr>
<td>g</td>
<td>Undefined</td>
</tr>
<tr>
<td>B-2.g</td>
<td>&quot;g&quot;, as defined in key scope B-2.</td>
</tr>
</tbody>
</table>

3.3.5.7.8 Example: Link redirection

This scenario outlines how different authors can redirect links to a common topic by using key definitions. This could apply to either `<xref>` or `<link>` elements.

A company wants to use a common DITA topic for information about recycling: `recycling.dita`. However, the topic contains a cross-reference to a topic that needs to be unique for each product line; each such topic contains product-specific URLs.

1. The editing team creates a `recycling.dita` topic that includes a cross-reference to the product-specific topic. The cross reference is implemented using a key reference:

   ```
   <xref keyref="product-recycling-info" href="generic-recycling-info.dita"/>
   ```

   The value of the `@href` attribute provides a fallback in the event that a product team forgets to include a key definition for "product-recycling-info".

2. Each product documentation group creates a unique key definition for "product-recycling-info". Each group authors the key definition in a DITA map, for example:

   ```
   <map>
   <!-- ... -->
   <keydef keys="product-recycling-info" href="acme-server-recycling.dita"/>
   <!-- ... -->
   </map>
   ```

   Each team can use the `recycling.dita` topic, and the cross reference in the topic resolves differently for each team.

3. A year later, there is an acquisition. The newly-acquired team wants to reuse Acme's common material, but it needs to direct its users to an external Web site that lists the URLs, rather than a topic in the product documentation. Their key definition looks like the following:

   ```
   ```

   When newly-acquired team uses the `recycling.dita` topic, it resolves to the external Web site; however for all other teams, the cross reference in the topic continues to resolves to their product-specific topic.
4. A new product team is formed, and the team forgets to include a key definition for "product-recycling-info" in one of their root maps. Because the cross reference in the recycling.dita topic contains a value for the @href attribute, the link falls back to generic-recycling-info.dita, thus avoiding a broken cross reference in the output.

3.3.5.7.9 Example: Link modification or removal

This scenario outlines how different authors can remove or modify a <link> element in a shared topic.

A company wants to use a shared topic for information about customer support. For most products, the shared topic should include a link to a topic about extended warranties. But a small number of products do not offer extended warranties.

1. Team one creates the shared topic: customer-support.dita. The topic contains the following mark-up:

```xml
<related-links>
  <link keyref="extended-warranties" href="common/extended-warranties.dita"/>
</related-links>
```

2. The teams that need the link to the topic about extended warranties can reference the customer-support.dita topic in their DITA maps. When processed, the related link in the topic resolves to the common/extended-warranties.dita topic.

3. The teams that do not want the related link to the topic about extended warranties can include a key definition that in their DITA map that does not include an @href attribute, for example:

```xml
<map>
  <!-- ... -->
  <keydef keys="extended-warranties"/>
  <!-- ... -->
</map>
```

When processed, the related link in the topic is not rendered.

4. Yet another team wants to simply have a paragraph about extended warranties printed. They define the key definition for "extended-warranties" as follows:

```xml
<map>
  <!-- ... -->
  <keydef keys="extended-warranties"/>
  <topicmeta>
    <linktext>This product does not offer extended warranties.</linktext>
  </topicmeta>
  <!-- ... -->
</map>
```

When this team renders their content, there is no hyperlink in the output, just the "The product does not offer extended warranties" statement.

Comment by Kristen Eberlein

This does not work in DITA-OT 1.8.4.

3.3.5.7.10 Example: Links from <term> or <keyword> elements

The @keyref attribute enables authors to specify that references to keywords or terms in a DITA topic should be rendered as a link to an associated resource.

In this scenario, a company with well-developed glossary wants to ensure that instances of a term that is defined in the glossary always include a link to the glossary topic.
1. An information architect adds values for the @keys attribute to all the of the <topicref > elements that are in the DITA map for the glossary, for example:

```xml
<map>
  <title>Company-wide glossary</title>
  <topicref keys="term-1" href="term-1.dita"/>
  <topicref keys="term-2" href="term-2.dita"/>
  <topicref keys="term-3" href="term-3.dita"/>
  <topicref keys="term-4" href="term-4.dita"/>
</map>
```

2. When authors refer to a term in a topic, they use the following mark-up:

```xml
<term keys="term-1"/>
```

When rendered, the content is pulled from the <title> element of the glossary topic. It also is a link to the glossary topic.

### 3.3.5.7.11 Example: conref redirection

The @conkeyref attribute enables authors to share DITA topics that reuse content; it enables each author to specify a different key definition for common keys.

In this scenario, Acme produces content for product that is also resold through a business partner. When the DITA content is published for the partner, several items must be different, including the following:

- Product names
- Standard notes that contain admonitions

Simply using the @conref attribute would not be possible for teams that use a component content management system where every DITA topic is addressed by a globally-unique identifier (GUID).

1. Authors reference the reusable content in their topics by using the @conkeyref attribute, for example:

```xml
<task>
  <title><keyword conkeyref="reuse/product-name"/> prerequisites</title>
  <!-- ... -->
  <note conkeyref="warning-1"/>
  <!-- ... -->
</task>
```

2. Authors create two different topics; one topic contain elements that contain the text that is appropriate for Acme, and the other topic contains elements that contain the text that is appropriate for the partner. Note that each reuse topic must contain the same DITA tags and values for the @id attribute, for example:

```xml
<topic id="reuse-oem">
  <title>Reuse topic for OEM partner</title>
  <body>
    <!-- ... -->
    <note id="warning-1"/>
    <p><keyword id="product-name"/></p>
    <!-- ... -->
  </body>
</topic>
```

3. The two versions of the DITA maps each contain different key definitions for "reuse". (This associates a key with the topic that contains the appropriate reusable elements.) For example:

```xml
<map>
  <!-- ... -->
  <keydef keys="reuse" href="acme-reuse.dita"/>
```
Figure 15: DITA map for Acme

<map>
  <!-- ... -->
  <keydef keys="reuse" href="oem-reuse.dita"/>
  <!-- ... -->
</map>

Figure 16: DITA map for OEM partner

When the DITA maps are published, the elements that are referenced by @conkeyref will use the reuse topic that is referenced by the <keydef > element in the map. The product names and warnings will be different in the output.

3.3.5.7.12 Example: Key scopes and omnibus publications

Key scopes enable you to create omnibus publications that include multiple submaps that have common keys.

In this scenario, a training organization wants to produce a deliverable that includes all of their training course materials. Each course manual uses common keys for standard parts of the course materials, including "prerequisites," "overview", "assessment", and "summary."

An information architect creates a DITA map that contains the following mark-up:

```xml
<map xml:lang="en">  
  <mapref href="course-1.ditamap"/>
  <mapref href="course-2.ditamap"/>
  <mapref href="course-3.ditamap"/>
  <topicref href="omnibus-summary.dita"/>
</map>
```

Each of the submaps contain <topicref> elements that reference resources using the @keyref attribute. Each submap uses common keys for standard parts of the course materials, including "prerequisites," "overview", "assessment", and "summary", and their key definitions resolve the key names to course-specific resources. For example:

```xml
<map xml:lang="en">  
  <title>Training course #1</title>
  <mapref href="course-1/key-definitions.ditamap"/>
  <topicref keyref="prerequisites"/>
  <topicref keyref="overview"/>
  <topicref keyref="assessment"/>
  <topicref keyref="summary"/>
</map>
```

Without using key scopes, the effective key definitions for the common keys resolve to those found in course-1.ditamap. This is not the desired outcome. By adding key scopes to the submaps, however, the information architect can ensure that the key references in the submaps resolve to the course-specific key definitions.

```xml
<map xml:lang="en">  
  <title>Training courses</title>
  <mapref href="course-1.ditamap" keyscope="course-1"/>
  <mapref href="course-2.ditamap" keyscope="course-2"/>
  <mapref href="course-3.ditamap" keyscope="course-3"/>
  <topicref href="omnibus-summary.dita"/>
</map>
```

The information architect does not set keys="summary" on the <topicref> element in the omnibus map. Doing so would mean that all key references to "summary" in the submaps would resolve to omnibus-
summary.dita, rather than the course-specific summary topics. This is because key definitions located in parent scopes override those located in child scopes.

3.3.5.7.13 Example: Keys and collaboration

Keys enable authors to collaborate and work with evolving content with a minimum of time spent reworking topic references.

Comment by Kristen Eberlein

In the 1.2 spec, this example was titled "Splitting or recombining targets." I think we need to find a more appropriate title, and I don’t know if what I've used is the correct choice. The example really is about link redirection with a minimum of fuss ...

In this scenario, authors collaborate on a publication that includes content for a product that is in the early stages of development. The company documentation is highly-structured and uses the same organization for all publications: "Introduction," "Example," and "Reference."

1. Author one creates a submap for the new product information. She knows the structure that the final content will have, but she does not want to create empty topics for information that is not yet available. She decides to initially author what content is available in a single topic. When more content is available, she'll create additional topics. Her DITA map looks like the following:

```xml
<map>
  <title>New product content</title>
  <topicref keys="1-overview 1-intro 1-example 1-reference" href="1-overview.dita"/>
</map>
```

2. Author two knows that he needs to add a `<topicref>` to the "Example" topic that will eventually be authored by author one. He references the not-yet-authored topic by key reference:

```xml
<topicref keyref="1-example"/>
```

His topic reference initially resolves to the `1-overview.dita` topic.

3. Author one finally gets the information that she was waiting on. She creates additional topics and modifies her DITA map as follows:

```xml
<map>
  <title>New product content</title>
  <topicref keys="1-overview href="1-overview.dita">
    <topicref keys="1-intro" href="1-intro.dita"/>
    <topicref keys="1-example" href="1-example.dita"/>
    <topicref keys="1-reference" href="1-reference.dita"/>
  </topicref>
</map>
```

Without needing to make any changes to the content, author two's topic reference now resolves to the `1-example.dita` topic.

3.3.6 Content inclusion (conref)

The DITA `@conref`, `@conkeyref`, `@conrefend`, and related attributes provide a mechanism for reuse of content fragments within DITA topics or maps.

Comment by Kristen Eberlein, 16 February 2014

Comment from Michael Priestley, 14 January:

"Does mention constraints, but in language that is inherited from DITA 1.1, so in fact it's not referring to constraints in the DITA 1.2 sense but using the word in a generic fashion. Needs to be updated to reflect DITA 1.2 meaning of constraint, intended processing behavior, and link to specific topic on constraints and conref/generlization."
I replaced two generic instances of the word constraint with the word "restriction".

Comment by Kristen Eberlein, 3 April 2014

In working through review #1 comments, I moved a section of the original topic into a child topic: "Handling xrefs and conrefs within a conref". Now I think this topic needs to become a container, probably not much more than a title and short description; the current content needs to be split into one or more other topics.

- The `@conref` or `@conkeyref` attribute can be used to pull the referenced content into the location of the referencing element. The combination of either of these attributes with the `@conrefend` attribute can be used to pull the content of a range of elements.
- The `@conref` attribute can be used in combination with the `@conaction` attribute to push content from the referencing element to the location of the referenced element.

### Pulling content to the referencing element

When the `@conref` or `@conkeyref` attribute is used alone, the referencing element acts as a placeholder for the referenced element, and the content of the referenced element is rendered in place of the referencing element.

The combination of the `@conrefend` attribute with either `@conref` or `@conkeyref` specifies a range of sibling elements that is rendered in place of the referencing element. See *The conrefend attribute* (see page 670) for examples of how to combine `@conrefend` with either `@conref` or `@conkeyref`.

### Pushing content from the referencing element

The `@conaction` attribute reverses the direction of reuse from pull to push. When the `@conref` or `@conkeyref` attribute is used in combination with the `@conaction` attribute, content can be rendered before, after, or in place of the referenced element, depending on the value of the `@conaction` attribute. See *The conaction attribute* (see page 667) for more details.

Note: The `@conaction` and `@conrefend` attributes cannot both be used within the same referencing element, so it is not possible to push a range of elements.

The identifier for the referenced element *MUST* be either absolute or resolvable in the context of the referencing element.

More formally, the DITA `@conref` attribute can be considered a transclusion mechanism similar to XInclude and to HyTime value references. DITA differs from these mechanisms, however, in that conref validity does not apply simply to the current content at the time of replacement, but to the ranges of possible content given the restrictions of both the referencing document type and the referenced document type. DITA compares the restrictions of each context to ensure the continued validity of the replacement content in its new context. A conref processor *MUST NOT* permit resolution of a reuse relationship that could be rendered invalid under the rules of either the reused or reusing content.

When pulling content with the conref mechanism – if the referenced element is the same type as the referencing element, and the list of domains declared on the `@domains` attribute in the referenced topic or map instance is the same as or a subset of the list of domains declared in the referencing document, the element set allowed in the referenced element is guaranteed to be the same as, or a subset of, the element set allowed in the referencing element. A processor resolving a conref *SHOULD* tolerate specializations of valid elements and *SHOULD* generalize elements in the pulled content fragment as needed for the referencing context.

When pushing content with the conref mechanism, the domain checking algorithm is reversed. In this case, the `@domains` attribute on the referenced document’s topic or map *MUST* be the same as or a superset of the domains that are declared on the referencing document. Once again, a processor resolving a conref *SHOULD* tolerate specializations of valid elements and *SHOULD* generalize elements in the pushed content fragment as needed for the referenced context.

All replacement of content based on `@conref` occurs after parsing of the document but prior to any styling or other transformational or presentational operations on the full topic.
The referenced element might replace the referencing element based on build-time or runtime conditions. For example, content such as product names or install paths might differ from one product to another. It is advantageous to separate such content from topic content which is reused for more than one product. When the content is reused in a different context, different resources are substituted as reference elements.

A fragment of DITA content, such as an XML document containing only a single paragraph without a topic or map ancestor, does not contain enough information for the conref processor to be able to determine the validity of a reference to it. Consequently, the value of a conref MUST specify one of the following items:

- A referenced element within a DITA map
- A referenced element within a DITA topic
- An entire DITA map
- An entire DITA topic

The attribute specifications on the resolved element can be drawn from both the referencing element and the referenced element, according to the following priority:

1. All attributes as specified on the referencing element, except for attributes which specify the value "-dita-use-conref-target". (The term "target" here refers to the referenced element.)
2. All attributes as specified on the referenced element except:
   a. The @id attribute
   b. Any attribute that is also specified on the referencing element, except when the value specified on the referencing element is "-dita-use-conref-target"
3. The @xml:lang attribute has special treatment as described in The xml:lang attribute (see page 107).

The only time the resolved element would include an attribute whose specified value is "-dita-use-conref-target" is when the referenced element had that attribute specified with the "-dita-use-conref-target" value and the referencing element either had no specification for that attribute or had it also specified with the "-dita-use-conref-target" value. If the final resolved element (after the complete resolution of any conref chain, as explained below) has an attribute with the "-dita-use-conref-target" value, that MUST be treated as equivalent to having that attribute unspecified.

A given attribute value on the resolved element comes in its entirety from either the referencing element or the referenced element; the attribute values of the referencing and referenced elements for a given attribute are never additive, even if the property (such as the audience type) takes a list of values.

If the referenced element has a @conref attribute specified, the above rules should be applied recursively with the resolved element from one referencing/referenced combination becoming one of the two elements participating in the next referencing/referenced combination. The result should preserve without generalization all elements that are valid in the originating context, even if they are not valid in an intermediate context. For example, if topicA and topicC allow highlighting, and topicB does not, then a content reference chain of topicA->topicB->topicC should preserve any highlighting elements in the referenced content. The result, however it is achieved, must be equivalent to the result of resolving the conref pairs recursively starting from the original referencing element in topicA.

The @conrefend attribute is used when referencing a range of elements with the conref mechanism. The @conref attribute references the first element in the range, while @conrefend points to the last element in the range. Although the start and end referenced elements MUST both be of the same type as the referencing element (or specialized from that element type), the intermediary, contiguous nodes in the middle of the range do not have to be the same.

Related Links

3.4.4.4 Conref and generalization for constraint modules (see page 188)

When documents use different constraints, conref and generalization processors can examine the @domains attribute to verify compatibility between the document instances.

3.3.6.1 Handling xrefs and conrefs within a conref

Comment by Kristen Eberlein , 11 December 2013
When referenced content contains a content reference or cross reference, the effective target of the reference depends on the form of address that is used in the referenced content.

- When the address used is a direct URI reference of any form other than a same-topic fragment identifier, processors **MUST** resolve it relative to the referenced content in its source location.
- When the address is a same-topic fragment identifier or a key reference, processors **MUST** resolve it relative to the location of the content reference.

This means that the same content reference or cross reference can resolve to different targets in different use contexts because "same topic" references resolve to the topic containing the reference and key references are resolved in the context of the key scope in effect for each use of the topic containing the reference. In the case of "same topic" fragment identifiers, it is the responsibility of the author of the content reference to ensure that any element IDs specified in same-topic fragment identifiers in the referenced content are either in the referenced content or provided by elements in the referencing topic.

**Examples**

Consider the following paragraphs in `paras-01.dita` that are intended to be used by reference from other topics:

```xml
<topic id="paras-01"><title>Paras for Reuse</title>
<body>
  <p id="p1">See <xref href="#paras-01/p5">P5</xref></p>
  <p id="p2">See <xref href="#topic-02.dita#topic02/fig-01">Figure 01</xref></p>
  <p id="p3">See <xref href="#./p5">P5</xref></p>
  <p id="p4">See <xref keyref="task-remove-cover">Remove the cover</xref></p>
  <p id="p5">Paragraph 5 in paras-01</p>
</body></topic>
```

Where these paragraphs are used by content reference from other topics, for example, the `using-topic-01` topic:

```xml
<topic id="using-topic-01"><title>Using Topic One</title>
<body>
  <p id="A" conref="paras-01.dita#paras-01/p1"/>
  <p id="B" conref="paras-01.dita#paras-01/p2"/>
  <p id="C" conref="paras-01.dita#paras-01/p3"/>
  <p id="D" conref="paras-01.dita#paras-01/p4"/>
  <p id="p5">Paragraph 5 in using-topic-01</p>
</body></topic>
```

Following resolution of the content references and processing of the `<xref>` elements in the referenced paragraphs, the resulting rendered cross references will be as follows:

- For paragraph "A" (conref to paragraph "p1"), the reference is to the paragraph "p5" in topic `paras-01.dita`, that is, the paragraph with the content "Paragraph 5 in paras-01". The reference in paragraph p1 is a direct URI reference that is not a "same topic" fragment identifier, therefore it can only be resolved to element "p5" in topic `paras-01.dita`.
- For paragraph "B" (conref to paragraph "p2"), the reference is to the element "fig-01" in topic `topic-02.dita`. Because this is a direct URI reference, every use of paragraph "p2" will resolve to the same target relative to topic `paras-01.dita` (the location of the cross reference as authored).
- For paragraph "C" (conref to paragraph "p3"), the reference is to element "p5" in topic "using-topic-01", the paragraph with the text "Paragraph 5 in using-topic-01". Because the URI reference is just a "same topic"
fragment identifier, the reference is resolved in the context of the topic making the content reference (topic using-topic-01). If the topic "using-topic-01" had not contained an element with the ID "p5", then that use of paragraph "p3" would result in a link resolution failure because the "same topic" fragment identifier would not be resolvable in that use context.

• For paragraph "D" (conref to paragraph "p4"), the reference is to whatever topic is bound to the key name "task-remove-cover" in each map context within which topic "using-topic-01" is used.

Consider this map:

```xml
<map>
  <topicgroup keyscope="prod-1">
    <topicref
      keys="task-remove-cover"
      href="prod-1-task-remove-cover.dita"
    />
    <topicref
      href="using-topic-01.dita"
    />
  </topicgroup>
  <topicgroup keyscope="prod-2">
    <topicref
      keys="task-remove-cover"
      href="prod-2-task-remove-cover.dita"
    />
    <topicref
      href="using-topic-01.dita"
    />
  </topicgroup>
</map>
```

The map establishes two key scopes "prod-1" and "prod-2". Within each of these branches of the map the key name "task-remove-cover" is bound to a different topic. The topic using-topic-01.dita is used from both branches. In each of those uses, the target of the key reference from conreffed paragraph "p4" will be different.

In the first branch, key scope "prod-1", the cross reference from paragraph "p4" will be to topic prod-1-task-remove-cover.dita. In the second branch, key scope "prod-2", the cross reference will be to topic prod-2-task-remove-cover.dita.

Related Links

4.12.13.1 The href attribute (see page 663)

The @href attribute is used to reference another DITA topic or map, a specific element inside a DITA topic or map, an external Web page, or another non-DITA resource.

3.3.7 Conditional processing (profiling)

Conditional processing, also known as profiling, is the filtering or flagging of information based on processing-time criteria.

DITA defines attributes that are specifically intended to enable filtering or flagging of individual elements. Those attributes are @audience, @platform, @product, @otherprops, @deliveryTarget, @props, and @rev (flagging only). These attributes enable the creation of topics and maps that can be dynamically configured at processing time to reflect a specific set of conditions, using the DITA-defined conditional-processing profile (DITAVAL).

Processors SHOULD be able to perform filtering and flagging using the attributes listed above. The @props attribute can be specialized to create new attributes, and processors SHOULD be able to perform conditional processing on specializations of @props.

Although metadata elements exist with similar names, such as the <audience> element, processors are not required to perform conditional processing using metadata elements.

Related Links

4.12.1.2 Metadata attribute group (see page 651)
The metadata attribute group includes common metadata attributes, several of which support conditional processing (filtering and flagging) or the creation of new attribute domain specializations.

4.9 DITAVAL elements (see page 414)

A conditional processing profile (DITAVAL file) is used to identify which values are to be used for conditional processing during a particular output, build, or some other purpose. The profile should have an extension of .ditaval.

3.2.3.2.1 Conditional processing attributes (see page 44)

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

3.3.7.1 Conditional processing attributes and metadata elements

Comment by Kristen Eberlein, 8 February 2014
Should this be a separate topic? What really is the focus -- a lot of the content really focuses on the coordination of attributes and elements ...? How should this content connect with that in the "Conditional processing attributes" topic in the "Markup" section?

Audience, platform, and product metadata can be expressed in two ways:

- Using attributes on elements within a topic or on <topicref> elements
- Using elements within the <prolog> element (topics) or <topicmeta> element (maps)

While the metadata elements are more expressive, the meaning of the values is the same and can be used in coordination. For example, the prolog elements can fully define the audiences for a topic, and then metadata attributes can be used within the content to identify the parts that apply to only some of those audiences.

Comment by Kristen Eberlein, 8 February 2014
What are we trying to convey by stating "the meaning of the values is the same"? It obviously isn't, as far as the fact that the elements cannot be used for conditional processing.

Do we want to provide an example of using elements and attributes in coordination? While this material has been in the spec for a long time, I've never seen DITA content that actually coordinated the attributes and elements.

Comment by robander, 21 April 2014
I don't think we should try and provide an example of that - it would be taken as the "way to do it", when as you say, I've never seen anybody implement it. I think it would probably be best to comment out the language, given that we've dropped the proposal to define this from (at least) DITA 1.2 and 1.3, due to lack of people willing to work on it.

Comment by robander, 2013
Deb's review pointed out that the attribute descriptions below are very inconsistent in their style / wording. This wording goes back to DITA 1.2 or earlier, but should be cleaned up when integrating the new material.

Comment by Kristen Eberlein, 6 March 2014
Do we really want to say that the conditional processing attributes are equivalent to the metadata elements?

Comment by robander, 21 April 2014
I don't think so. Maybe time to remove that, given the failed efforts to coordinate them?

@audience

The end user or reader. This attribute is equivalent to the <audience> element. The value of the @audience attribute can be used to reference a more complete description of an audience that is specified in an <audience> element; use the same value for the @name attribute on the <audience> element and the @audience attribute.
The `@audience` attribute takes a space-delimited list of values, which might or might not match the name value of any `<audience>` elements. The attribute also can include groups of values specified using the same syntax as generalized attributes within `@props`; see Attribute generalization (see page 152) for details on grouping syntax.

`@platform`

The operating system, hardware, or other environment. This attribute is equivalent to the `<platform>` element.

The `@platform` attribute takes a space-delimited list of values, which might or might not match the content of a `<platform>` element in the prolog. The attribute also can include groups of values specified using the same syntax as generalized attributes within `@props`; see Attribute generalization (see page 152) for details on grouping syntax.

`@product`

The product or component name, version, brand, or internal code or number. This attribute is equivalent to the `<prodinfo>` element.

The `@product` attribute takes a space-delimited list of values, which may or might or might not match the value of the `<prodname>` element in the prolog. The attribute also can include groups of values specified using the same syntax as generalized attributes within `@props`; see Attribute generalization (see page 152) for details on grouping syntax.

`@deliveryTarget`

The intended delivery target of the content, for example "html", "pdf", or "epub". This attribute is a replacement for the now deprecated `@print` attribute.

The `@deliveryTarget` attribute is defined in the deliveryTargetAttDomain, which is integrated into all OASIS-provided document-type shells. If this domain is not integrated into a given document-type shell, the `@deliveryTarget` attribute will not be available.

`@rev`

The identifier for the revision level. The `@rev` attribute takes a space-delimited list of values. For example, if a paragraph was changed or added during revision 1.1, the `@rev` attribute might contain the value "1.1".

`@otherprops`

Metadata qualification values about the content. This attribute is equivalent to the `<othermeta>` element.

The attribute takes a space-delimited list of values, which might or might not match the values of `<othermeta>` elements in the prolog. The attribute also can include groups of values specified using the same syntax as generalized attributes within `@props`; see Attribute generalization (see page 152) for details on grouping syntax. The grouping syntax matches the original `@otherprops` grouping syntax that was defined in DITA 1.0 and 1.1. That usage was reserved exclusively for `@otherprops`; it was deprecated in DITA 1.2.

`@props`

A generic attribute for conditional processing values. Starting with DITA 1.1, the `@props` attribute can be specialized to create new conditional processing attributes. See Attribute generalization (see page 152).

### 3.3.7.2 Examples of how conditional processing attributes are used

Each conditional processing attribute takes zero or more space-delimited string values or grouped values. For example, you can use the `@product` attribute to identify that an element applies to two particular products.

```xml
<p>Set the configuration options:
<ul>
  <li product="extendedprod">Set foo to bar</li>
</ul>
```
<ul>
  <li product="basicprod extendedprod">Set your blink rate</li>
  <li>Do some other stuff</li>
  <li>Do a special thing for Linux</li>
</ul>

Figure 17: DITA source that uses @product attribute

Groups can be defined on @audience, @product, @platform, or @otherprops using the generalized attribute syntax found in Attribute generalization (see page 152). This is intended to provide subcategories of the attribute; for example, the following could be used to indicate that a step applies to one application server and two databases. Each group consists of one or more space-delimited string values; groups cannot be nested.

```xml
<steps>
  <step><cmd>Common step</cmd></step>
  <step product="appserver(mySERVER) database(ABC dbOtherName)" />
  <cmd>Do something special for databases ABC or OtherName when installing on mySERVER</cmd>
</steps>
```

Figure 18: Grouped values in the @product attribute

If both grouped values and ungrouped values are found in a single attribute, the ungrouped values must be treated as a single group. The following values for the @product attribute are equivalent:

```
product="a database(dbA dbB) b appserver(mySERVER) c"
product="product(a b c) database(dbA dbB) appserver(mySERVER)"
```

If two groups with the same name are found in a single attribute, they should be treated as if all values are specified in the same group. The following values for the @otherprops attribute are equivalent:

```
otherprops="groupA(a b) groupA(c) groupZ(APPNAME)"
otherprops="groupA(a b c) groupZ(APPNAME)"
```

### 3.3.7.3 Evaluating conditional processing attributes

Comment by robander, 21 April 2014

Would like to add a shortdesc. Also: Deb Bissantz in review1 suggested we may want to rethink or define the term "evaluate", given that it is not used much outside of the title.

At processing time, a conditional processing profile can be used to specify values that you want to include, exclude, or flag.

For example, a publisher might want to flag information that applies to administrators, and to exclude information that applies to the extended product. The following conditional processing profile could work in conjunction with the DITA source in Figure 17: DITA source that uses product attribute (see page 90) to accomplish that:

```
<val>
  <prop att="audience" val="administrator" action="flag"/>
  <startflag><alt-text>ADMIN</alt-text></startflag>
</val>
```

At output time, the paragraph is flagged, and the first list item is excluded (since it applies to extendedprod). The second list item is still included; even though it does apply to extendedprod, it also applies to basicprod, which was not excluded.

The result should look something like the following:

**ADMIN** Set the configuration options:
• Set your blink rate
• Do some other stuff
• Do a special thing for Linux

If a conditional processing attribute is set to an empty value, for example, product="", this is equivalent to an unspecified attribute. An empty group within an attribute is equivalent to an unspecified group. For example, product="database()" is equivalent to product="", and product="database() A" is equivalent to product="A".

If two groups of the same name exist on different attributes, a rule specified for that group will evaluate the same way on each attribute. For example, if the group "sampleGroup" is specified within both @platform and @otherprops, a DITAVAL rule for sampleGroup="value" will evaluate the same in each attribute. If there is a need to distinguish between similarly named groups on different attributes, the best practice is to use more specific group names such as platformGroupname and productGroupname. Alternatively, DITAVAL rules can be specified based on the attribute name rather than the group name.

If the same group name is used in different attributes, a complex scenario could be created where different defaults are specified for different attributes, with no rule set for a group or individual value. In this case a value might end up evaluating differently in the different attributes. For example, a DITAVAL can set a default of "exclude" for @platform and a default of "include" for @product. If no rules are specified for group oddgroup(), or for the value oddgroup="edgecase", the attribute platform="oddgroup(edgecase)" will evaluate to "exclude" while product="oddgroup(edgecase)" will resolve to "include".

Comment by Kristen Eberlein, 8 February 2014
We need to clarify the wording here. "Might end up evaluating"?

3.3.7.4 Filtering

By default, values in conditional processing attributes that are not defined in a DITAVAL profile evaluate to "include". For example, if the value audience="novice" is used on a paragraph, but this value is not defined in a DITAVAL profile, the attribute evaluates to "include". However, the DITAVAL profile can change this default to "exclude", so that any value not explicitly defined in the DITAVAL profile will evaluate to "exclude". The DITAVAL profile also can be used to change the default for a single attribute; for example, it can declare that values in the @platform attribute default to "exclude", while those in the @product attribute default to include. See DITAVAL elements (see page 414) for information on how to set up a DITAVAL profile and how to change default behaviors.

When deciding whether to include or exclude a particular element, a processor should evaluate each attribute independently:

1. For each attribute:
   • If all the values in a single attribute evaluate to "exclude", the attribute evaluates to "exclude".
   • If all the values in any single group within an attribute evaluate to "exclude", the attribute evaluates to "exclude".

2. If any single attribute evaluates to exclude, the element is excluded.

For example, if a paragraph applies to three products and the publisher has chosen to exclude all of them, the processor should exclude the paragraph. This is true even if the paragraph applies to an audience or platform that is not excluded. But if the paragraph applies to an additional product that has not been excluded, then its content is still relevant for the intended output and should be preserved.

Similarly, with groups, a step might apply to one application server and two database applications:

```
<steps>
  <step><cmd>Common step</cmd></step>
```

---

DITA Version 1.3 Specification

Copyright © OASIS Open 2005, 2014. All Rights Reserved.
If a publisher decides to exclude the application server mySERVER, then the appserver() group evaluates to exclude. This may be done by setting product="mySERVER" to exclude or by setting appserver="mySERVER" to exclude. This means the step should be excluded, regardless of how the values "ABC" or "dbOtherName" evaluate. If a rule is specified for both product="mySERVER" and appserver="mySERVER", the rule for the more specific group name "appserver" takes precedence.

Similarly, if both "ABC" and "dbOtherName" evaluate to exclude, then the database() group evaluates to exclude and the element should be excluded regardless of how the "mySERVER" value is set.

In more advanced usage, a DITAVAL can be used to specify a rule for a group name. For example, an author could create a DITAVAL rule that sets product="database" to "exclude". This is equivalent to setting a default of "exclude" for any value in a database group; it would also exclude the more common usage of "database" as a single token within the @product attribute. Thus when the token "myDB" appears in a database() group within the @product attribute, the full precedence for determining whether to include or exclude the value is as follows:

1. Check for a DITAVAL rule for database="myDB"
2. Check for a DITAVAL rule for product="myDB"
3. Check for a DITAVAL rule for product="database" (default for the database group)
4. Check for a DITAVAL rule for "product" (default for the @product attribute)
5. Check for a default rule for all conditions (default of include or exclude for all attributes)
6. Otherwise, evaluate to "include"

### 3.3.7.5 Flagging

When deciding whether to flag a particular element, a processor should evaluate each value. Wherever an attribute value that has been set as flagged appears (for example, audience="administrator"), the processor should add the flag. When multiple flags apply to a single element, multiple flags should be rendered, typically in the order that they are encountered.

Flagging might be done using text (for example, bold text against a colored background) or using images. When the same element evaluates as both flagged and filtered (for example, flagged because of a value for the @audience attribute and filtered because of a value for the @product attribute value), the element should be filtered.

**Comment by Kristen Eberlein , 8 February 2014**

Should such a element also be flagged if it is evaluated to being included?

### 3.3.7.6 Conditional processing for specific output formats

By default, the content of most elements is included in all output media. The DITA map provides a means to suppress element content from appearing in print-oriented media, or from appearing in non-print-oriented media, such as HTML.

**Comment by Kristen Eberlein , 3 April 2014**

This topic needs reworking to reflect the fact that @deliveryTarget works on elements in topics, while @print works only on map elements.
Authors can specify whether individual topics or groups of topics referenced in a DITA map should be included for processing to print-oriented outputs such as PDF. Each `<map>` and `<topicref>` element supports the `@print` and `@deliveryTarget` attributes.

**Note:** Beginning with DITA 1.3, the `@print` attribute is deprecated. It is replaced with a conditional processing attribute: `@deliveryTarget`. See `@deliveryTarget` for more details.

### @print attribute

The `@print` attribute supports the following enumerated values, each of which control the way that print-oriented processors handle the inclusion or exclusion of topics or groups of topics.

<table>
<thead>
<tr>
<th>Value</th>
<th>Example</th>
<th>Print-oriented processing</th>
<th>Non-print-oriented processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unspecified (default)</td>
<td><code>&lt;topicref href=&quot;foo.dita&quot;&gt;</code></td>
<td>Topics are <strong>included</strong> in output.</td>
<td>Topics are <strong>included</strong> in output.</td>
</tr>
<tr>
<td>yes</td>
<td><code>&lt;topicref href=&quot;foo.dita&quot; print=&quot;yes&quot;&gt;</code></td>
<td>Topics are <strong>included</strong> in output.</td>
<td>Topics are <strong>included</strong> in output.</td>
</tr>
<tr>
<td>printonly</td>
<td><code>&lt;topicref href=&quot;foo.dita&quot; print=&quot;printonly&quot;&gt;</code></td>
<td>Topics are <strong>included</strong> in output.</td>
<td>Topics are <strong>excluded</strong> from the output.</td>
</tr>
<tr>
<td>no</td>
<td><code>&lt;topicref href=&quot;foo.dita&quot; print=&quot;no&quot;&gt;</code></td>
<td>Topics are <strong>excluded</strong> from the output.</td>
<td>Topics are <strong>included</strong> in output.</td>
</tr>
<tr>
<td>-dita-use-conref-target</td>
<td><code>&lt;topicref conref=&quot;#foo-topic&quot; print=&quot;-dita-use-conref-target&quot;&gt;</code></td>
<td>Topics derive a value for the <code>@print</code> attribute from the <code>@print</code> attribute of the referenced element.</td>
<td>Topics derive a value for the <code>@print</code> attribute from the <code>@print</code> attribute of the referenced element. See Using the -dita-use-conref-target value (see page 666) for more information.</td>
</tr>
</tbody>
</table>

**Note:** If a value for the `@print` is not specified explicitly in a map element, but is specified in a map that references the map element, the `@print` value cascades to the referenced map. If the `@print` value is not specified on the referencing map, a default of "yes" is assumed.

Use `print="printonly"` to identify transitional topics that you want included only in highly contextual or linear print-oriented output.

If the referenced topic should be excluded from all output formats, set the `@processing-role` attribute to "resource-only" instead of using the `@print` attribute. Content within that topic can still be referenced for display in other locations.

### @deliveryTarget attribute

The intended delivery target of the content, for example "html", "pdf", or "epub". This attribute is a replacement for the now deprecated `@print` attribute.
The @deliveryTarget attribute is defined in the deliveryTargetAttDomain, which is integrated into all OASIS-provided document-type shells. If this domain is not integrated into a given document-type shell, the @deliveryTarget attribute will not be available.

Comment by Kristen Eberlein, 2 February 2014
Content parallel to that in the preceding section needed here. I added a definition list with a conref to remind us what we currently use elsewhere.

Comment by Kristen Eberlein, 3 April 2014
The following is from comments in review #1:
Jarno Elovirta: "The @deliveryTarget is intended to replace @print. Does @deliveryTarget allow the value " -dita-use-conref-target" like @print does?"
Robert Anderson: "It is allowed on any CDATA element, but only really useful on elements with an enumerated attribute (and even then, mostly only when the attribute is required). My understanding is that this attribute is CDATA to allow for expansion to new delivery targets."

3.3.8 Branch filtering

The DITAVAL format provides a way to specify a set of conditions that are used to conditionally process DITA content. While these conditions are often stored outside of the content and applied globally by a processor, the <ditavalref> element can be used to reference a DITAVAL document in order to filter only a subset of content within or referenced by a map. This also provides the ability to process a single branch of content multiple times, while applying unique conditions to each instance of the branch.

3.3.8.1 Conditional processing (profiling) on a subset of content

The <ditavalref> element is designed to help manage conditional processing in two distinct situations. If a separate set of DITAVAL conditions is applied globally to the content, those take precedence over any conflicting conditions specified within a DITAVAL referenced by <ditavalref>.

1. A set of DITAVAL conditions is applied globally to a content set. However, one subset of information needs to be filtered using a different set of DITAVAL conditions. This is a common use case when content is aggregated from different authoring groups or companies. In this case, the <ditavalref> element can be used to selectively filter the subset of information, without affecting the rest of the content.

2. A set of content contains information that is common to all users, along with a subset of information that varies based on a condition such as audience or platform. The subset of information needs to be repeated once for each condition. For example, a set of software documentation contains common information, as well as installation instructions that vary by operating system. In this case, the <ditavalref> element can be used to publish installation instructions that are specific to each operating system, without repeating the common information that is applicable to all operating systems.

In this case, a fully resolved view of the map contains multiple instances of a single branch of content. This might result in duplicate keys, duplicate key scopes, and URIs that specify the same resource with conflicting conditions. Metadata inside of the <ditavalref> element is available to provide control over these values, so that keys, key scopes, and URIs can be individually referenced within a branch. This means that processors MUST calculate the effects of branch filtering on keys and URIs before they can establish the complete key space for a map.

It is possible for a branch with <ditavalref> already in effect to specify an additional <ditavalref>, so that a subset of the original branch receives additional filter conditions. In this situation, suffixes and prefixes are added using an outer-to-inner approach. For example, using the file name basefile.dita, a branch that added two prefixes and two suffixes would result in parentPrefix-childPrefix-basefile-childSuffix-parentSuffix.dita. In that sample, the hyphens are part of the specified prefix or suffix (they are not added automatically).
It is an error if `<ditavalref>` driven branch cloning results in multiple distinct copies of a topic having the same resolved name. Processors SHOULD report an error in such cases. Processors MAY recover by using an alternate naming scheme for the conflicting topics.

One or more `<ditavalref>` elements can appear as a child of the `<map>`, which results in the filters being applied to the entire map. Using the `<ditavalref>` element as a child of a reference to a locally-scoped map (such as within `<mapref>`) is similar, and results in the conditions being applied to the referenced map. Using the `<ditavalref>` element within a reference to a map whose `@scope` attribute evaluates to "peer" or "external" has no effect.

Examples to illustrate each of these conditions are available in Examples of branch filtering (see page 97).

### 3.3.8.2 Branch filtering: Implications of processing order

Because the branch filtering process may result in new or renamed keys, key scopes, or URIs, the full effects of the branch filtering process MUST be calculated by processors before they can construct the effective map and key scope structure.

The `@keyref` attribute and related attributes are explicitly disallowed on `<ditavalref>` for the same reason; otherwise, usage of those attributes on `<ditavalref>` would require that `@keyref` be resolved on all `<ditavalref>` elements before keys and key scopes could be established for the map as a whole.

Comment by Kristen Eberlein, 8 February 2014

Jarno Elvirta suggests that we simply state "branch filtering MUST be processed before the effective map and key space are constructed."

Jarno also suggests: "You could have a note in the spec that the spec does not mandate a given processing order or style of processing. In cases where the spec defines something using a given processing order, implementations may implement this in any way they want, but the results must match."

Comment by Kristen Eberlein, 8 February 2014

Another suggestion from Jarno Elvirta:

"Hmm... could you phrase that shortdesc so that effective map and key space construction has a dependency on branch filtering results, and that may impose processing order requirements on implementations?"

In general, the DITA specification shies away from mandating processing order, and so publication results can vary slightly depending on the order in which processes are run. In this case, any DITAVAL conditions that are specified external to the map, such as a parameter to a publishing process, take precedence over conditions referenced with the `<ditavalref>` element. For example, if the value `audience="admin"` is globally excluded by the publication process, that content will be excluded even from a branch where a `<ditavalref>` reference attempts to set `audience="admin"` to "include".

There is explicitly no requirement that global filters and filters specified with `<ditavalref>` be applied at the same time in a publishing process.

Processors should consider the following points when determining a processing order:

- If links are generated based on the map hierarchy, those links should be created using the renamed keys and URIs that result from evaluating `<ditavalref>`, to ensure that the links are consistent within the modified branches.
- If conrefs are resolved in topics before the `<ditavalref>` filter conditions are evaluated, content that applies to multiple audiences can be brought in; it later can be selectively filtered by branch. For example, if a set of installation steps is pulled in with conref (from outside the branch), it might contain information that is later filtered by platform during the `<ditavalref>` evaluation; this results in a copy of the steps that specific to
each operating system. If conref is processed after the `<ditavalref>`, content might be pulled in that has not been appropriately filtered for the new context.

- The same scenario applies to conref values that push content into the branch; pushing content prior to resolving the `<ditavalref>` conditions allows content for multiple conditions to be pushed into the branch, and then filtered by branch based on the `<ditavalref>` conditions. If the branch using `<ditavalref>` pushes content elsewhere, resolving `<ditavalref>` first could result in multiple copies of the content to be pushed (one for each branch), resulting in multiple potentially conflicting copies pushed to the new destination.

### 3.3.8.3 Examples of branch filtering

These examples illustrate the processing expectations for various scenarios involving DITAVAL references. Processing examples use either before and after markup samples or expanded syntax that shows equivalent markup without the domain.

**Example: Using a single `<ditavalref>` element to supply filtering conditions for a branch**

Consider the following DITA map and the DITAVAL file that is referenced from a `<ditavalref>` element:

```xml
<map>
  <topicref href="intro.dita"/>
  <topicref href="install.dita"/>
  <ditavalref href="novice.ditaval"/>
  <topicref href="do-stuff.dita" audience="admin"/>
  <!-- more topics -->
</map>
```

*Figure 19: input.ditamap:*

```xml
<val>
  <prop att="audience" val="novice" action="include"/>
  <prop att="audience" val="admin" action="exclude"/>
</val>
```

*Figure 20: Contents of novice.ditaval*

When this content is published, the following processing occurs:

- The first topic (intro.dita) does not use any of the conditions that are specified in novice.ditaval. It is published normally, potentially using other DITAVAL conditions that are specified externally.
- The second topic (install.dita) is filtered using any external conditions as well as the conditions that are specified in novice.ditaval.
- The third topic (do-stuff.dita) is filtered using any external conditions as well as the conditions that are specified in novice.ditaval.
- The fourth topic (advanced-stuff.dita) is removed from the map entirely, because it is filtered out with the conditions that are specified for the branch.

**Comment by robander , 21 April 2014**

Review 1: David Helfinstine asks if this is true in the case where a global set of conditions sets "admin" to include. Need to validate whether the language addresses this; my interpretation is that the global condition can only win for "exclude", not "include". I feel this was discussed in the early reviews. Otherwise it is not possible to selectively filter, even while saying "Overall at a publication level, this publication includes admin info".

None of the resources that are filtered by novice.ditaval are renamed. This ensures that external links to those topics are stable, regardless of whether the DITAVAL is used.
Example: Using multiple <ditavalref> elements to create multiple branch copies

Consider the following DITA map which contains a branch with three peer <ditavalref> elements; two of the <ditavalref> elements contain sub-elements that specify how effective resource names are constructed:

```xml
<map>
  <!-- Beginning of installing branch -->
  <topicref href="intro.dita"/>
  <topicref href="install.dita">
    <ditavalref href="win.ditaval"/>
    <ditavalmeta><dvrResourceSuffix>-apple</dvrResourceSuffix></ditavalmeta>
  </ditavalref>
  <ditavalref href="linux.ditaval">
    <ditavalmeta><dvrResourceSuffix>-linux</dvrResourceSuffix></ditavalmeta>
  </ditavalref>
  <topicref href="do-stuff.dita">
    <topicref href="mac-specific-stuff.dita" platform="mac"/>
  </topicref>
  <!-- End of installing branch -->
  <topicref href="cleanup.dita"/>
</map>
```

Figure 21: input.ditamap

```xml
<val>
  <prop att="platform" val="win" action="include"/>
  <prop att="platform" action="exclude"/>
</val>
```

Figure 22: Contents of win.ditaval

```xml
<val>
  <prop att="platform" val="mac" action="include"/>
  <prop att="platform" action="exclude"/>
</val>
```

Figure 23: Contents of mac.ditaval

```xml
<val>
  <prop att="platform" val="linux" action="include"/>
  <prop att="platform" action="exclude"/>
</val>
```

Figure 24: Contents of linux.ditaval

When a processor evaluates this markup, it results in three copies of the installing branch. The following processing takes place:

- The first topic (intro.dita) is published normally, potentially using other DITAVAL conditions that are specified externally.
- The installing branch appears three times, once for each DITAVAL. The branches are created as follows:
  - The first branch uses the first DITAVAL file (win.ditaval). Resources use their original names as specified in the map. The mac-specific-stuff.dita topic is removed. The resulting branch, with indenting to show the hierarchy, matches the original without the mac topic:
    ```
    install.dita
do-stuff.dita
    ...more topics and nested branches...
cleanup.dita
    ```
• The second branch uses the second DITAVAL file (mac.ditaval). Resources are renamed based on the `<dvrResourceSuffix>` element. The `mac-specific-stuff.dita` topic is included. The resulting branch, with indenting to show the hierarchy, is as follows:

```
install-apple.dita
do-stuff-apple.dita
mac-specific-stuff-apple.dita
...more topics and nested branches...
cleanup-apple.dita
```

• The third branch uses the last DITAVAL file (linux.ditaval). Resources are renamed based on the `<dvrResourceSuffix>` element. The `mac-specific-stuff.dita` topic is removed. The resulting branch, with indenting to show the hierarchy, is as follows:

```
install-linux.dita
do-stuff-linux.dita
...more topics and nested branches...
cleanup-linux.dita
```

The example used three DITAVAL files to avoid triple maintenance of the installing branch in a map; the following map is functionally equivalent, but it requires parallel maintenance of each branch.

```
<map>
  <topicref href="intro.dita"/>
  <!-- Windows installing branch -->
  <topicref href="install.dita">
    <ditavalref href="win.ditaval"/>
    <ditavalref href="do-stuff.dita">
      <!-- more topics and nested branches -->
    </ditavalref>
    <topicref href="cleanup.dita"/>
  </topicref>
  <!-- Mac installing branch -->
  <topicref href="install.dita">
    <ditavalref href="mac.ditaval">
      <ditavalmeta><dvrResourceSuffix>-apple</dvrResourceSuffix></ditavalmeta>
    </ditavalref>
    <topicref href="do-stuff.dita">
      <topicref href="mac-specific-stuff.dita" platform="mac"/>
      <!-- more topics and nested branches -->
    </topicref>
    <topicref href="cleanup.dita"/>
  </topicref>
  <!-- Linux installing branch -->
  <topicref href="install.dita">
    <ditavalref href="linux.ditaval">
      <ditavalmeta><dvrResourceSuffix>-linux</dvrResourceSuffix></ditavalmeta>
    </ditavalref>
    <topicref href="do-stuff.dita">
      <!-- more topics and nested branches -->
    </topicref>
    <topicref href="cleanup.dita"/>
  </topicref>
  <!-- Several chapters worth of other material -->
</map>
```

**Figure 25: input.ditamap**

**Example: Using `<ditavalref>` as a direct child of `<map>`**

The following map is equivalent to processing all contents of the map with the conditions in the `novice.ditaval` file. If additional conditions are provided externally (for example, as a parameter to the publishing process), the global conditions take precedence.

```
<map>
  <title>Sample map</title>
  <ditavalref href="novice.ditaval"/>
```

DITA Version 1.3 Specification

Copyright © OASIS Open 2005, 2014. All Rights Reserved.

Page 99 of 747
Example: Using `<ditavalref>` in a reference to a map

Similarly, in the following example, `other.ditamap` is pulled into another map. The `<ditavalref>` indicates that all of the content in `other.ditamap` should be filtered using the conditions in `some.ditaval`.

```xml
<topicref href="parent.dita">
  <topicref href="other.ditamap" format="ditamap">
    <ditavalref href="some.ditaval"/>
  </topicref>
</topicref>

Figure 26: Map fragment
```

```xml
<map>
  <topicref href="nestedTopic1.dita">
    <topicref href="nestedTopic2.dita"/>
  </topicref>
  <topicref href="nestedTopic3.dita"/>
</map>

Figure 27: Contents of `other.ditamap`
```

This markup is functionally equivalent to applying the conditions in `some.ditaval` to the topics that are referenced in the nested map. For the purposes of filtering, it could be rewritten in the following way; the extra `<topicgroup>` container is used here to ensure filtering is not applied to `parent.dita`, as it would not be in the original example:

```xml
<topicref href="parent.dita">
  <topicgroup>
    <ditavalref href="some.ditaval"/>
    <topicref href="nestedTopic1.dita">
      <topicref href="nestedTopic2.dita"/>
    </topicref>
    <topicref href="nestedTopic3.dita"/>
  </topicgroup>
</topicref>
```

For the purposes of filtering, this map could also be rewritten as:

```xml
<topicref href="parent.dita">
  <topicref href="nestedTopic1.dita">
    <ditavalref href="some.ditaval"/>
    <topicref href="nestedTopic2.dita"/>
  </topicref>
  <topicref href="nestedTopic3.dita">
    <ditavalref href="some.ditaval"/>
  </topicref>
</topicref>
```

Example: Using `<ditavalref>` within a branch that already uses `<ditavalref>`

In the following map fragment, a set of operating system conditions applies to installation instructions. Within that common branch, a subset of content applies to different audiences.

```xml
<topicref href="install.dita">
  <ditavalref href="linux.ditaval"/>
  <ditavalref href="mac.ditaval">
    <ditavalmeta><dvrResourceSuffix>-mac</dvrResourceSuffix></ditavalmeta>
  </ditavalref>
  <ditavalref href="win.ditaval">
    <ditavalmeta><dvrResourceSuffix>-win</dvrResourceSuffix></ditavalmeta>
  </ditavalref>
  <topicref href="perform-install.dita">
    <ditavalref href="linux.ditaval"/>
    <ditavalref href="mac.ditaval">
      <ditavalmeta><dvrResourceSuffix>-mac</dvrResourceSuffix></ditavalmeta>
    </ditavalref>
    <ditavalref href="win.ditaval">
      <ditavalmeta><dvrResourceSuffix>-win</dvrResourceSuffix></ditavalmeta>
    </ditavalref>
  </topicref>
</topicref>
```
In this case, the effective map contains three instances of the complete branch. The branches are filtered by operating system. Within each operating system instance, the configuration sub-branch is repeated; it is filtered once for novice users and then again for advanced users.

As a result, there are actually six instances of the configuration sub-branch:

1. The first instance is filtered using the conditions in `linux.ditaval` and `novice.ditaval`. For this instance, the resource `configure.dita` is treated as the resource `configure-novice.dita`. There is no renaming based on `linux.ditaval`, and `novice.ditaval` adds the suffix "-novice".

2. The second instance is filtered using the conditions in `linux.ditaval` and `advanced.ditaval`. For this instance, the resource `configure.dita` is treated as the resource `configure-admin.dita`. There is no renaming based on `linux.ditaval`, and `advanced.ditaval` adds the suffix "-admin".

3. The third instance is filtered using the conditions in `mac.ditaval` and `novice.ditaval`. For this instance, the resource `configure.dita` is treated as the resource `configure-mac-novice.dita`. The `mac.ditaval` resource adds the suffix "-mac", resulting in `configure-mac.dita`, and then `novice.ditaval` adds the additional suffix "-novice".

4. The fourth instance is filtered using the conditions in `mac.ditaval` and `advanced.ditaval`. For this instance, the resource `configure.dita` is treated as the resource `configure-mac-admin.dita`. The `mac.ditaval` resource adds the suffix "-mac", resulting in `configure-mac.dita`, and then `advanced.ditaval` adds the additional suffix "-admin".

5. The fifth instance is filtered using the conditions in `win.ditaval` and `novice.ditaval`. For this instance, the resource `configure.dita` is treated as the resource `configure-win-novice.dita`. The `win.ditaval` resource adds the suffix "-win", resulting in `configure-win.dita`, and then `novice.ditaval` adds the additional suffix "-novice".

6. The sixth instance is filtered using the conditions in `win.ditaval` and `advanced.ditaval`. For this instance, the resource `configure.dita` is treated as the resource `configure-win-admin.dita`. The `win.ditaval` resource adds the suffix "-win", resulting in `configure-win.dita`, and then `advanced.ditaval` adds the additional suffix "-admin".

Example: Multiple non-equivalent copies of the same file copied to the same name

The following map fragment contains several error conditions that result in name clashes:

```xml
<!-- other topics-->
</topicref>
<!!-- Begin configuration sub-branch -->
<topicref href="configure.dita">
  <ditavalref href="novice.ditaval">
    <ditavalmeta><dvrResourceSuffix>-novice</dvrResourceSuffix></ditavalmeta>
  </ditavalref>
  <ditavalref href="advanced.ditaval">
    <ditavalmeta><dvrResourceSuffix>-admin</dvrResourceSuffix></ditavalmeta>
  </ditavalref>
</topicref>
<!!-- Other config topics -->
</topicref>
<!!-- End configuration sub-branch -->
</topicref>
```

```
In this case, the effective map contains three instances of the complete branch. The branches are filtered by operating system. Within each operating system instance, the configuration sub-branch is repeated; it is filtered once for novice users and then again for advanced users.

As a result, there are actually six instances of the configuration sub-branch:

1. The first instance is filtered using the conditions in `linux.ditaval` and `novice.ditaval`. For this instance, the resource `configure.dita` is treated as the resource `configure-novice.dita`. There is no renaming based on `linux.ditaval`, and `novice.ditaval` adds the suffix "-novice".

2. The second instance is filtered using the conditions in `linux.ditaval` and `advanced.ditaval`. For this instance, the resource `configure.dita` is treated as the resource `configure-admin.dita`. There is no renaming based on `linux.ditaval`, and `advanced.ditaval` adds the suffix "-admin".

3. The third instance is filtered using the conditions in `mac.ditaval` and `novice.ditaval`. For this instance, the resource `configure.dita` is treated as the resource `configure-mac-novice.dita`. The `mac.ditaval` resource adds the suffix "-mac", resulting in `configure-mac.dita`, and then `novice.ditaval` adds the additional suffix "-novice".

4. The fourth instance is filtered using the conditions in `mac.ditaval` and `advanced.ditaval`. For this instance, the resource `configure.dita` is treated as the resource `configure-mac-admin.dita`. The `mac.ditaval` resource adds the suffix "-mac", resulting in `configure-mac.dita`, and then `advanced.ditaval` adds the additional suffix "-admin".

5. The fifth instance is filtered using the conditions in `win.ditaval` and `novice.ditaval`. For this instance, the resource `configure.dita` is treated as the resource `configure-win-novice.dita`. The `win.ditaval` resource adds the suffix "-win", resulting in `configure-win.dita`, and then `novice.ditaval` adds the additional suffix "-novice".

6. The sixth instance is filtered using the conditions in `win.ditaval` and `advanced.ditaval`. For this instance, the resource `configure.dita` is treated as the resource `configure-win-admin.dita`. The `win.ditaval` resource adds the suffix "-win", resulting in `configure-win.dita`, and then `advanced.ditaval` adds the additional suffix "-admin".

Example: Multiple non-equivalent copies of the same file copied to the same name

The following map fragment contains several error conditions that result in name clashes:

```xml
<!-- other topics-->
</topicref>
<!!-- Begin configuration sub-branch -->
<topicref href="configure.dita">
  <ditavalref href="novice.ditaval">
    <ditavalmeta><dvrResourceSuffix>-novice</dvrResourceSuffix></ditavalmeta>
  </ditavalref>
  <ditavalref href="advanced.ditaval">
    <ditavalmeta><dvrResourceSuffix>-admin</dvrResourceSuffix></ditavalmeta>
  </ditavalref>
</topicref>
<!!-- Other config topics -->
</topicref>
<!!-- End configuration sub-branch -->
</topicref>
```
In this sample, the effective map that results from evaluating the branch filter conditions will have several clashes. In some cases the same file must be processed with conflicting conditions, using the same URI. In addition, because no key scope is added or modified, keys in the branch are duplicated in such a way that only one version is available for use. Evaluating the filtered branches in this sample will result in the following equivalent map:

```html
<topicref href="a.dita" keys="a"> <!-- a.dita to be filtered by one.ditaval -->
  <topicref href="b.dita" keys="b"/>  <!-- b.dita to be filtered by one.ditaval -->
</topicref>

<topicref href="a.dita" keys="a"> <!-- a.dita to be filtered by two.ditaval; key "a" ignored -->
  <topicref href="b.dita" keys="b"/>  <!-- b.dita to be filtered by two.ditaval; key "b" ignored -->
</topicref>

<topicref href="a.dita"/>

<topicref href="c-token.dita" keys="c">
  <!-- c-token.ditaval to be filtered by one.ditaval -->
</topicref>

<topicref href="c-token.dita" keys="c">
  <!-- c-token.ditaval to be filtered by two.ditaval, key "c" ignored -->
</topicref>
```

As an additional example, in a situation where resource file names map directly to generated XHTML file names, this map results in the following conflicts:

1. a.dita generates a.html using three alternate set of conditions -- from one.ditaval, two.ditaval, and no ditaval.
2. b.dita generates b.html using two alternate set of conditions -- from one.ditaval and two.ditaval.
3. c.dita generates c-token.html using both extra sets of conditions.

### 3.3.9 Chunking

Content can be chunked (divided or merged into new output documents) in different ways for the purposes of delivering content and navigation. For example, content best authored as a set of separate topics might need to be delivered as a single Web page. A map author can use the `@chunk` attribute to split up multi-topic documents into component topics or to combine multiple topics into a single document as part of output processing.

**Examples of use**

Here are examples of how the `@chunk` attribute can be used:

**Reuse of a nested topic**

A content provider creates a set of topics as a single document. Another user wants to incorporate only one of the nested topics from the document. The new user can reference the nested topic from a DITA map, using the `@chunk` attribute to specify that the topic should be produced in its own document.

**Identification of a set of topics as a unit**

A curriculum developer wants to compose a lesson for a SCORM LMS (Learning Management System) from a set of topics without constraining reuse of those topics. The LMS can save and restore the learner's progress through the lesson if the lesson is identified as a referenceable unit. The curriculum developer defines the collection of topics with a DITA map, using the `@chunk` attribute to identify the learning module as a unit before generating the SCORM manifest.

### 3.3.9.1 Using the `@chunk` attribute

The specification defines tokens for the `@chunk` attribute that cover the most common chunking scenarios. These tokens may be used to override whatever default chunking behavior is set by a processor. Chunking is necessarily format specific, with chunked output required for some and not supported for other rendered formats.
Chunking is also implementation specific with some implementations supporting some, but not all, chunking methods, or adding new methods to the standard ones described in this specification.

The value of the @chunk attribute consists of one or more space delimited tokens. Tokens are defined in three categories: for selecting topics, for setting chunking policies, and for defining how the chunk values impact rendering. It is an error to use two tokens from the same category on a single <topicref> element.

**Selecting topics**

These values describe what portion of a target document is referenced. Such tokens are only useful when addressing a document that is made up of multiple topics. These values are ignored when the element on which they are specified does not reference a topic. Recognized values include:

- **select-topic**: The "select-topic" token is used to select an individual topic without any ancestors, descendents, or peers from within the same document.
- **select-document**: The "select-document" token is used to select all topics in the target document.
- **select-branch**: The "select-branch" token is used to select the target topic together with its descendents.

**Policies for splitting or combining documents**

Two tokens are defined for setting chunking policies. Each token applies only to the current <topicref> or <topicref> specialization, except when used on the <map> element, in which case the value establishes a policy for the entire map.

- **by-topic**: The "by-topic" token establishes a policy for the current <topicref> (or <topicref> specialization) where a separate output chunk is produced for each of the selected topics.
- **by-document**: The "by-document" token establishes a policy for the current <topicref> (or <topicref> specialization) where a single output chunk is produced for the referenced topic or topics.

**Rendering the selection**

The following tokens affect how the chunk values impact rendering of the map or topics.

- **to-content**: The "to-content" token indicates that the selection should be rendered as a new chunk of content.
  - When specified on a <topicref> or <topicref> specialization, this means that the topics selected by this <topicref> and its children will be rendered as a single chunk of content.
  - When specified on the <map> element, this indicates that the contents of all topics referenced by the map are to be rendered as a single document.
  - When specified on a <topicref> or <topicref> specialization that contains a title but no target, this indicates that processors MUST generate a title-only topic in the rendered result, along with any topics referenced by child <topicref> elements (and <topicref> specializations) of this <topicref>. The rendition address of the generated topic is determined as defined for the @copy-to attribute. If the @copy-to attribute is not specified and the <topicref> has no @id attribute, the address of the generated topic is not required to be predictable or consistent across rendition instances.

  For cross references to <topicref> elements, if the value of the @chunk attribute is "to-content" or is unspecified, the cross reference is treated as a reference to the target topic. If the reference is to a <topicref> with no target, it is treated as a reference to the generated title-only topic.

- **to-navigation (DEPRECATED)**: The "to-navigation" token is deprecated in DITA 1.3. In earlier releases, the "to-navigation" token indicate that a new chunk of navigation should be used to render the current selection (such as an individual Table of Contents or related links). When specified on the <map> element, this token indicates that the map should be presented as a single navigation chunk. If a cross reference is made to a <topicref> that has a title but no target, and the @chunk value of that <topicref> is set to "to-navigation", the resulting cross reference is treated as a reference to the rendered navigation document (such as an entry in the table of contents).

Some tokens or combinations of tokens might not be appropriate for all output types. When unsupported or conflicting tokens are encountered during output processing, processors SHOULD produce warning or error messages. Recovery from such conflicts or other errors is implementation dependent.
There is no default value for the `@chunk` attribute on most elements and the `@chunk` attribute does not cascade from container elements, meaning that the `@chunk` value on one `<topicref>` is not passed to its children. A default by-xxx policy for an entire map may be established by setting the `@chunk` attribute on the `<map>` element, which will apply to any `<topicref>` that does not specify its own by-xxx policy.

When no `@chunk` attribute values are given, chunking behavior is implementation dependent. When variations of this sort are not desired, a default for a specific map can be established by including a `@chunk` attribute value on the `<map>` element.

When creating new documents via chunk processing, the storage object name or identifier (if relevant) is determined as follows:

1. If an entire map is used to generate a single chunk (by placing `to-content` on the `<map>` element), the resource name SHOULD be taken from the resource name of the map.
2. If the `@copy-to` attribute is specified, the resource name is taken from the `@copy-to` attribute.
3. If `@copy-to` is not specified and the by-topic policy is in effect, the resource name SHOULD be taken from the `@id` attribute of the topic.
4. If `@copy-to` is not specified and the by-document policy is in effect, the resource name SHOULD be taken from the resource name of the referenced document.

When following these steps results in resource name clashes, processors MAY recover by generating alternate resource identifiers. For example, when two chunked topics use the same `@id` attribute, a processor could recover by combining the original resource name with the `@id` value instead of using only the `@id` value.

**Implementation-specific tokens and future considerations**

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

For example: “acme:level2” could be a token for the Acme DITA Toolkit that requests the “level2” chunking method.

### 3.3.9.2 Chunking examples

The following examples cover many common chunking scenarios, such as splitting one document into many rendered objects or merging many documents into one rendered object.

In the examples below, an extension of “.xxxx” is used in place of the actual extensions that will vary by output format. For example, when the output format is HTML, the extension may actually be “.html”, but this is not required.

The examples below assume the existence of the following files:

- `parent1.dita`, `parent2.dita`, etc., each containing a single topic with id P1, P2, etc.
- `child1.dita`, `child2.dita`, etc., each containing a single topic with id C1, C2, etc.
- `grandchild1.dita`, `grandchild2.dita`, etc., each containing a single topic with id GC1, GC2, etc.
- `nested1.dita`, `nested2.dita`, etc., each containing two topics: parent topics with id N1, N2, etc., and child topics with ids N1a, N2a, etc.
- `ditabase.dita`, with the following contents:

```
<ditabase xml:lang="en-us">
  <topic id="x">
    <title>Topic X</title><body><p>content</p></body>
  </topic>
  <topic id="y">
    <title>Topic Y</title><body><p>content</p></body>
  </topic>
  <topic id="y1">
    <title>Topic Y1</title><body><p>content</p></body>
  </topic>
  <topic id="y1a">
```
1. The following map causes the entire map to generate a single output chunk.

```
<map chunk="to-content">
  <topicref href="parent1.dita">
    <topicref href="child1.dita"/>
    <topicref href="child2.dita"/>
  </topicref>
</map>
```

2. The following map will generate a separate chunk for every topic in every document referenced by the map. In this case, it will result in the topics `P1.xxxx`, `N1.xxxx`, and `N1a.xxxx`.

```
<map chunk="by-topic">
  <topicref href="parent1.dita">
    <topicref href="nested1.dita"/>
  </topicref>
</map>
```

3. The following map will generate two chunks: `parent1.xxxx` will contain only topic P1, while `child1.xxxx` will contain topic C1, with topics GC1 and GC2 nested within C1.

```
<map>
  <topicref href="parent1.dita">
    <topicref href="child1.dita" chunk="to-content">
      <topicref href="grandchild1.dita"/>
      <topicref href="grandchild2.dita"/>
    </topicref>
  </topicref>
</map>
```

4. The following map breaks down portions of `ditabase.dita` into three chunks. The first chunk `Y.xxxx` will contain only the single topic Y. The second chunk `Y1.xxxx` will contain the topic Y1 along with its child Y1a. The final chunk `Y2.xxxx` will contain only the topic Y2. For navigation purposes, the chunks for Y1 and Y2 are still nested within the chunk for Y.

```
<map>
  <topicref href="ditabase.dita#Y" copy-to="Y.dita" chunk="to-content select-topic">    
    <topicref href="ditabase.dita#Y1" copy-to="Y1.dita" chunk="to-content select-branch"/>
    <topicref href="ditabase.dita#Y2" copy-to="Y2.dita" chunk="to-content select-topic"/>
  </topicref>
</map>
```

5. The following map will produce a single output chunk named `parent1.xxxx`, containing topic P1, with topic Y1 nested within P1, but without topic Y1a.

```
<map chunk="by-document">
  <topicref href="parent1.dita" chunk="to-content">
    <topicref href="ditabase.dita#Y1" chunk="select-topic"/>
  </topicref>
</map>
```
6. The following map will produce a single output chunk, `parent1.xxxx`, containing topic P1, topic Y1 nested within P1, and topic Y1a nested within Y1.

```
<map chunk="by-document">
  <topicref href="parent1.dita" chunk="to-content">
    <topicref href="ditabase.dita#Y1" chunk="select-branch"/>
  </topicref>
</map>
```

7. The following map will produce a single output chunk, `P1.xxxx`. The topic P1 will be the root topic, and topics X, Y, and Z (together with their descendents) will be nested within topic P1.

```
<map chunk="by-topic">
  <topicref href="parent1.dita" chunk="to-content">
    <topicref href="ditabase.dita#Y1" chunk="select-document"/>
  </topicref>
</map>
```

8. The following map will produce a single output chunk named `parentchunk.xxxx` containing topic P1 at the root. Topic N1 will be nested within P1, and N1a will be nested within N1.

```
<map chunk="by-document">
  <topicref href="parent1.dita" chunk="to-content" copy-to="parentchunk.dita">
    <topicref href="nested1.dita" chunk="select-branch"/>
  </topicref>
</map>
```

9. The following map will produce two output chunks. The first chunk named `parentchunk.xxxx` will contain the topics P1, C1, C3, and GC3. The "to-content" token on the reference to `child2.dita` causes that branch to begin a new chunk named `child2chunk.xxxx`, which will contain topics C2 and GC2.

```
<map chunk="by-document">
  <topicref href="parent1.dita" chunk="to-content" copy-to="parentchunk.dita">
    <topicref href="child1.dita" chunk="select-branch"/>
    <topicref href="child2.dita" chunk="to-content select-branch" copy-to="child2chunk.dita">
      <topicref href="grandchild2.dita"/>
    </topicref>
    <topicref href="child3.dita"/>
    <topicref href="grandchild3.dita" chunk="select-branch"/>
  </topicref>
</map>
```

10. The following map produces a single chunk named `nestedchunk.xxxx`, which contains topic N1 with no topics nested within.

```
<map>
  <topicref href="nested1.dita#N1" copy-to="nestedchunk.dita" chunk="to-content select-topic"/>
</map>
```

11. In DITA 1.3, the "to-navigation" chunk is deprecated. In earlier releases, the following map produced two navigation chunks, one for P1, C1, and the other topic references nested under `parent1.dita`, and a second for P2, C2, and the other topic references nested under `parent2.dita`.

```
<map>
  <topicref href="parent1.dita" navtitle="How to set up a web server" chunk="to-navigation">
    <topicref href="child1.dita" chunk="select-branch"/>
  </topicref>
  <topicref href="child2.dita" navtitle="How to ensure database security"/>
</map>
```
3.3.10 Translation and localization

DITA has features that facilitate preparing content for translation and working with multilingual content, including the `<xml:lang>` attribute, the `@dir` attribute, and the `@translate` attribute. In addition, the `<index-sort-as>` element provides support for index sorting in languages in which the index sort order must be modified by the author or translator.

3.3.10.1 The `@xml:lang` attribute

The `@xml:lang` attribute specifies the language and (optional) locale of the element content. The `@xml:lang` attribute applies to all attributes and content of the element where it is specified, unless it is overridden with `@xml:lang` on another element within that content.

The `@xml:lang` attribute SHOULD be explicitly set on the root element of each map and topic.

Setting the `@xml:lang` attribute in the DITA source ensures that processors handle content in a language- and locale-appropriate way. If the `@xml:lang` attribute is not set, processors assume a default value which might not be appropriate for the DITA content. When the `@xml:lang` attribute is specified for a document, DITA processors MUST use the specified value to determine the language of the document.

Setting the `@xml:lang` attribute in the source language document facilitates the translation process; it enables translation tools (or translators) to simply change the value of the existing `@xml:lang` attribute to the value of the target language. Some translation tools support changing the value of an existing `@xml:lang` attribute, but they do not support adding new markup to the document that is being translated. Therefore, if source language content does not set the `@xml:lang` attribute, it might be difficult or impossible for the translator to add the `@xml:lang` attribute to the translated document.

If the root element of a map or a top-level topic has no value, 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.

The `@xml:lang` attribute is described in the XML Recommendation. Note that the recommended style for the `@xml:lang` attribute is lowercase language and (optional) uppercase, separated by a hyphen, for example, "en-US" or "sp-SP" or "fr".

Recommended use in topics

For a DITA topic that contains a single language, set the `@xml:lang` attribute on the highest-level element that contains content.

When a DITA topic contains more than one language, set the `@xml:lang` attribute on the highest-level element to specify the primary language and locale that applies to the topic. If part of a topic is written in a different language, authors should ensure that the part is enclosed in an element with the `@xml:lang` attribute set appropriately. This method of overriding the default document language applies to both block and inline elements that use the alternate language. Processors SHOULD style each element in a way that is appropriate for its language as identified by the `@xml:lang` attribute.

Recommended use in maps

The `@xml:lang` attribute can be specified on the `<map>` element. The `@xml:lang` attribute cascades within the map in the same way that it cascades within a topic. However, since the `@xml:lang` attribute is an inherent property of the XML document, the value of the `@xml:lang` attribute does not cascade from one map to another.
or from a map to a topic; the value of the @xml:lang attribute that is specified in a map does not override @xml:lang values that are specified in other maps or in topics.

The primary language for the map SHOULD be set on the @map element. The specified language remains in effect for all child <topicref> elements, unless a child specifies a different value for the @xml:lang attribute.

When no @xml:lang value is supplied locally or on an ancestor, a processor-determined default value is assumed.

**Recommended use with the @conref or @conkeyref attribute**

When a @conref or @conkeyref attribute is used to include content from one element into another, the processor MUST use the effective value of the @xml:lang attribute from the referenced element, that is, the element that contains the content. If the referenced element does not have an explicit value for the @xml:lang attribute, the processor SHOULD default to using the same value that is used for topics that do not set the @xml:lang attribute.

This behavior is shown in the following example, where the value of the @xml:lang attribute of the included note is obtained from its parent <section> element that sets the @xml:lang attribute to "fr". When the `installingAcme.dita` topic is processed, the <note> element with the @id attribute set to "mynote" has an effective value for the @xml:lang attribute of "fr".

```xml
<?xml version="1.0"?>
<!DOCTYPE task PUBLIC "//OASIS//DTD DITA Task//EN" "task.dtd">
<task xml:lang="en" id="install_acme">
  <title>Installing Acme</title>
  <shortdesc>Step-by-step details about how to install Acme.</shortdesc>
  <taskbody>
    <prereq>
      <p>Special notes when installing Acme in France:</p>
      <note id="mynote" conref="warningsAcme.dita#topic_warnings/frenchwarnings"/>
    </prereq>
  </taskbody>
</task>
```

**Figure 28: installingAcme.dita**

```xml
<?xml version="1.0"?>
<!DOCTYPE topic PUBLIC "//OASIS//DTD DITA Topic//EN" "topic.dtd">
<topic id="topic_warnings">
  <title>Warnings</title>
  <body>
    <section id="qqwwee" xml:lang="fr">
      <title>French warnings</title>
      <p>These are our French warnings.</p>
      <note id="frenchwarnings">Note in French!</note>
    </section>
    <section xml:lang="en">
      <title>English warnings</title>
      <p>These are our English warnings.</p>
      <note id="englishwarnings">Note in English!</note>
    </section>
  </body>
</topic>
```

**Figure 29: warningsAcme.dita**

### 3.3.10.2 The @dir attribute

The @dir attribute provides direction to processors about how bi-directional text should be rendered.

Bi-directional text is text that contains text in both text directionalities, right-to-left (RTL) and left-to-right (LTR). For example, languages such as Arabic, Hebrew, Farsi, Urdu, and Yiddish have text written from right-to-left; however, numerics and embedded sections of Western language text are written from left to right. Some multilingual documents also contain a mixture of text segments in two directions.

DITA contains the following attributes that have an effect on bi-directional text processing:
@xml:lang
Identifies the language and locale, and so can be used to identify text that requires bi-directional rendering.

@dir
Identifies or overrides the text directionality. It can be set to "ltr", "rtl", "lro", or "rlo"

In general, properly-written mixed text does not need any special markers; the Unicode bidirectional algorithm positions the punctuation correctly for a given language. The processor is responsible for displaying the text properly. However, some rendering systems might need directions for displaying bidirectional text, such as Arabic, properly. For example, Apache FOP might not render Arabic properly unless the left-to-right and right-to-left indicators are used.

The use of the @dir attribute and the Unicode algorithm is explained in the article Specifying the direction of text and tables: the dir attribute (http://www.w3.org/TR/html4/struct/dirlang.html#adef-dir) . This article contains several examples of how to use the @dir attribute set to either "ltr" or "rtl". There is no example of setting the @dir attribute to either "lro" or "rlo", although it can be inferred from the example that uses the <bdo> element, a now-deprecated W3C mechanism for overriding the entire Unicode bidirectional algorithm.

Comment by Kristen Eberlein, 3 February 2014
The <xref> goes to the a portion of the W3C HTML 4.01 Specification, not an article with the given title. It also does mention overrides.

Recommended usage

The @dir attribute, together with the @xml:lang attribute, is essential for rendering table columns and definition lists in the proper order.

In general text, the Unicode Bidirectional algorithm, as specified by the @xml:lang attribute together with the @dir attribute, provides for various levels of bidirectionality:

- Directionality is either explicitly specified via the xml:lang attribute in combination with the @dir attribute on the highest level element (topic or derived peer for topics, map for ditamaps) or assumed by the processing application. If used, the @dir attribute SHOULD be specified on the highest level element in the topic or document element of the map.
- When embedding a right-to-left text run inside a left-to-right text run (or vice-versa), the default direction might provide incorrect results based on the rendering mechanism, especially if the embedded text run includes punctuation that is located at one end of the embedded text run. Unicode defines spaces and punctuation as having neutral directionality and defines directionality for these neutral characters when they appear between characters having a strong directionality (most characters that are not spaces or punctuation). While the default direction is often sufficient to determine the correct directionality of the language, sometimes it renders the characters incorrectly (for example, a question mark at the end of a Hebrew question might appear at the beginning of the question instead of at the end or a parenthesis might render incorrectly). To control this behavior, the @dir attribute is set to "ltr" or "rtl" as needed, to ensure that the desired direction is applied to the characters that have neutral bidirectionality. The "ltr" and "rtl" values override only the neutral characters (for example, spaces and punctuation), not all Unicode characters.

  Note: Problems with Unicode rendering can be caused by the rendering mechanism. The problems are not due to the XML markup itself.

- Sometimes you might want to override the default directionality for strongly bidirectional characters. Overrides are done using the "lro" and "rlo" values, which overrides the Unicode Bidirectional algorithm. This override forces a direction on the contents of the element. These override attributes give the author a brute force way of setting the directionality independent of the Unicode Bidirectional algorithm. The gentler "ltr" and "rtl" values have a less radical effect, only affecting punctuation and other so-called neutral characters.

For most authoring needs, the "ltr" and "rtl" values are sufficient. Use the override values only when you cannot achieve the desired effect using the the "ltr" and "rtl" values.
Implementation precautions

Comment by Kristen Eberlein, 30 January 2014
The actual content of this section is not about "Implementation precautions". It specifies what the DITA TC expects from processors.

Applications that process DITA documents, whether at the authoring, translation, publishing, or any other stage, SHOULD fully support the Unicode bidirectional algorithm to correctly implement the script and directionality for each language that is used in the document.

Comment by Kristen Eberlein, 30 January 2014
Let's define a meaning for processors early on -- probably in the "Terminology" topic -- and then we can avoid descriptive verbage like this.

Applications SHOULD ensure every highest-level topic element and the root map element explicitly assign the @dir and @xml:lang attribute.

Comment by Kristen Eberlein, 30 January 2014
Can we replace "Applications SHOULD ensure every highest-level topic element and the root map element explicitly assign the @dir and @xml:lang attribute." with "Applications SHOULD ensure that every topic and map explicitly assigns the @dir and @xml:lang attribute on the root element."

Related Links
What you need to know about the BIDI algorithm and inline markup (http://www.w3.org/International/articles/inline-bidi-markup/)
XHTML Bi-directional Text Attribute Module (http://www.w3.org/TR/2004/WD-xhtml2-20040722/mod-bidi.html)
Specifying the direction of text and tables: the dir attribute (http://www.w3.org/TR/html4/struct/dirlang.html#adef-dir)
HTML 4.0 Common Attributes (http://www.htmlhelp.com/reference/html40/attrs.html)

3.3.11 Context hooks for help systems and other user assistance

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

Context hook and windowing information is ignored if the processor (the software producing the help or user assistance files) does not support this metadata.

Comment by Kristen Eberlein, 8 February 2014
I'd like to move forward with a standard definition of processor. In the above sentence, are we referring to all processors (for example, DITA-OT, DITA2Go, Ditac, RoboHelp) or are we referring to processors specifically oriented towards generating online help, such as RoboHelp? BTW, I use RoboHelp as an example because it is the online help application that I know can generate output from DITA topics.

User interfaces for software application often are linked to user assistance (such as help systems and tool tips) through context hooks. Context hooks are simply identifiers that associate a part of the user interface with the location of a help topic. Context hooks can be direct links to URIs, but more they are indirect links that use numeric context identifiers and context strings, in conjunction with an intermediary mapping file, to associate the context hook with the location of the help topic. A similar mapping file is used in the software application to associate the user interface controls with the context hook. A help API typically is used to simplify the process for both the software developer and help author. The mapping files that are used to associate user interface
controls and context hooks, and context hooks and help topics, are known as header files, map files, or alias files, depending upon the help API.

Context hooks can define either one-to-one or one-to-many relationships with user interface controls and topics. Context hook information is defined within DITA topics and DITA maps through attributes of the `<resourceid>` element:

@appid
An ID that is used by an application to identify the topic.

@ux-context-string
Contains the value of a user-assistance context-string that is used to identify the topic.

@ux-source-priority
Contains a value that indicates the precedence of context hooks in the map and context hooks in the topic. This makes it possible to avoid problems where context hooks defined in the DITA map potentially conflict with those defined in the topics; the values of the `@ux-source-priority` attribute indicates how potential conflicts should be resolved. This usage is defined only when the attribute is used on a `<resourceid>` within a DITA map; using this attribute on a `<resourceid>` within a topic is undefined and reserved for future use.

@ux-windowref
Contains the name of the window to be used to display the help topic. The window characteristics are separately defined in a `<ux-window>` element in the DITA map.

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

The `<ux-window>` element provides attributes of `@top`, `@left`, `@height`, `@width`, `@on-top`, `@features`, `@relative`, and `@full-screen`.

### 3.3.12 Sorting

Processors can be configured to sort elements. Common sort processing includes glossary entries that are sorted to produce automatically-generated or automatically-sorted glossary lists, lists of parameters or reference entries in custom navigation structures, tables sorted based on the contents of cells in a specific column or row, and more.

Each element to be sorted must have some inherent text on which it will be sorted. This text is the base sort phrase for the element. For elements that have titles, the base sort phrase usually is the content of the `<title>` element. For elements that do not have titles, the base sort phrase might be literal content in the DITA source, or it might be generated or constructed based on the semantics of the element involved; for example, it could be constructed from various attribute or metadata values. Processors that perform sorting SHOULD explicitly document how the base sort phrase is determined for a given element.

The `<sort-as>` element can be used to specify an effective sort phrase when the base sort phrase is not appropriate for sorting. For index terms, the `<index-sort-as>` element can be used to specify the effective index entry.

The details of sorting and grouping are implementation specific. Processors might provide different mechanisms for defining or configuring collation and grouping details. Even where the `<sort-as>` element is specified, two processors might produce different sorted and grouped results because they might use different collation and grouping rules. For example, one processor might be configured to sort English terms before non-English terms, while another might be configured to sort them after. The grouping and sorting of content is subject to local editorial rules.

Related Links
4.5.3.5 sort-as (see page 363)

For elements that are sorted, the `<sort-as>` element provides text that is combined with the base sort phrase to construct the effective sort phrase. The text can be specified in the content of the `<sort-as>` element or in the `@value` attribute on the `<sort-as>` element. The `<sort-as>` element is useful for elements where the base sort phrase is inadequate or non-existent, for example, a glossary entry for a Japanese Kanji phrase.

4.4.2.5 index-sort-as (see page 342)

The `<index-sort-as>` element specifies a sort phrase under which an index entry would be sorted.

3.4 Configuration, specialization, and constraints

The extension facilities of DITA allow existing vocabulary and constraint modules to be combined to create specific DITA document types. Additionally, vocabulary modules can be extended to create more-specialized markup to meet new requirements not satisfied by existing markup.

If you want to create a specialization or constraint using the DITA XSD schema, you need to use the XSD redefine facility. However, this facility does not allow sequence groups to be directly constrained. Thus, to support specialization and constraints, it may be necessary to refactor content models into named groups that can be redefined. In order to keep the XSD, RELAX NG, and DTD implementations as consistent as possible, the DITA Technical Committee only refactored those content models that were required for OASIS-provided schemas. These were the strict task body and machinery-industry task model. The other DITA content models distributed by OASIS have not been refactored.

You MAY modify the OASIS-provided XSD module to refactor the content model if required by your constraint. You can either modify the OASIS-provided XSD module or extend the RELAX NG-to-XSD generation process to add your special case. You SHOULD notify the DITA Technical Committee of your constraint requirement, so the TC can consider adding the required refactoring to the OASIS-provided XSDs.

Comment by Kristen Eberlein , 7 October 2014

This is not the correct place for this information. I added it here so that we do not forget to include it in the spec.

3.4.1 Overview of DITA extension facilities

DITA provides three extension facilities: configuration, constraint, and specialization.

**Configuration**

Configuration enables the definition of DITA document types that include only the vocabulary modules that are required for a given set of documents. There is no need to modify the vocabulary modules. Configurations are implemented as document type shells.

**Constraint**

Constraint enables the unilateral modification of content models and attribute lists for individual elements. There is no need to modify the vocabulary modules. Constraints are implemented as constraint modules, which are integrated into document type shells.

**Specialization**

Specialization enables the creation of new element types. Specializations are implemented as vocabulary modules, which are integrated into document type shells.

Specialization hierarchies are implemented as sets of vocabulary modules, each of which declares the markup and entities that are unique to a specialization. The separation of the markup vocabulary and its implementing declarations into modules makes it easy to extend the hierarchy, because new modules can be added without affecting existing document types. It also makes it easy to assemble design elements from different sources into a single document-type shell and makes it easy to reuse specific parts of the specialization hierarchy in more than one document type shell.
**Governed document-type declarations**

DITA documents are governed by DITA document types that represent the combination of one or more structural types (maps or topics), domain vocabularies, and constraint modules; such document types define the set of element types and attributes that are available to a specific document. In short, DITA provides a framework by which XML vocabulary and constraint modules can be combined in an infinite number of ways to create specific document types, as well as a set of base modules that serve as the base for further configuration, constraint, or specialization.

DITA documents are typically governed by a conforming DITA document-type shell. However, the conformance of a DITA document is a function of the document instance, not its governing grammar. Therefore conforming DITA documents are not required to use a conforming document-type shell.

There might be compelling or practical reasons to use non-conforming document-type shells. For example, a document might use a document type shell that does not conform to the DITA requirements for shells in order to meet the needs of a specific application or tool. Such a non-conforming document-type shell still might enable the creation of conforming DITA content.

### 3.4.1.1 Recognized XML-document grammar mechanisms

The DITA standard recognizes three XML-document grammar mechanisms by which conforming DITA vocabulary modules and document types can be constructed: document type declarations (DTDs), XML Schema declarations (XSDs), and RELAX NG grammars.

This specification defines implementation requirements for all of these document grammar mechanisms. The OASIS DITA Technical Committee recognizes that other XML grammar languages might provide similar modularity and extensibility mechanisms. However, the Technical Committee has not yet defined implementation requirements for those languages so their conformance cannot be determined.

Of these three document grammar mechanisms, RELAX NG grammars offer the easiest-to-use syntax and the most expressive constraints. For this reason, the RELAX NG definitions of the standard DITA vocabularies are the normative versions from which the DTD and XSD versions are automatically generated. Open-source tools are being developed to generate conforming DTD- and XSD-based document type shells, vocabulary modules, and constraint modules from RELAX NG grammars.

**Related Links**

[Tools for generating DTD or XSD from RELAX NG](#)

### 3.4.1.2 Using RELAX NG for document type shells, vocabulary modules, and constraint modules

The RELAX NG specification defines two syntaxes for RELAX NG grammars: the XML syntax and the compact syntax. The two syntaxes are functionally equivalent, and either syntax can be reliably converted into the other, for example, by using the open-source Trang tool. The OASIS DITA Technical Committee uses the RELAX NG XML syntax for the normative versions of the XML grammar files that comprise the DITA release.

The DITA Technical Committee chose the RELAX NG XML syntax for the following reasons:

**Easy use of foreign markup**

The DITA coding requirements depend on this feature of RELAX NG in order to capture metadata about document type shells and modules; such metadata is required in order to generate the DTD- and XSD-based versions of the grammar files.

The foreign vocabulary feature also can be used to include Schematron rules directly in RELAX NG grammars. Schematron rules can check for patterns that either are not expressible with RELAX NG directly or that would be inconvenient to express.

**RELAX NG <div> element**

---

*Copyright © OASIS Open 2005, 2014. All Rights Reserved.*
This general grouping element allows for arbitrary organization and grouping of patterns within grammar documents. Such grouping tends to make the grammar documents easier to work with, especially in XML-aware editors.

**Capability of expressing precise restrictions**

RELAX NG is capable of expressing constraints that are more precise than is possible with either DTDs or XSDs. For example, RELAX NG patterns can be context specific such that the same element type can allow different content or attributes in different contexts.

If you plan to generate DTD- or XSD-based modules from RELAX NG modules, avoid RELAX NG features that cannot be translated into DTD or XSD constructs. When RELAX NG is used directly for DITA document validation, the document type shells for those documents can integrate constraint modules that use the full power of RELAX NG to enforce constraints that cannot be enforced by DTDs or XSDs.

The DITA use of RELAX NG depends on the RELAX NG DTD Compatibility specification, which provides a mechanism for defining default attribute values and embedded documentation. Processors that use RELAX NG for DITA documents in which required attributes (for example, the @domains and @class attributes) are not explicitly present must implement the DTD compatibility specification in order to get default attribute values.

DITA practitioners can author DITA modules using one RELAX NG syntax, and then use tools to generate modules in the other syntax. The resulting RELAX NG modules are conforming if there is a one-to-one file correspondence. The resulting RELAX NG XML might not contain the metadata that is necessary to generate corresponding modules that are DTD- or XSD-based, as processors are not required to process the annotations and information from foreign vocabularies. Conforming RELAX NG-based DITA modules, whether compact or XML syntax, MAY omit the annotations and foreign elements that are required to enable generation of other XML grammars, such as DTDs and XML Schema.

The DITA coding requirements are defined for the RELAX NG XML syntax. Document type shells, vocabulary modules, and constraint modules that use the RELAX NG compact syntax MUST use the same file naming and organization requirements as those defined for the RELAX NG XML syntax, substituting ".rnc" for ".rng" as the file name extension.

RELAX NG grammars for DITA document-type shells, vocabulary modules, and constraint modules MAY do the following:

- Use the `<a:documentation>` element anywhere that foreign elements are allowed by RELAX NG. The `<a:documentation>` element refers to the `<documentation>` element type from the http://relaxng.org/ns/compatibility/annotations/1.0 as defined by the DTD compatibility specification. The prefix "a" is used by convention.
- Use `<div>` to group pattern declarations.
- Include embedded Schematron rules or any other foreign vocabulary. Processors MAY ignore any foreign vocabularies within DITA grammars that are not in the http://relaxng.org/ns/compatibility/annotations/1.0 or http://dita.oasis-open.org/architecture/2005/ namespaces.

DITA grammar documents SHOULD contain the following:

- An initial set of comments or a `<moduleMetadata>` element that describes the grammar document and indicates the URNs or absolute URLs by which the grammar should be referenced in RELAX NG grammar references. See Module description markup for use in DITA-specific RELAX NG grammars (see page 115).
- Comments or `<div>` elements that identify each section of the grammar
- Sections that occur in the order specified in these coding requirements
- (For section titles that use `<div>` elements) The following format:

  ```xml
  <div>
    <a:documentation>SECTION TITLE</a:documentation>
  </div>
  ```
3.4.1.3 Generating DTD and XSD modules from RELAX NG

RELAX NG grammars for DITA document-type shells, vocabulary modules, and constraint modules can be used to generate DTD- and XSD-based document-type shells and vocabulary modules. This requires that the RELAX NG grammar documents contain DITA-specific metadata that defines the module type, module header comment, namespace prefixes, and public identifiers for the components to be generated. All RELAX NG grammars defined by the DITA specification include this metadata.

RELAX NG provides the simplest syntax for creating document type shells. This makes RELAX NG ideally suited as the base format for defining DITA vocabulary. However, since many DITA-aware tools require the use of DTDs or XSD schemas, there is a practical requirement to generate DTDs and XSD schemas from RELAX NG.

The generation of DTD and XSD modules from RELAX NG requires additional metadata that is not inherent in the grammar definition:

- The module type (document type shell, vocabulary module, or constraint module)
- The header comment for the file
- The short name for vocabulary and constraint modules
- The public identifiers to use for each component file to be generated
- The @domains attribute contribution for vocabulary and constraint modules
- (For foreign vocabularies) The DTD and XSD files to include
- (For foreign vocabularies) The namespace prefix to use for DTD declarations

This DITA-required metadata can be defined using the following mechanisms:

- A `<moduleDesc>` element within the RELAX NG `<grammar>` element
- Attributes in the DITA architecture namespace on specific declarations

The `<moduleDesc>` element enables specification of the metadata that is required in order to generate conforming DTD- and XSD-based modules from the RELAX NG grammars.

The use of `<moduleDesc>` is not required. However, when it is used, the `<moduleDesc>` element MUST be specified as a direct child of the RELAX NG `<grammar>` element and SHOULD be the first child of `<grammar>`. The `<moduleDesc>` element functions as a foreign element as defined in the RELAX NG specification. The `<moduleDesc>` element is defined as a specialization of `<topic>`. Because it is in a namespace, it cannot be a strictly-conforming DITA topic as DITA elements may not be in a namespace. However, if the `<moduleDesc>` element is processed in terms of the `@class` attributes, it will be recognized as a DITA topic and otherwise conforms to all rules for conforming topic specializations.

The `<moduleDesc>` element type is defined in the `vocabularyModuleDescMod.rng` vocabulary module file.
Namespace

The `<moduleDesc>` element is in the DITA architecture namespace: http://dita.oasis-open.org/architecture/2005/.

`<moduleDesc>` element type

`<moduleDesc>`

Contains the module description metadata. It contains the following sub-elements:

`<moduleTitle>`

Provides a descriptive title for the module or document type shell. It contains text.

`<headerComment>`

Contains the header comment for the module. The content is text with whitespace preserved (@xml:space value of "preserve"). The header comment is used as the header for all generated components.

`<moduleMetadata>`

Contains additional metadata for the module or document type shell. It contain the following sub-elements:

`<moduleType>`

Indicates the module type. It contains one of the following keywords:

attributedomain

  The grammar defines an attribute domain.

base

  The grammar is one of the DITA base modules. Base modules can only be defined by the DITA Technical Committee. Examples of base modules are the commonElementsMod.rng and metaDeclMod.rng modules.

constraint

  The grammar defines a constraint module.

elementdomain

  The grammar defines an element domain.

map

  The grammar defines a map type.

mapshell

  The grammar is a map type shell for a map type.

topic

  The grammar defines a topic type.

topicshell

  The grammar is a document type shell for a topic type.

`<modulePublicIds>`

For vocabulary and constraint modules, defines the public IDs for the files that are generated from the grammar. Each sub-element specifies the public identifier to use for a specific generated file. It contains the following sub-elements:

`<dtdMod>`

  The public identifier for the DTD-syntax .mod file to be generated from the grammar.

`<dtdEnt>`

  The public identifier for the DTD-syntax .ent file to be generated from the grammar. This element is required for element domain modules; it is not used for attribute domain modules.

`<rncMod>`
The public identifier for the RELAX NG compact syntax (RNC) module file to be generated from the grammar.

<rngMod>

The public identifier for the RELAX NG XML syntax (RNG) module file (the file that contains the <moduleMetadata> element).

<xsdMod>

The public identifier for the XSD schema module file to be generated from the grammar.

<domainsContribution>

Required for modules, not used for shells. Defines the @domains attribute contribution for the module, for example, "(@topic hi-d)". This is the value that MUST be added to the @domains value for root map or topic types that integrate the module.

<shellPublicIds>

For document type shells, defines the public IDs for the files that are generated from the grammar. Each sub-element specifies the public identifier to use for a specific generated file. It contains the following sub-elements:

<dtShell>

The public identifier for the DTD document type shell to be generated from the grammar.

<rncShell>

The public identifier for the RELAX NG compact syntax (RNC) document type shell to be generated from the grammar.

<rngShell>

The public identifier for the RNG document type shell (the file that contains the <moduleMetadata> element).

<xsdShell>

The public identifier for the XSD schema document type shell to be generated from the grammar.

Example: RELAX NG grammars with module metadata

The following code provides the descriptive metadata for a vocabulary module for a topic type:

```xml
<?xml version="1.0" encoding="UTF-8"?>
        xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
        xmlns="http://relaxng.org/ns/structure/1.0"
        datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
  <moduleDesc xmlns="http://dita.oasis-open.org/architecture/2005/">
    <moduleTitle>DITA Module Description Module</moduleTitle>
    <headerComment><![CDATA[
    HEADER
    -----------------------------------------------
    MODULE:    DITA Module Description Module
    VERSION:   1.3
    DATE:      September 2013
  <!-- ... -->]]></headerComment>
    <modulePublicIds>
      <dtdMod>-//OASIS//ELEMENTS DITA Vocabulary Module Description//EN</dtdMod>
      <dtdEnt>-//OASIS//ENTITIES DITA Vocabulary Module Description//EN</dtdEnt>
      <xsdMod>urn:oasis:names:tc:dita:xsd:vocabularyModuleDescMod.xsd</xsdMod>
      <rncMod>urn:oasis:names:tc:dita:rnc:vocabularyModuleDescMod.rnc</rncMod>
      <rngMod>urn:oasis:names:tc:dita:rng:vocabularyModuleDescMod.rng</rngMod>
    </modulePublicIds>
  </moduleDesc>
</grammar>
```
The following code provides the descriptive metadata for a document type shell for a topic type:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="vocabularyModuleDesc.rng" schematypens="http://relaxng.org/ns/structure/1.0"/>
    xmlns="http://relaxng.org/ns/structure/1.0"
    xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
    datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
    <moduleDesc xmlns="http://dita.oasis-open.org/architecture/2005/">
        <moduleTitle>DITA Module Description Module</moduleTitle>
        <headerComment><![CDATA[
            =======================================================
            HEADER
            =======================================================
            MODULE:    DITA Module Description Topic Type Shell
            VERSION:   1.3
            DATE:      September 2013
        ]]></headerComment>
        <moduleMetadata>
            <moduleType>topicshell</moduleType>
            <moduleShortName>vocabularyModule</moduleShortName>
            <shellPublicIds>
                <dtdShell>http://oasis/DTD DITA Vocabulary Module Description//EN</dtdShell>
                <xsdShell>urn:oasis:names:tc:dita:xsd:vocabularyModuleDesc.xsd</xsdShell>
            </shellPublicIds>
        </moduleMetadata>
    </moduleDesc>
</grammar>
```

### 3.4.2 Configuration (Document type shells)

A given DITA map or topic document is governed by a DITA document type that defines the set of structural modules (topic or map types), domain modules, and constraints modules that the map or topic can use.

A DITA document type is defined by the following:

- The set of modules that are declared in the `@domains` attribute on the root element of the document
- The values of the `@class` attributes of all the elements in the document

If the `@domains` attribute declares both structural and domain vocabulary modules, then the `@domains` attribute alone can serve to define the DITA document type. The information on the `@domains` and `@class` attributes is sufficient to implement all DITA-defined processing and constraint checking on documents (for example, determining if a referenced element in a content reference has a set of modules that are compatible with the modules that are used by the referencing element's document).

Thus, DITA does not require that conforming DITA documents have an associated document-type definition as long as all required attributes are explicit in the document instances. However, most DITA documents have an associated XML-schema document. Such documents enable validation using normal XML processors; they also can provide default values for the `@domains` and `@class` attributes. While the DITA specification only defines coding requirements for DTD, RELAX NG, and XML Schema documents, conforming DITA documents MAY use other document-type constraint languages, such as Schematron.
According to the coding requirements for DITA document types, a document type shell always is implemented as a top-level file that only includes and configures vocabulary modules; a document type shell never directly defines new element or attribute types.

Two document type shells define the same DITA document type if they integrate the same set of vocabulary and constraint modules. For example, a document type shell that is an unmodified copy of the OASIS-provided, document type shell for topic (topic.dtd, topic.xsd, topic.rng, or topic.rnc) defines the same DITA document type as the original document-type shell. However, the new document type shell has the following differences:

- It is a distinct file that is stored in a different location.
- (If it uses a system identifier) It has a distinct system identifier.
- (If it has a public identifier) It has a unique public identifier.

Document type shells that are not provided by OASIS MUST have a unique public identifier, if public identifiers are used. Document type shells that are not provided by OASIS MUST NOT indicate OASIS as the owner; the public identifier or URN for such document type shells SHOULD reflect the owner or creator of the document type shell. For example, if example.com creates a copy of the document type shell for topic, an appropriate public identifier would be "//example.com/DTD DITA Topic//EN", where "example.com" is the owner identifier component of the public identifier. An appropriate URN would be "urn:example.com:names:dita:xsd:topic.xsd".

Note: The public or system identifier that is associated with a given document type shell is not, by itself, necessarily distinguishing. This is because two different document-type shells, owned by different owners, can define the same DITA document type as indicated by the effective value of the @domains attribute.

Comment by Kristen Eberlein, 15 February 2014

What is the point of this note? It repeats information previously covered.

Eberlein, 5 April: Dick Hamilton made the following comment in review #1:

"I agree with Kris. Also, if it is kept, shouldn't it be a non-normative note (or is that implied by calling it a note)."

While the DITA specification includes a starter set of document type shells, these document type shells are not mandatory.

Note: Even if an initial implementation does not require configuration, constraint, or specialization, it can be useful to create new document-type shells. That way, if modification is required in the future, documents will not need to be modified to point to a new document-type shell.

DITA document-type shells MUST follow the coding requirements that are defined in this specification. This ensures consistency of implementation.

3.4.2.1 DTD document-type shell: Coding requirements

A document type shell integrates one or more topic type or map type modules, zero or more domain modules, and zero or more constraint modules. A DTD document type shell is organized into sections, where each section contains a specific type of declaration.

DTD document type shells MUST NOT directly declare element types or attributes. A DTD document type shell MUST conform to the following coding requirements.

Each section of the shell is introduced by a comment. Shells SHOULD use these comments to identify each section of the shell. Each section SHOULD be present in the shell DTD, even if the section contains no declarations, and MUST occur in the order they are presented here. The ordering is required by the XML rules for entity declaration precedence and also serve to enable automatic shell creation and modification. Shells
SHOULD have an initial set of comments that describe the shell and indicate the public identifiers, URNs, or absolute URLs by which the shell should be referenced in DOCTYPE declarations.

**Topic or map entity inclusions**

The topic or map entity declarations section includes the .ent file for the top-level topic or map type the shell is configuring.

Topic shells SHOULD use the comment:

```xml
<!-- ============================================================= -->
<!--                    TOPIC ENTITY DECLARATIONS                  -->
<!-- ============================================================= -->
```

Map shells SHOULD use the comment:

```xml
<!-- ============================================================= -->
<!--                    MAP ENTITY DECLARATIONS                    -->
<!-- ============================================================= -->
```

This section MUST declare and reference as an external parameter entity the .ent file for the topic or map module where the entity is named %typename-dec. For example:

```xml
<!ENTITY % concept-dec
  PUBLIC "-//OASIS//ENTITIES DITA 1.3 Concept//EN"
  "concept.ent">
%concept-dec;
```

This section MUST declare and reference as an external parameter entity the .ent file for any additional structural specialization modules on which the primary specialization depends; the entity is named %typename-dec. For example, if a troubleshooting specialization is specialized from topic but includes elements from task, it will need to include the .ent file for task:

```xml
<!ENTITY % task-dec
  PUBLIC "-//OASIS//ENTITIES DITA 1.3 Task//EN"
  "task.ent">
%task-dec;
```

**Domain entity inclusions**

The domain entity inclusions section includes the entity declaration files for each element domain integrated by the document type, as well as for structural modules whose ancestry needs to be declared in the @domains attribute. This section SHOULD use the comment:

```xml
<!-- ============================================================= -->
<!--                    DOMAIN ENTITY DECLARATIONS                 -->
<!-- ============================================================= -->
```

For each element domain included in the shell, this section MUST declare an external parameter entity for the domain's entity declaration file and immediately reference the entity. The entity name for the domain declaration consists of the domain name plus the dec suffix. In the following example, the entity file for the highlight domain is included in the document type shell:

```xml
<!ENTITY % hi-d-dec PUBLIC
  "-//OASIS//ENTITIES DITA Highlight Domain//EN"
  "highlightDomain.ent">
%hi-d-dec;
```

**Domain attribute inclusions**
The domain attribute inclusions section includes the entity declaration files for each attribute domain integrated by the document type. This section SHOULD use the comment:

```xml
<!-- ============================================================= -->
<!--                    DOMAIN ATTRIBUTE DECLARATIONS              -->
<!-- ============================================================= -->
```

For each attribute domain included in the shell, this section MUST declare an external parameter entity for the domain's entity declaration file and immediately reference the entity. The entity name for the domain declaration consists of the domain name plus the `ent` suffix. In the following example, the entity file for a new attribute domain is included in the document type shell:

```xml
<!ENTITY % newAtt-d-dec PUBLIC
    "-/My Company//ENTITIES New Attribute Domain//EN"
    "newAttDomain.ent"
>%newAtt-d-dec;
```

**Element extension redefinitions**

The element extension redefinition section contains redefinitions of element name parameter entities to reflect the integration of domain-provided element types into base content models. This section SHOULD use the comment:

```xml
<!-- ============================================================= -->
<!--                    DOMAIN EXTENSIONS                          -->
<!-- ============================================================= -->
```

For each element that is extended by one or more domains, the document type shell redefines the entity for the element. The new definition is a disjunctive list of alternatives comprising the literal name of the element followed by the element extension entity from each domain that is providing specializations. In the following example, the entity for the `<pre>` element is redefined to allow specializations from the programming, software, and user interface domains:

```xml
<!ENTITY % pre        "pre        |
    %pr-d-pre; |
    %sw-d-pre; |
    %ui-d-pre;">
```

The value of the entity can omit any base types from which other types listed are specialized. For example, the preceding example could omit the `<pre>` element, effectively allowing only specializations of `<pre>`, but not `<pre>` itself:

```xml
<!ENTITY % pre        "%pr-d-pre; |
    %sw-d-pre; |
    %ui-d-pre;">
```

**Note:** Omitting base types from domain extensions constitutes a form of constraint. The constraint MUST be represented by a constraint module that declares the `@domains` attribute declaration for the constraint. For the omission of `<pre>` in the preceding example the constraint might be called "noBasePre-c" and would be declared in a file named "noBasePreConstraint.mod", containing the following declarations:

```xml
<!ENTITY noBasePre-c-pre  "%pr-d-pre; | %sw-d-pre; | %ui-d-pre;">
<!ENTITY noBasePre-c-att  "(topic noBasePre-c)" >
<!ENTITY % pre          "%noBasePre-c-pre ;">
```

**Attribute extension redefinitions**
The attribute extension redefinition section integrates the declarations of specializations of the base and props attributes (defined in attribute domain modules included in the attribute domain inclusion section). This section must use the comment:

```xml
<!-- ============================================================= -->
<!--                    DOMAIN ATTRIBUTE EXTENSIONS                -->
<!-- ============================================================= -->
```

The entities for extending the props and base attributes have a null value by default:

```xml
<!ENTITY % props-attribute-extensions  "" >
<!ENTITY % base-attribute-extensions   "" >
```

For each attribute domain included by the shell, the shell **MUST** redefine the entity that is extended. The new definition is a list of the attribute extension entities for the domains that are providing specializations.

```xml
<!ENTITY % props-attribute-extensions "%newAtt-d-attribute;
%othernewAtt-d-attribute;">  
<!ENTITY % base-attribute-extensions "%newfrombaseAtt-d-attribute;
%othernewfrombaseAtt-d-attribute;">  
```

**Topic nesting redefinitions**

The topic nesting section contains redefinitions of the topic nesting control parameter entities defined by the topic modules integrated in the shell. This section **SHOULD** use the comment:

```xml
<!-- ============================================================= -->
<!--                    TOPIC NESTING OVERRIDES                    -->
<!-- ============================================================= -->
```

For each topic type integrated in the shell, the document type shell can control nesting of subtopics by redefining the `topictype-info-types` entity. The definition is usually an OR list of topic types that can be nested in the corresponding parent topic type. Use the literal root element name of each topic, not the corresponding element entity, as in the following example:

```xml
<!ENTITY % concept-info-types "concept | myTopicType">  
```

The document type shell can also set the default for most topic types by defining the global `info-types` entity, for example:

```xml
<!ENTITY % info-types "concept | myTopicType">  
```

**Domain declaration redefinition**

The domain declaration redefinition section sets the effective value of the `@domains` attribute for the topic or map type modules integrated into the shell. This section **SHOULD** use the comment:

```xml
<!-- ============================================================= -->
<!--                    DOMAINS ATTRIBUTE OVERRIDE                 -->
<!-- ============================================================= -->
```

The document type shell **MUST** redefine the `included-domains` entity to list the values for element and attribute domains, constraints, and structural specializations that are either included in the document type or have elements that are reused in the document type. In the following example, entities are included for both the troubleshooting specialization and for the task specialization on which troubleshooting depends; for the
element domains hi-d, ut-d, ui-d, pr-d, and sw-d; for the attribute domain newAtt-d; for the constraint noBasePre-c; and; :

```xml
<!ENTITY included-domains "&troubleshooting-att;
&task-att;
&hi-d-att;
&ut-d-att;
&ui-d-att;
&pr-d-att;
&sw-d-att;
&newAtt-d-att;
&noBasePre-c-ph;"
>
```

For a domain or structural module, the @domains attribute value entity is declared in the .ent file. For constraint modules, the @domains attribute value entity is declared in the module’s .mod file since constraint modules do not use separate .ent files.

**Content constraint module inclusions**

The content constraint module inclusion section includes constraint modules that override the base content models for structural or domain types integrated in the shell. This section SHOULD use the comment:

```xml
<!-- ==============================================================
  <!--                    CONTENT CONSTRAINT INTEGRATION             
  ============================================================== -->

For each constraint module integrated in the shell, the shell MUST declare an external parameter entity for the constraint’s .mod file and immediately reference the entity. The entity name for the constraint declaration consists of the constraint module name plus the -c-def suffix. For example, this constraint inclusion for the task topic type constrains the DITA 1.2 relaxed task content model to match the more constrained DITA 1.1 task content model:

```xml
<!ENTITY % strictTaskbody-c-def PUBLIC "-//OASIS//ELEMENTS DITA 1.2 Strict Taskbody Constraint//EN"
  "strictTaskbodyConstraint.mod"
>%strictTaskbody-c-def;
```

**Structural definition inclusions**

The structural definition inclusion section includes the element type declaration (.mod) files for each topic or map type integrated into the shell. For topic shells, this section SHOULD use the comment:

```xml
<!-- ==============================================================
  <!--                    TOPIC ELEMENT INTEGRATION                  
  ============================================================== -->

For map shells, this section SHOULD use the comment:

```xml
<!-- ==============================================================
  <!--                    MAP ELEMENT INTEGRATION                    
  ============================================================== -->

For each structural type integrated in the document type, the document type shell MUST declare and reference an external parameter entity for the structural type module’s .mod file. The entity name consists of the name of the structural type plus a -type suffix. For example:

```xml
<!ENTITY % topic-type PUBLIC "-//OASIS//ELEMENTS DITA Topic//EN"
  "topic.mod"
>%topic-type;
```
Modules should be included in ancestry order, starting with topic or map. Where modules need to be included for non-ancestor dependencies, they should be declared immediately before the structural type module that references them.

For example, a specialization called `cppAPIRef` that specialized from `reference` would include the modules for `topic`, `reference`, and `cppAPIRef` in that order.

When a structural module includes a dependency on elements from another module that is not part of its ancestry, then the module it depends on should be included immediately before the module that depends on it, along with any ancestor modules not already included. For example, if a specialization called `cppProgrammingTask` specialized from `task` and needed to include some elements from `cppAPIRef`, then it would include the modules for `topic`, `task`, `reference`, `cppAPIRef`, and `cppProgrammingTask` in that order.

### Element domain definition inclusions

The element domain definition inclusion section includes the element definition files for each element domain integrated into the shell. This section *SHOULD* use the comment:

```xml
<!-- --------------------------------------------------------------- -->
<!--                    DOMAIN ELEMENT INTEGRATION                 -->
<!-- --------------------------------------------------------------- -->
```

For each element domain used in the document type, the document type shell *MUST* declare and reference an external parameter entity for the domain definition module file (`mod`). The entity name consists of the domain name plus a `-def` suffix. For example:

```xml
<!ENTITY % hi-d-def PUBLIC
    "-//OASIS//ELEMENTS DITA Highlight Domain//EN"
  "highlightDomain.mod"
>%hi-d-def;
```

### 3.4.2.2 XSD document-type shell: Coding requirements

A shell document type integrates one or more topic type or map type modules, zero or more domain modules, and zero or more constraint modules. A shell XSD is organized into sections, where each section contains a specific type of declaration.

An XSD document type shell *MUST* conform to the following coding requirements. XSD document type shells *MUST NOT* directly declare element types or attributes (except for the `@domains` attribute, which always reflects the details of the domains and structural types integrated by the shell).

DITA XSDs use the XML Schema redefine feature (xs:redefine) to override base group definitions for content models and attribute lists. This facility is analogous to the parameter entities used for DTDs. Unlike DTD parameter entities, an xs:redefine both includes the XSD file it redefines and holds the redefinition applied to the groups in the included XSD file. Thus, for XSD files that define groups, the file may be included via xs:include if it is used without modification or via xs:redefine if any of its groups are redefined.

Shell XSDs are organized into sections. Each section of the shell XSD is introduced by a comment. Shells *SHOULD* use these comments to identify each section of the shell. Each section *SHOULD* be present in the shell XSD, even if the section contains no declarations, and *SHOULD* occur in the order they are presented here. Shell XSDs *SHOULD* have an initial set of comments that describe the shell and indicate the URNs or absolute URLs by which the shell should be referenced from document instances or otherwise associated with documents. Shell XSDs *SHOULD* use the XSD appinfo and documentation elements to contain additional documentation about the shell.

### Element domain inclusions
The element domain inclusion section contains includes of each element domain integrated by the shell. This section SHOULD use the comment:

```xml
<!-- ================ ELEMENT DOMAINS ===================== -->
```

For each element domain used by the map or topic type, the shell XSD must have an `xs:include` element that includes the XSD module for that domain. For example:

```xml
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:programmingDomain.xsd:1.2"/>
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:softwareDomain.xsd:1.2"/>
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:highlightDomain.xsd:1.2"/>
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:uiDomain.xsd:1.2"/>
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:indexingDomain.xsd:1.2"/>
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:utilitiesDomain.xsd:1.2"/>
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:hazardstatementDomain.xsd:1.2"/>
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:abbreviateDomain.xsd:1.2"/>
```

Attribute domain inclusions

The attribute domain inclusion section contains includes of each attribute domain integrated by the shell. This section SHOULD use the comment:

```xml
<!-- ================ ATTRIBUTE DOMAINS ===================== -->
```

For each attribute domain used by the map or topic type, the shell XSD must have an `xs:include` element that includes the XSD module for that domain. For example:

```xml
<xs:include schemaLocation="urn:example.com:dita:domains:newAtt.xsd"/>
```

Group inclusions

The group inclusion section contains includes or redefinitions of the group definitions for the structural types integrated in the shell. Group redefinitions are used to integrate domain-provided element and attribute types into base content models. This section SHOULD use the comment:

```xml
<!-- ================ GROUP DEFINITIONS ===================== -->
```

For both map and topic shells, this section MUST include or redefine the common element group, the metadata declaration group, and the table model group.

For topic shells, this section MUST include or redefine the group XSD for each topic type used by the shell. For example, from a shell for the task topic type:

```xml
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:taskGrp.xsd:1.2"/>
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:metaDeclGrp.xsd:1.2"/>
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:tblDeclGrp.xsd:1.2"/>
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:topicGrp.xsd:1.2"/>
```

For map shells, this section MUST include or redefine the group XSD for each map type used by the shell (that is, the module for the specialization of `<map>` the shell uses, as well as any ancestor map types from which the shell's map element is specialized). For example, from the learningMap shell:

```xml
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:tblDeclGrp.xsd:1.2"/>
<xs:redefine schemaLocation="urn:oasis:names:tc:dita:xsd:mapGrp.xsd:1.2">
  <xs:group name="topicref">
    <xs:choice>
      <xs:group ref="topicref"/>
      <xs:group ref="mapgroup-d-topicref"/>
    </xs:choice>
  </xs:group>
</xs:redefine>
```
For each element extended by one or more domains, the document type shell **MUST** redefine the model group for the element to a list of alternatives including the literal name of the element and the element extension model group from each domain that is providing specializations. To integrate a new domain in the document type shell use the schema `<redefine>` mechanism to manage the number of domains used by the document type shell. The model group requires a reference to itself to extend the base model group. To see an example, look at the `topic.xsd` schema document.

```
<xsd:group name="pre">
  <xsd:choice>
    <xsd:group ref="pre" />
    <xsd:group ref="pr-d-pre" />
    <xsd:group ref="ul-d-pre" />
    <xsd:group ref="sw-d-pre" />
  </xsd:choice>
</xsd:group>
```

To add domains to a new structural type you can copy the contents of the parent structural type domains schema document into the document type shell. Add or remove the model group from the new domain to the appropriate named group.

```
<xsd:group name="pre">
  <xsd:choice>
    <xsd:group ref="pre" />
    <xsd:group ref="pr-d-pre" />
    <xsd:group ref="domainName-d-element"/>
  </xsd:choice>
</xsd:group>
```

For each attribute extended by one or more domains, the document type shell **MUST** redefine the attribute extension model group for the attribute to a list of alternatives including the literal name of the attribute and the attribute extension model group from each domain that is providing specializations. To integrate a new attribute domain in the document type shell use the schema `<redefine>` mechanism to manage the number of attribute domains used by the document type shell.

```
<xsd:attributeGroup name="props-attribute-extensions">
  <xsd:attributeGroup ref="props-attribute-extensions"/>
  <xsd:attributeGroup ref="newAtt-d-attribute"/>
  <xsd:attributeGroup ref="othernewAtt-d-attribute"/>
</xsd:attributeGroup>
```

```
<xsd:attributeGroup name="base-attribute-extensions">
  <xsd:attributeGroup ref="base-attribute-extensions"/>
  <xsd:attributeGroup ref="newfrombaseAtt-d-attribute"/>
</xsd:attributeGroup>
```
Module inclusions

The module inclusion section includes the module XSD files for the structural types used in the shell. This section SHOULD use the comment:

```xml
<!-- =================  MODULE INCLUDE DEFINITION  ==================  -->
```

For each map or topic type used by the shell, this section MUST include either the module XSD file for that type or a constraint module for that type. It MUST also include any other module XSD files required by the topic or map types, normally the common element module, meta declaration module, and table declaration module. For example:

```xml
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:commonElementMod.xsd:1.2"/>
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:tblDeclMod.xsd:1.2"/>
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:metaDeclMod.xsd:1.2"/>
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:topicMod.xsd:1.2"/>
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:conceptMod.xsd:1.2"/>
```

Structural definition inclusions

The structural definition inclusion section includes the element type declaration (*Mod.xsd) files for each topic or map type integrated into the shell. For topic shells, this section SHOULD use the comment:

```xml
<!-- ============================================================= -->
<!--                    TOPIC ELEMENT INTEGRATION                  -->
<!-- ============================================================= -->
```

For map shells, this section SHOULD use the comment:

```xml
<!-- ============================================================= -->
<!--                    MAP ELEMENT INTEGRATION                    -->
<!-- ============================================================= -->
```

For example, if a troubleshooting specialization is specialized from topic but includes elements from task, then the task structural model MUST be included in the document shell as well as its structural dependency on the @domains attribute. See domains attribute (see page 146) for details.

```xml
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:taskMod.xsd:1.2"/>
```

Comment by robander, October 6 2014

Following sentence needs to be fixed, not sure of intent - "When getting the domains attribute value from the [element] from the [file] files ..."

Note: When getting the @domains attribute value from the <dita:domainsModule> from the *Mod.xsd files you will need to add the element type hierarchy information for the structural definition to the base value. For instance in the troubleshooting specialization case you would need to add the following to the @domains attribute:

```xml
<xs:attributeGroup name="domains-att">
    <xs:attribute name="domains" type="xs:string"
        default="(topic troubleshooting+task) (topic task) etc..."
    />
</xs:attributeGroup>
```

Domains attribute declaration
The @domains attribute declaration section contains the declaration of the @domains attribute for the shell. This section SHOULD use the comment:

```xml
<!-- ------------------------ DOMAINS ATTRIBUTE DECLARATION ------------------------ -->
```

The shell must declare the @domains attribute such that the @domains attribute value reflects each vocabulary module and constraint module integrated by the shell. The declaration has the form:

```xml
<xs:attributeGroup name="domains-att">
  <xs:attribute name="domains" type="xs:string"
    default="domain usage declarations"/>
</xs:attributeGroup>
```

Where *domain usage declarations* is a sequence of domain usage specifications (see *domains attribute* (see page 146) for details). For example, from the learningMap shell:

```xml
<xs:attributeGroup name="domains-att">
  <xs:attribute name="domains" type="xs:string"
    default="(map mapgroup-d)
    (topic delay-d)
    (topic indexing-d)
    (topic learningmeta-d)
    (map learningmap-d)"
/>  
</xs:attributeGroup>
```

### Info types definition

Each topic type defines an info types group that defines the default set of allowed subordinate topics for that topic type. Topic shells may redefine this group to change the effective set of allowed subordinate topics.

The info types section contains the definition of the effective value of the info types groups for topics used by the shell. This section SHOULD use the comment:

```xml
<!-- ------------------------ INFO TYPES DEFINITION ------------------------ -->
```

This section MUST NOT be included in map shells.

The shell MUST define a model group with the name `info-types`. This model group can define a list of allowed subordinate topics. If the topic type should not allow subordinate topics, then the default value for the `info-types` model group MUST be defined as an empty group, as follows:

```xml
<xs:group name="info-types">
  <xs:sequence/>
</xs:group>
```

The document type shell can control how topics are allowed to nest within specific topic types by redefining the topic-type-specific info types group, named `topictype-info-types`. The info-types group is declared in the module XSD file for a given topic type. For example, in a shell for the concept topic type, allowing concept or generic topic to nest within concept:

```xml
<xs:redefine schemaLocation="urn:oasis:names:tc:dita:xsd:conceptMod.xsd:1.2">
  <xs:group name="concept-info-types">
    <xs:choice>
      <xs:group ref="concept-info-types"/>
      <xs:group ref="topic"/>
    </xs:choice>
  </xs:group>
</xs:redefine>
```

Note that XSD rules require that the redefined group include a reference to itself in addition to any other components specified for the redefined group.
3.4.2.3 RELAX NG document-type shell: Coding requirements

A document type shell integrates one or more topic type or map type modules, zero or more domain modules, and zero or more constraint modules.

RELAX NG document-type shells MUST NOT directly declare element types or attributes. A RELAX NG document-type shell MUST conform to the following coding requirements, as well as the requirements defined in *Using RELAX NG for document type shells, vocabulary modules, and constraint modules* (see page 113).

RELAX NG modules are self-integrating, so that document type shells only need to include vocabulary modules; there is no separate specification required in order to integrate domain and nested topic elements into the base content models. Also, constraint modules simply override the patterns that they constrain directly in the constraint module itself. However, the default value for the @domains attribute cannot automatically be constructed using RELAX NG facilities. Thus, the @domains attribute value has to be directly specified in the document type shell. In addition, the names of element types that define @id attributes of type "ID" must be listed in the redefinition of the "any" pattern that is required in each document type shell.

![Note: For modules that use the <moduleDesc> element, the @domains attribute contribution SHOULD be in the <domainsContribution> element. This enables tools to automatically construct the @domains attribute value in document type shells as an aid to authoring or automatically generating document type shells.](image)

Comment by Kristen Eberlein, 15 February 2014
Why is this a note?

Constraint modules are used by importing the constraint module rather than the module that the constraint modifies. Constraint modules refer to the base module to be constrained and redefine patterns as needed to implement the constraint. In addition, you can disallow base types extended by domains by overriding the base type’s pattern in the document type shell within the reference to the domain module for the domain. In this case, the constraint represented by the pattern redefinition MUST be declared in the @domains attribute. The @domains contribution for the constraint can be documented using a <domainsContribution> element from the DITA namespace. For example:

```xml
<div>
  <a:documentation>MODULE INCLUSIONS</a:documentation>
  <include href="topicMod.rng"/>
  <include href="hazardstatementDomainMod.rng"/>
  <include href="highlightDomainMod.rng">
    <ditaarch:domainsContribution>
      (topic hi-d-noUnderline-c)
    </ditaarch:domainsContribution>
    <define name="u">
      <notAllowed></notAllowed>
    </define>
  </include>
  <include href="indexingDomainMod.rng"/>
  <include href="utilitiesDomainMod.rng"/>
</div>
```

Comment by Kristen Eberlein, 15 February 2014

Let's be clear about what the above codeblock is illustrating. Is it a constraint module? A document type shell that integrates a constraint module that removes <u> from the highlighting domain?

[Eberlein, 5 April 2014: Eliot added the following comments in review #1:

"The example is exactly what the lead-in paragraph describes: a shell that integrates a domain and imposes a constraint directly within the domain inclusion, declaring the domains contribution within the include for the domain module being constrained."
Dick Hamilton added: "I agree, but I don't think it would hurt to introduce the example with something like this: "The following example shows how to document the @domains contribution in a constraint module that removes the &lt;u&gt; element from the highlight domain:"

Eliot then commented: "It is a shell that has an in-place constraint as described in the immediately-preceding paragraph. It is demonstrating the use of the @ditaarch:domainsContribution attribute to declare the constraint for use in @domains."

Root element declaration

Document type shells must use the RELAX NG start declaration to specify the root element of the document type. The &lt;start&gt; element MUST define exactly one root element, and that element MUST be defined using a reference to a tagname.element pattern. The root element declaration SHOULD start with the header "ROOT ELEMENT DECLARATION".

For example:

```xml
<div>
  <a:documentation>ROOT ELEMENT DECLARATION</a:documentation>
  <start combine="choice">
    <ref name="topic.element"/>
  </start>
</div>
```

Module inclusions

The document type shell must include at least the module for the map or topic type that the shell is configuring. It also must include the modules for any domain or additional structural modules. The document type shell MUST include any domain or structural modules that are named as dependencies in the @domains attribute value. The module inclusion section SHOULD start with the header "MODULE INCLUSIONS".

For example:

```xml
<div>
  <a:documentation>MODULE INCLUSIONS</a:documentation>
  <include href="topicMod.rng"/>
  <include href="highlightDomainMod.rng"/>
  <include href="utilitiesDomainMod.rng"/>
  <include href="indexingDomainMod.rng"/>
  <include href="hazardstatementDomainMod.rng"/>
</div>
```

Override of "any" patterns

The "any" pattern is used as content for the &lt;foreign&gt;, &lt;unknown&gt; and &lt;required-cleanup&gt; elements. The "any" pattern should accept any element with any content or attributes. The RELAX NG DTD compatibility processing rules require that any elements that match two or more patterns have consistent declarations for all attributes with respect to their ID type. In DITA, &lt;topic&gt;, &lt;map&gt;, &lt;anchor&gt;, and all specializations of those element declare the @id attribute as an XML ID (datatype "ID"). This requires that shells override the "any" pattern to exclude these element types from the pattern that matches any element name with any attributes and any content, because this pattern will match the @id attribute without specifying that it is an XML ID, thus resulting in an ID type conflict. These elements should be added explicitly to the choice group that defines the "any" pattern, and this is done by contributing these elements to the "idElements" pattern. The set of element types that have to be reflected in this pattern depends on the set of map or topic types and domains integrated by the shell, but at a minimum, any topic types or map types have to be listed. For maps, the &lt;anchor&gt; element and any specializations of &lt;anchor&gt; have to be listed. In addition, foreign vocabularies integrated through specializations of &lt;foreign&gt; may also define similar patterns that require the same accommodation (see Specializing foreign or unknown content (see page 153)).

Comment by Kristen Eberlein , 15 February 2014

How do we want to markup a specific pattern? &lt;markupname&gt; element with an @outputclass value?
The pattern “idElements” is defined in the commonElementsMod.rng module as an empty pattern. Modules that define element types with @id attributes of type ID then contribute to this pattern (see RELAX NG: Coding requirements for element type declarations (see page 169)).

This section SHOULD start with the header “OVERRIDE OF "any" PATTERNS”.

The general structure of this pattern is:

```xml
<doc>
  <a:documentation>OVERRIDE OF "any" PATTERNS</a:documentation>
  <define name="any">
    <zeroOrMore>
      <choice>
        <ref name="idElements"/>
        <element>
          <anyName>
            <except>
              <name>tagname</name>
            </except>
          </anyName>
          <zeroOrMore>
            <attribute>
              <anyName/>
            </attribute>
            <text/>
          </zeroOrMore>
        </element>
      </choice>
    </zeroOrMore>
  </define>
</doc>
```

Where the `<name>` element within the `<except>` element is repeated for each element type that is used from the document type shell that defines the @id attribute as type "ID".

Comment by Kristen Eberlein, 6 April 2014

Comments from review #1:

“Eliot Kimber: "We may need to say something about providing for foreign domains, which require you to include exclusions for the namespaces used by those domains."

Rewrite suggested by Dick Hamilton: "Repeat the `<name>` element within the `<except>` for each element type in the document type shell that defines the @id attribute as type "ID".

Example: Document type shell for a topic information type

The following example illustrates the RELAX NG markup used to construct a document type shell for the strict task information type:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
  xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0">
<moduleDesc xmlns="http://dita.oasis-open.org/architecture/2005/">
</moduleDesc>
</grammar>
```
<grammar>

<!-- ============================================================= -->
<!--                 ROOT ELEMENT DECLARATION                      -->
<!-- ============================================================= -->
<start>
  <ref name="task.element"/>
</start>

<!-- ============================================================= -->
<!--                 DITA DOMAINS ATTRIBUTE                        -->
<!-- ============================================================= -->
<define name="domains-att" combine="interleave">
  <optional>
    <attribute name="domains"
      a:defaultValue=""
      topic task strictTaskbody-c
      topic task
      topic abbrev-d
      topic hazard-d
      topic hi-d
      topic indexing-d
      topic pr-d
      topic sw-d
      topic ui-d
      topic ut-d"
    />
  </optional>
</define>

<!-- ============================================================= -->
<!--                 MODULE INCLUSIONS                             -->
<!-- ============================================================= -->
<include href="strictTaskbodyConstraintMod.rng"/>
<include href="abbreviateDomainMod.rng"/>
<include href="/base/rng/hazardstatementDomainMod.rng"/>
<include href="/base/rng/highlightDomainMod.rng"/>
<include href="/base/rng/indexingDomainMod.rng"/>
<include href="programmingDomainMod.rng"/>
<include href="softwareDomainMod.rng"/>
<include href="uiDomainMod.rng"/>
<include href="/base/rng/utilitiesDomainMod.rng"/>

<!-- ============================================================= -->
<!--                 ID-DEFINING-ELEMENT OVERRIDES                 -->
<!-- ============================================================= -->
<define name="any">
  <zeroOrMore>
    <choice>
      <ref name="idElements"/>
      <element>
        <anyName>
          <except>
            <name>topic</name>
            <name>task</name>
          </except>
        </anyName>
        <zeroOrMore>
          <attribute>
            <anyName/>
            <attribute/>
            <zeroOrMore>
              <ref name="any"/>
            </zeroOrMore>
          </element>
          <text/>
        </choice>
      </element>
    </zeroOrMore>
  </choice>
</define>
</grammar>
Example: Document type shell for a map information type

The following example illustrates the RELAX NG markup used to construct a document type shell for the base map information type:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="../../../checkShell.sch" schematypens="http://purl.oclc.org/dsd1/schematron"/>
<?xml-model href="../../../../vocabularyModuleDesc/rng/vocabularyModuleDesc.rng"
schematypens="http://relaxng.org/ns/structure/1.0"/>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0">
  <moduleDesc xmlns="http://dita.oasis-open.org/architecture/2005/">
    <moduleTitle>DITA Highlight Domain</moduleTitle>
    <headerComment xml:space="preserve">
      ==================================================
      HEADER
      ==============================================================
      MODULE:    DITA Base Map DTD (only base domains)
      VERSION:   1.2
      DATE:      April 2010
      ==============================================================
      ==============================================================
      PUBLIC DOCUMENT TYPE DEFINITION
      TYPICAL INVOCATION
      Refer to this file by the following public identifier or an
      appropriate system identifier:
      PUBLIC "-/OASIS/DTD DITA Base Map//EN"
      Delivered as file "basemap.dtd"
      The public ID above refers to the latest version of this DTD.
      To refer to this specific version, you may use this value:
      PUBLIC "-/OASIS/DTD DITA 1.2 Base Map//EN"
      ==============================================================
      SYSTEM:     Darwin Information Typing Architecture (DITA)
      PURPOSE:    DTD to describe DITA maps
      ORIGINAL CREATION DATE:
      April 2010
      (C) Copyright OASIS Open 2010
      All Rights Reserved.
      UPDATES:
      2010.09.20 RDA: Add topic-based domains
      ==============================================================
    </headerComment>
    <moduleMetadata>
      <moduleType>mapshell</moduleType>
      <moduleShortName>basemap</moduleShortName>
      <shellPublicIds>
        <dtdShell>-//OASIS//DTD DITA Base Map//EN</dtdShell>
        <xsdShell>urn:oasis:names:tc:dita:xsd:basemap.xsd</xsdShell>
      </shellPublicIds>
    </moduleMetadata>
  </moduleDesc>
  <div>
    <a:documentation>ROOT ELEMENT DECLARATION</a:documentation>
    <start>
      <ref name="map.element"/>
    </start>
  </div>
  <div>
    <a:documentation>DITA DOMAINS ATTRIBUTE</a:documentation>
    <define name="domains-att">
      <optional>
        <attribute name="domains"
          a:defaultValue=""
          topic delay=d>
```
### 3.4.2.4 DTD public identifiers

Each document-type shell (.dtd file) or module component (.mod or .ent file) has a public identifier. The public identifier can reference either the latest version or a specific version of the document-type shell or module component.

**Format**

The public identifiers use the following format:

```
"-//OASIS//DTD DITA version information-type//EN"
```

where:

- **version** either is a DITA version number (for example, 1.0, 1.1, 1.2, or 1.3) or is omitted entirely.
- **information-type** is the name of the topic or map type in camel case, for example, Concept or BookMap.

Note that "OASIS" is the owner identifier; this indicates that the artifacts are owned by OASIS. The keyword "DITA" is a convention that indicates that the artifact is DITA-related.
Public identifiers for document-type shells that are not defined by the OASIS DITA Technical Committee SHOULD use an owner identifier that unambiguously indicates the owner of the shell, for example, "ACME" or "EXAMPLE>COM".

The entity resolution catalog that is part of the DITA DTD package lists the full set of document-type shells, module components, and their associated public identifiers.

**Topic and topic-based specializations**

The identifiers listed below are declared by the OASIS DITA Technical Committee for the modules that are shipped with DITA 1.3. Values that use a version number of 1.3 refer specifically to the modules that are delivered with the DITA 1.3 specification. Values that use a version number of 1.x refer to the modules in the latest approved DITA 1.x specification. Values without a version number refer to the modules in the latest approved DITA specification, regardless of version.

**Base topic**

PUBLIC "-//OASIS//DTD DITA Base Topic//EN" "basetopic.dtd"
PUBLIC "-//OASIS//DTD DITA 1.3 Base Topic//EN" "basetopic.dtd"
PUBLIC "-//OASIS//DTD DITA 1.x Base Topic//EN" "basetopic.dtd"

**Concept**

PUBLIC "-//OASIS//DTD DITA Concept//EN" "concept.dtd"
PUBLIC "-//OASIS//DTD DITA 1.3 Concept//EN" "concept.dtd"
PUBLIC "-//OASIS//DTD DITA 1.x Concept//EN" "concept.dtd"

**DITAbase**

PUBLIC "-//OASIS//DTD DITA Composite//EN" "ditabase.dtd"
PUBLIC "-//OASIS//DTD DITA 1.3 Composite//EN" "ditabase.dtd"
PUBLIC "-//OASIS//DTD DITA 1.x Composite//EN" "ditabase.dtd"

**Glossary**

PUBLIC "-//OASIS//DTD DITA Glossary//EN" "glossary.dtd"
PUBLIC "-//OASIS//DTD DITA 1.3 Glossary//EN" "glossary.dtd"
PUBLIC "-//OASIS//DTD DITA 1.x Glossary//EN" "glossary.dtd"

---

**Note:** The glossary.dtd file is provided only for backward compatibility with DITA 1.1. Use the glossentry.dtd file in new documents.

**Glossary entry**

PUBLIC "-//OASIS//DTD DITA Glossary Entry//EN" "glossentry.dtd"
PUBLIC "-//OASIS//DTD DITA 1.3 Glossary Entry//EN" "glossentry.dtd"
PUBLIC "-//OASIS//DTD DITA 1.x Glossary Entry//EN" "glossentry.dtd"

**Learning assessment**

PUBLIC "-//OASIS//DTD DITA 1.3 Learning Assessment//EN" "learningAssessment.dtd"
PUBLIC "-//OASIS//DTD DITA 1.x Learning Assessment//EN" "learningAssessment.dtd"
PUBLIC "-//OASIS//DTD DITA Learning Assessment//EN" "learningAssessment.dtd"

**Learning content**

PUBLIC "-//OASIS//DTD DITA 1.3 Learning Content//EN" "learningContent.dtd"
PUBLIC "-//OASIS//DTD DITA 1.x Learning Content//EN" "learningContent.dtd"
PUBLIC "-//OASIS//DTD DITA Learning Content//EN" "learningContent.dtd"

**Learning overview**
Map and map-based specializations

The identifiers listed below are declared by the OASIS DITA Technical Committee for the modules that are shipped with DITA 1.3. Values that use a version number of 1.3 refer specifically to the modules that are delivered with the DITA 1.3 specification. Values that use a version number of 1.x refer to the modules in the latest approved DITA 1.x specification. Values without a version number refer to the modules in the latest approved DITA specification, regardless of version.

Base map
Comment by Kristen Eberlein, 6 March 2014
Content needed here

Comment by Kristen Eberlein, 6 March 2014
Content needed here

Map

Subject scheme

DITAVAL for conditional processing

The identifiers listed below are declared by the OASIS DITA Technical Committee for the modules that are shipped with DITA 1.3. Values that use a version number of 1.3 refer specifically to the modules that are delivered with the DITA 1.3 specification. Values that use a version number of 1.x refer to the modules in the latest approved DITA 1.x specification. Values without a version number refer to the modules in the latest approved DITA specification, regardless of version.
3.4.2.5 XML Schema catalog identifiers

Each Schema document must be uniquely identified using a Uniform Resource Name (URN). Each Schema document has a version-independent as well as a version-specific URN.

**Format**

An XML resolver should be used to uniquely identify an XML Schema in order to map to a local resource. The DITA specification includes both URN- and URL-based versions of the XML schemas. The URL versions are included as a convenience for use with tools that do not support catalog based resolution and are not normative.

The catalog identifiers use the following patterns.

<table>
<thead>
<tr>
<th>Creator</th>
<th>Pattern</th>
</tr>
</thead>
</table>

where:

- `subcommittee-specialization` indicates the subcommittee and specialization, for example, "machinery" or "learning".
- `filename` is the base name of the document-type shell or module component.
- `version` either is a DITA version number (for example, 1.0, 1.1, 1.2, or 1.3) or is omitted entirely.

Note that "oasis" is the owner identifier; this indicates that the artifacts are owned by OASIS. The keyword "dita" is a convention that indicates that the artifact is DITA-related. The pattern "urn:oasis:names:tc:dita" is a convention that indicates that the artifact is DITA-related and defined by the OASIS DITA Technical Committee.

Catalog identifiers for document-type shells that are not defined by the OASIS DITA Technical Committee or DITA Technical Committee subcommittees **SHOULD** use an owner identifier that unambiguously indicates the owner of the shell, for example, "example.com". In that case, the an appropriate URN would be "urn:example.com:names:dita:xsd:topic.xsd".

The entity resolution catalog that is part of the DITA Schema package lists the full set of document-type shells, module components, and their associated catalog identifiers.

**Topic and topic-based specializations**

The identifiers listed below are declared by the OASIS DITA Technical Committee for the modules that are shipped with DITA 1.3. Values that use a version number of 1.3 refer specifically to the modules that are delivered with the DITA 1.3 specification. Values that use a version number of 1.x refer to the modules in the latest approved DITA 1.x specification. Values without a version number refer to the modules in the latest approved DITA specification, regardless of version.

**Base topic**

- urn:oasis:names:tc:dita:xsd:basetopic.xsd:1.x
- urn:oasis:names:tc:dita:xsd:basetopic.xsd

**Concept**

- urn:oasis:names:tc:dita:xsd:concept.xsd:1.x
- urn:oasis:names:tc:dita:xsd:concept.xsd

**DITAbase**
Glossary

Note: The glossary.xsd file is provided only for backward compatibility with DITA 1.1. Use the glossentry.xsd file in new documents.

Glossary entry

Learning assessment

Learning content

Learning overview

Learning plan

Learning summary

Machinery task

Reference
Map and map-based specializations

The identifiers listed below are declared by the OASIS DITA Technical Committee for the modules that are shipped with DITA 1.3. Values that use a version number of 1.3 refer specifically to the modules that are delivered with the DITA 1.3 specification. Values that use a version number of 1.x refer to the modules in the latest approved DITA 1.x specification. Values without a version number refer to the modules in the latest approved DITA specification, regardless of version.

Base map

urn:oasis:names:tc:dita:xsd:basemap.xsd:1.x
urn:oasis:names:tc:dita:xsd:basemap.xsd

Bookmap

urn:oasis:names:tc:dita:xsd:bookmap.xsd:1.3
urn:oasis:names:tc:dita:xsd:bookmap.xsd:1.x
urn:oasis:names:tc:dita:xsd:bookmap.xsd

Classification


Learning bookmap


Learning map

3.4.3 Specialization

The specialization feature of DITA allows for the creation of new element types and attributes that are explicitly and formally derived from existing types. The resulting specialization allows for the blind interchange of all conforming DITA content and a minimum level of common processing for all DITA content. It also allows specialization-aware processors to add specialization-specific processing to existing base processing.

Specializations are explicitly declared in documents

The specialization feature of DITA defines both a specialization hierarchy declaration syntax used in document instances and a set of document type implementation requirements. The specialization declarations allow processors to determine what set of specializations and associated local constraints a given DITA document uses. The specialization declarations for individual elements and attributes allow processors to determine what the type hierarchies of those elements and attributes are, from which processors can determine the most appropriate (or available) processing to apply.

Specialization enables controlled extension

Specialization allows you to define new kinds of information (new structural types or new domains of information), while reusing as much of existing design and code as possible, and minimizing or eliminating the costs of interchange, migration, and maintenance.

In traditional XML applications, all semantics for a given element instance are bound to the element type, such as `<para>` for a paragraph or `<title>` for a title. The XML specification provides no built-in mechanism for relating two element types to say "element type B is a subtype of element type A". However, in most documentation-focused XML applications there is often a clear hierarchy of types. For example, in a technical manual, there might be generic sections and more specialized sections, e.g., "Troubleshooting" or "Assembly Procedures". The presentation of the generic and specialized sections might be identical, but the more specialized sections might have more restrictive constraints or include additional element types relevant only to
those section types. While these relationships might be understood by authors and system implementors, the XML standard provides no direct way to express the relationship, to say explicitly "A Troubleshooting section is a generic section and must conform to all requirements of generic sections". Having created the element type <section> and implemented presentation processing for it and then having later created the element type <troubleshooting>, there is no obvious mechanism for having all <troubleshooting> elements automatically get the processing associated with <section> elements. To get that behavior someone has to explicitly update all processors involved to apply <section> processing to <troubleshooting>.

The DITA specialization feature provides a standard mechanism for saying explicitly, using normal XML syntax, "A Troubleshooting section is a generic section and must conform to all requirements of generic sections" and, having said that, makes it possible for generic section processing to be applied to troubleshooting sections with no further effort.

**When to use or not use specialization**

Specialization is used when new structural types or new domains are needed. DITA specialization can be used when you want to make changes to your design for the sake of increased consistency or descriptiveness or have specific needs for output that cannot be addressed using the current data model. Specialization is not normally used for simply creating different output types, as DITA documents may be transformed to different outputs.

Do not use specialization to simply eliminate unneeded or unwanted element types from specific content models. The content models for element types defined in vocabulary modules can be configured using separately-defined constraint modules without the need to create new specializations. See Constraints (see page 178).

Use specialization when you are dealing with new semantics (new, meaningful categories of information, either in the form of new structural types or new domains). The new semantics can be encoded as part of a specialization hierarchy, that allows them to be processed by existing specialization-aware transforms or transformed back to more general equivalents ("generalization") for processing by transforms that only understand the unspecialized base types. Use constraints to configure content models and attribute lists without changing semantics.

**Types of specialization hierarchy**

There are two kinds of specialization hierarchy: one for structural types (with topic or map at the root) and one for domains (with elements in topic or map at their root, or the @props or @base attributes). Structural types define topic or map structures, such as concept or task or reference, which often apply across subject areas (for example, a user interface task and a programming task may both consist of a series of steps). Domains define markup for a particular information domain or subject area, such as programming, or hardware. Each type of vocabulary module represents an "is a" hierarchy, in object-oriented terms, with each structural type or domain being a subclass of its parent. For example, a specialization of task is still a task and a specialization of the user interface domain is still part of the user interface domain. A given domain can be used with any map or topic type, as appropriate for the domain. In addition, specific structural types may require the use of specific domains.

**Specialization of attributes**

With structural specializations you can limit the allowed values of attributes defined on the base types of specialized types. You can also define new attributes through domain specializations based off of the @props attribute (for conditional processing) or the @base attribute (for other simple token attributes).

**non-normative:** As a general practice, structural specializations should not limit the values of the built-in selection attributes. Use constraint modules to define specific value lists for built-in selection attributes.

Attribute specialization allows you to define new conditional processing attributes that can be used for filtering and flagging (specializations of @props) or new attributes with no existing equivalent that can be managed and generalized in the same way as conditional processing attributes (specializations of @base).

New attributes need to be specialized from either @props or @base:

- Attributes specialized from @props are recognized as conditional processing attributes
- Attributes specialized from @base have no existing behavior associated with them
• Values in specialized attributes should be preserved during generalization and respecialization as for @props.
• While generalized, the attribute values should still be understandable by both general and specialized behaviors, and be treated as equivalent to their specialized form. For example, conditional filtering should work the same way on specialized attributes and on generalized attributes.

### 3.4.3.1 Vocabulary modules

Vocabulary modules are atomic units of XML vocabulary definition (element types and attributes). A given DITA element type or attribute is declared in exactly one vocabulary module.

Vocabulary modules MUST reflect the coding requirements that are defined in this specification for each recognized grammar mechanism. These requirements ensure that vocabulary modules of a given type follow the same coding patterns for how their components are named and organized.

Vocabulary modules that are intended to be used outside of a narrowly-restricted context SHOULD have one or more associated, globally-unique names by which the modules can be referenced without regard to their local storage location. The globally-unique names can be public identifiers, URNs, or absolute URLs.

There are three types of vocabulary module:

- **structural module**
  - A vocabulary module that defines exactly one top-level map or topic type. Structural modules also can define specializations of, or reuse elements from, domain or other structural modules, creating dependencies on these other modules. Structural modules are either topic vocabulary modules or map vocabulary modules.
  - A topic vocabulary module MUST define exactly one top-level topic type. It MAY define additional topic types that are then allowed to occur as subordinate topics within the top-level topic. However, such subordinate topic types MAY NOT be used as the root elements of conforming DITA documents. For example, a given top-level topic type may require the use of subordinate topic types that would only ever be meaningful in the context of their containing type and thus would never be candidates for standalone authoring or aggregation using maps. In that case, the subordinate topic type can be declared in the module for the top-level topic type that uses it. However, in most cases, potential subordinate topics should be defined in their own vocabulary modules.
  - A map vocabulary module MUST define exactly one element type that specializes map.

- **element domain module**
  - A vocabulary module that defines one or more element types that specialize element types that are used within maps or topics.

- **attribute domain module**
  - A vocabulary module that defines exactly one specialization of either the @base or @props attribute.

A given vocabulary module exists in an exclusive hierarchy relative to its ancestor modules. For example, the <concept> topic type is defined in the concept topic module and is itself derived from the topic topic module (that is, the topic-defining structural module that defines the topic type <topic>). Likewise, the <task> topic type is defined in the task topic module and is derived from the <topic> topic type. Thus the concept and task topic types are children of the <topic> topic type in the module hierarchy rooted at the <topic> topic vocabulary module.

All topic types MUST ultimately be specialized from <topic>. All map types MUST ultimately be specialized from <map>. Domain elements intended for use in topics MUST ultimately be specialized from elements that are defined in the topic module. Domain elements intended for use in maps MUST ultimately be specialized from elements defined by or used in the map module (maps share some element types with topics but no map-specific elements may be used within topics). Domain attributes MUST ultimately be specialized from either the @base or @props attribute.
Each vocabulary module has an associated short name, which is used to identify the module in @class and @domains attribute values. While module names need not be globally unique, module names MUST be unique within the scope of a given specialization hierarchy. The short name MUST be a valid XML name token.

For structural types, the module name MUST be the same as the root element. For example, "task" is the name of the structural vocabulary module whose root element is <task>. For domains, the name is assigned by the developer of the vocabulary module. By convention, domain names end with "-d" and are kept short; for example, "ui-d" for the user interface domain and "pr-d" for the programming domain.

When integrated into document types shells, vocabulary modules MAY be further restricted through the use of constraint modules. See Constraints (see page 178).

### 3.4.3.2 Requirements for specialized element types and attributes

When you specialize one element from another, or a new attribute from @props or @base, the new element or attribute MUST obey certain rules in order to be a conforming specialization.

A specialized element:

- MUST have a properly-formed @class attribute specifying inheritance from its parent.
- MUST NOT have a more inclusive content model than its parent has.
- MUST NOT have attributes that its parent lacks.
- MUST NOT have values or value ranges of these attributes that are more extensive than those in the parent.

An attribute specialized from the @props or @base attribute:

- MUST follow the rules for attribute domain specialization.
- MUST NOT have values or value ranges that are more extensive than those of the parent.
- MUST conform to the rules for conditional processing values, that is, alphanumeric space-delimited values. In generalized form, the values MUST conform to the rules for attribute generalization.
- MUST be declared as a global attribute. Attribute specializations cannot be limited to specific element types.

DITA elements are never in a namespace. Only the @DITAArchVersion attribute is in a DITA-defined namespace. All other attributes, except for those defined by the XML standard, are in no namespace.

This limitation is imposed by the details of the @class attribute syntax, which makes it impractical to have namespace-qualified names for either vocabulary modules or individual element types or attributes. Elements included as descendants of the DITA <foreign> element type may be in any namespace.

**Note:** For this reason, designers of domain modules that are intended for wide use should take care to define element type and attribute names that are unlikely to conflict with names used in other domains, for example, by using a domain-specific prefix on all names.

### 3.4.3.3 @class attribute

Each DITA element declares its specialization hierarchy as the value of the @class attribute. The @class attribute usually provides a mapping from the current name of the element to its more general equivalents, but it also can provide a mapping from the current name to more specialized equivalents. All specialization-aware processing can be defined in terms of @class attribute values without reference to a given element tagname.

**Specialization hierarchy declaration requirements**

Values for the @class attribute MUST conform to the following syntax requirements:

- An initial "-" or "+" character followed by one or more spaces, "-" for element types defined in structural vocabulary modules, "+" for element types defined in domain modules.
- A sequence of one or more module/type pair tokens of the form "modulename/typename", with each pair of tokens separated by one or more spaces, where modulename is the short name of the vocabulary module
and typename is the element type name. Tokens are ordered left to right from most general to most specialized.

- At least one trailing space character (" "). The trailing space ensures that string matches on module/name pairs can always include a leading and trailing space in order to reliably match full tokens.

When the @class attribute is declared in an XML grammar, it must be declared with a default value. In order to support generalization round-tripping (generalizing specialized content into a generic form and then returning it to the specialized form) the default value must not be fixed. This allows a generalization process to overwrite the default values defined by a general document type with specialized values taken from the document being generalized.

When a vocabulary module declares new element types, it must provide a @class attribute for each element type that it declares. The @class attribute must include a mapping for every structural type or domain in the specialized type's ancestry, even those in which no element renaming occurred. The mapping must start with the value for the base type (for example topic or map), and finish with the current element type.

A vocabulary module must not change the @class attribute for elements that it does not specialize, but simply reuses by reference from more generic levels. For example, if task, bctask, and guitask use the <p> element without specializing it, they must not declare mappings for it.

Authors should not modify the @class attribute.

Examples

The @class attribute for the task topic type's <step> element is:

```xml
<!ATTLIST step         class  CDATA "- topic/li task/step ">
```

This tells us that the <step> element is equivalent to the <li> element in a generic topic. It also tells us that <step> is equivalent to a <step> in a task topic, which we already knew, but it's worth noting this in the attribute because it enables round-trip migration between upper level and lower level types without loss of information.

While a given element's tagname is normally the same as the typename of the last token in the @class value, this is not required. Processors that perform generalization may transform elements from specialized types to less-specialized types, leaving the values of the @class attribute unchanged (thus preserving knowledge of the original most-specialized form). For example, if a user runs a generalizing transformation that maps all elements to their first @class value, but preserves their content and attribute values, then the user can follow it up with a "specialize" transformation that maps all elements to their last @class value (preserving content and attribute values), and provide a full round trip for all content between the two document types, using nothing but two generic transformations and the information in the @class attribute.

The @class attribute tells a processor what general classes of elements the current element belongs to. DITA scopes elements by module type (for example topic type, domain type, or map type) instead of document type, which lets document type developers combine multiple topic types in a single document without complicating transformation logic.

The sequence of values in the @class attribute is important because it tells processors which value is the most general and which is most specific. This is especially important for "specializing" transformations, where you can apply a general rule that says: if the element doesn't have a mapping to the target topic type, simply use the last value of the @class attribute (and assume that the specialized topic type is reusing some general element declarations, which only have mappings for the level at which they were declared).

```xml
<appstep class="- topic/li task/step bctask/appstep ">
  <cmd class="- topic/ph task/cmd ">A specialized step</cmd>
</appstep>
```

Figure 30: Example of structural type element with @class attribute

```xml
<wintitle class="+ topic/keyword ui-d/wintitle ">A specialized keyword</wintitle>
```

Figure 31: Example of domain element with @class attribute
While this example is trivial, more complicated hierarchies (say, five levels deep, with renaming occurring at levels two and four only) make explicit intermediate values essential.

The specialization hierarchy for a given element type must reflect any intermediate modules between the base type and the specialization type, as shown in this example:

```xml
<widowname class="- topic/keyword task/keyword guitask/windowname ">
...</windowname>
```

**Figure 32: Example of `@class` attribute with intermediate value**

The intermediate values are necessary so that generalizing and specializing transformations can map values simply and accurately. For example, if `task/keyword` was missing as a value, and a user decided to generalize this `guitask` up to a `task` topic, then the transformation would have to guess whether to map to `keyword` (appropriate if task is more general than guitask, which it is) or leave it as `windowname` (appropriate if task were more specialized, which it isn't). By always providing mappings for more general values, processors can then apply the simple rule that missing mappings must by default be to more specialized values than the one we are generalizing to, which means the last value in the list is appropriate. For example, when generalizing `<guitask>` to `<task>`, if a `<p>` element has no target value for `<task>`, we can safely assume that `<p>` does not specialize from `<task>` and should not be generalized.

### 3.4.3.4 `@domains` attribute

Structural types *MUST* declare the domain vocabulary modules and constraint modules that they use, as well as any dependencies that they have on elements from other (non-ancestor) structural modules. This is done using a `@domains` attribute value that is a sequence of parenthesized module ancestry specifications. The `@domains` attribute is declared on the root element for each topic or map type. Structural modules *SHOULD* declare their structural ancestry, and structural modules *MUST* do so when they have dependencies on structural modules other than their ancestors.

Each structural, element, and attribute domain defines its module ancestry as a parenthesized sequence of space-separated module names from root module to provided module.

For element domains, the group syntax is:

```
'(', modulename, (' ', modulename)+, ')'
```

Comment by Kristen Eberlein, 8 April 2014

Jarno Elovirta made the following comment in review #1; neither Robert nor I knew what to make of it:

This needs to contain reuse. If domain comes before structural reuse, then:

```
'(', modulename, (' ', modulename)+, ('+', domain-or-attribute-reuse-modulename)*('++', structural-reuse-modulename)*,')'
```

If unordered, then:

```
'(', modulename, (' ', modulename)+, (('+', domain-or-attribute-reuse-modulename) | ('++', structural-reuse-modulename))*,')'
```

For attribute domains, the group syntax is:

```
a('', attname, (' ', attname)+, ')
```

The module ancestry specifications are added to the effective value of the `@domains` attribute to form a set of specifications, one for each domain that is used by the topic or map type.

The `@domains` values for the different module types are as follows:

**structural modules**
The structural type ancestry. For example: `(topic concept glossentry)`.

When a structural module has elements that directly reference or specialize elements from another structural, domain, or attribute module, the structural module's `@domains` specification **MUST** include the names of the other modules as dependencies following the name of the structural domain.

Dependencies on domain or attribute specialization modules are appended to the reusing module's value, or to previous dependencies, separated by "+". Dependencies on structural specialization modules are appended to the reusing module's value, or to previous dependencies, separated by "++".

A dependency exists on an element domain module, or a structural module, whenever a specialization uses one or more elements from such a module. A dependency exists on an attribute domain module whenever a specialization binds one or more of its element attribute definitions to definitions in the attribute domain module.

For example, a `cppAPIRef` specialization that has a dependency on the `cpp-d` element domain and `compilerTypeAtt-d` attribute domain would need to include a domain value that showed the dependencies as follows: `(topic reference cppApiRef+cpp-d+compilerTypeAtt-d)`. A `codeChecklist` specialization that has a dependency on the `pr-d` domain and the task structural specialization would show dependencies as follows: `(topic reference codeChecklist+pr-d++task)`.

**structural module dependencies**

The structural type ancestry of the structural type whose elements are being reused. For example, if a troubleshooting module reuses elements from the task structural module, then it must include both a dependency on task in the troubleshooting ancestry value, and the ancestry of task in a separate value:

`(topic troubleshooting++task) (topic task)`

For more information on the syntax for expressing structural module dependencies, see .

---

**Comment by Kristen Eberlein, 7 March 2014**

The above cross reference was broken in the stage 3 proposal. The "sharing-elements-across-modules" topic does not contain information on syntax for expressing structural module dependencies.

---

**constraint modules**

The structural type ancestry followed by the name of the constraint domain. For example: `(topic task strictTaskbody-c)`.

**element domains**

The structural type ancestry and, if applicable, the domain module ancestry from which the domain is specialized. For example: `(topic hi-d) (topic pr-d cpp-d)`.

**attribute domains**

The attribute specialization hierarchy. For example: `a(props mySelectAttribute)`.

The `@domains` attribute allows processors to determine whether two elements use compatible domains. For example, when pasting content from one topic into another topic within an editor, the editor can use the `@domains` attribute to determine if the topic domains of the paste target are compatible with the topic domains of the paste source, and therefore whether the pasted content needs to be generalized before it can be pasted. Likewise, processors can use the value of the `@domains` attribute to determine if they have whatever may be necessary to support a particular domain.

Another example is when an element references an element that is a more specialized version of the element, for example, a `<li>` element in a concept topic references a `<step>` element in a task topic. During processing, the `<step>` element will be generalized back to a `<li>` element.

The effective value of the `@domains` attribute is constructed using integration mechanisms that are specific to each XMLgrammar. Each domain and constraint module **MUST** provide a `@domains`-attribute value fragment.
that can be used by DITA document types to construct the effective value of the @domains attribute. Each structural vocabulary module SHOULD provide a @domains-attribute value fragment. See Configuration (Document type shells) (see page 118).

Example: Task with multiple domains

In this example, the task topic integrates the domains for describing user interfaces (ui-d), software (sw-d), and programming (pr-d).

```xml
<task id="mytask" class="- topic/topic task/task"
     domains="(topic ui-d) (topic sw-d) (topic pr-d)">
<!-- ... -->
</task>
```

If the document also used a specialization of the programming domain to describe C++ programming, the new domain would need a separate entry in the @domains attribute, for example:

```xml
<task id="mytask" class="- topic/topic task/task"
     domains="(topic ui-d) (topic sw-d) (topic pr-d) (topic pr-d cpp-d)">
     ...
</task>
```

3.4.3.5 Domain attribute syntax for cross-specialization dependencies

When a structural specialization reuses elements from another structural or domain specialization, it has a dependency on the source specialization module that must be declared in the @domains attribute in order for conref and generalization processing to function correctly.

The @domains attribute must include a parenthetical expression that lists the ancestry for the current specialization. The final value, for the current specialization, must include the modules that it depends on, separated by either a "+" for a dependency on a domain specialization or a "++" for a dependency on another structural specialization.

In addition, it must include a separate parenthetical expression for each reused domain or structural specialization that lists the ancestry of the reused specialization.

Example: Dependency on a domain specialization

For example, a codeConcept specialization could specialize from <concept> and include elements from the programming domain. Its ancestry, and the ancestry of the domain specialization that it reuses from, would be declared as:

```xml
domains="...(topic concept codeConcept+pr-d) (topic pr-d)"
```

Example: Dependency on a structural specialization

For example, a checklist specialization could specialize from <reference> and include elements from <task>. Its ancestry, and the ancestry of the structural specialization that it reuses from, would be declared as:

```xml
domains="...(topic reference checklist++task) (topic task)"
```
Example: Dependency on multiple specializations

For example, a codeChecklist specialization could specialize from `<reference>` and include elements from both `<task>` and the programming domain. Its ancestry, and the ancestry of the domain and structural specializations that it reuses from, would be declared as:

```
domains="...(topic reference checklist++task+pr-d) (topic task) (topic pr-d)"
```

### 3.4.3.6 Generalization

Specialized content can be generalized to any ancestor type. The generalization process can preserve information about the former level of specialization to allow round-tripping between specialized and unspecialized forms of the same content.

Generalization serves several purposes:

- Migration of content (for example, when retiring an unsuccessful specialization)
- Temporary round-tripping (for example, when moving content through a process that is not specialization aware and has only been enabled for instances of the base structural type)
- Reuse of specialized content in an environment that does not support one or more of its specializations (which may be thought of as a special case of round-tripping).

When generalizing for migration, the `@class` attribute and `@domains` attribute should be absent from the generalized instance document, so that the default values in the DITA document-type shell will be used. When generalizing for round-tripping, the `@class` attribute and `@domains` attribute SHOULD retain the original specialized values in the generalized instance document.

All DITA documents contain a mix of markup from at least one structural type and zero or more domains. When generalizing the document, the generalizer can leave a structural type or domain as-is, or can generalize that type or domain to any of its ancestors.

Comment by Kristen Eberlein, 16 February 2014

Can we replace the phrase "the generalizer" with "the processor"?

The generalizer can supply the source and target modules for each generalization, for example, "generalize from reference to topic". The generalizer can specify multiple target modules, for example, "generalize from reference to topic and from ui-d to topic". When the source and target modules are not supplied, generalization is assumed to be from all structural types to the base (topic or map), and no generalization is performed for domains.

The generalizer also can supply the target document-type shell. When the target document type is not supplied, the generalized document will not contain a reference to a DITA document-type shell. With the exception of topic nesting restrictions, it is possible to generate a document type shell based on the `@class` and `@domains` attributes in the specialized documents. If the `@domains` attribute includes all structural, domain, and constraint modules that are used, the `@domains` attribute alone is sufficient to enable the generation of a document type shell.

A generalization process SHOULD be able to handle cases where it is given:

- Just source modules for generalization (in which case the designated source types are generalized to topic or map)
- Just target modules for generalization (in which case all descendants of the target are generalized to that target)
- Both (in which case only the specified descendants of the target are generalized to that target)

For each structural type instance, the generalization process checks whether the structural type instance is a candidate for generalization, or whether it has domains that are candidates for generalization. It is important to
be selective about which structural type instances to process; if the process simply generalizes every element based on its @class attribute values, an instruction to generalize "reference" to "topic" could leave an APIReference topic with an invalid content model, since any elements it reuses from "reference" would have been renamed to topic-level equivalents.

For specific concerns when generalizing structural types with dependencies on non-ancestor modules, see Generalization with cross-specialization dependencies (see page 151).

For specific concerns when generalizing attributes, see Attribute generalization (see page 152).

The @class attribute for the root element of the structural type is checked before generalizing structural types:

<table>
<thead>
<tr>
<th>Source module unspecified</th>
<th>Source module specified</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Target module unspecified</strong></td>
<td>Generalize this structural type to its base ancestor</td>
</tr>
<tr>
<td><strong>Target module specified</strong></td>
<td>Check whether the @class attribute contains the target module. If it does contain the target, rename the element to the value associated with the target module. Otherwise, ignore the element.</td>
</tr>
</tbody>
</table>

The @domains attribute for the root element of the structural type is checked before generalizing domains:

<table>
<thead>
<tr>
<th>Source module unspecified</th>
<th>Source module specified</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Target module unspecified</strong></td>
<td>Do not generalize domain specializations in this structural type.</td>
</tr>
<tr>
<td><strong>Target module specified</strong></td>
<td>Check whether the @domains attribute contains the target module. If it does, generalize to the target module. Otherwise, skip the structural type instance unless it is itself a candidate for generalization.</td>
</tr>
</tbody>
</table>

For each element in a candidate structural type instance:

<table>
<thead>
<tr>
<th>Source module unspecified</th>
<th>Source module specified</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Target module unspecified</strong></td>
<td>If the @class attribute starts with &quot;.&quot; (part of a structural type), rename the element to its base ancestor equivalent. Otherwise ignore it.</td>
</tr>
<tr>
<td><strong>Target module specified</strong></td>
<td>Check whether the @class attribute contains the target module; rename the element to the value associated</td>
</tr>
</tbody>
</table>
When renaming elements during round-trip generalization, the generalization process SHOULD preserve the values of all attributes. When renaming elements during one-way or migration generalization, the process SHOULD preserve the values of all attributes except the @class and @domains attribute, both of which should be supplied by the target document type.

### 3.4.3.7 Generalization with cross-specialization dependencies

Dependencies across specializations limit generalization targets to those that either preserve the dependency or eliminate them. Some generalization targets will not be valid and should be detected before generalization occurs.

When a structural specialization has a dependency on a domain specialization, then the domain cannot be generalized without also generalizing the reusing structural specialization.

For example, if programming domain elements in codeConcept were generalized to their topic equivalents without also generalizing codeConcept elements, then the places where the content model of a codeConcept element includes programming domain elements would become invalid. However, codeConcept could be generalized to concept or topic, without generalizing programming domain elements, as long as the target document type includes the programming domain.

When a structural specialization has a dependency on another structural specialization, then both must be generalized together to a common ancestor.

For example, if the task elements in checklist were generalized without also generalizing checklist elements, then the checklist content models that referenced task elements would be broken. And if the checklist elements were generalized to topic without also generalizing the task elements, then the task elements would be out of place, since they cannot be validly present in topic. However, checklist and task can be generalized together to any ancestor they have in common: in this case topic.

Generalizing processes SHOULD detect invalid generalization target combinations and report them as errors.

### 3.4.3.8 Sharing elements across specializations

The process of specialization enables easy reuse of elements from ancestor specializations; however, it is also possible to reuse elements from non-ancestor specializations, as long as the dependency is properly declared to prevent invalid generalization or conref processing.

A structural specialization can incorporate elements from unrelated domains or other structural specializations by referencing them in the content model of a specialized element. For example, a specialized reference topic could declare a specialized list (<apilist>) in which each <apilistitem> contains an <apiname> element that is borrowed from the programming domain. Or a specialized concept topic could declare a specialized <process> section that contains the <steps> elements borrowed from task.

The elements included in this manner must be specialized from ancestor content that is valid in the new context. For example, a specialized element can include the <steps> elements in its content model if it could validly include <ol>, because <steps> is specialized from <ol>. If the reusing and reused specializations share common ancestry, the reused elements must be valid in the reusing context at every level they share in common.
For example, suppose there are two specializations of `<task>`: `<mytask>` and `<yourtask>`. The specializer of `<mytask>` wants to borrow `<yoursteps>` from `<yourtask>` instead of creating their own equivalent specialization. The `<yoursteps>` element is specialized from `<steps>` in `<task>`. The specializer of `<mytask>` can only include `<yoursteps>` where `<steps>` would be allowed; they cannot incorporate it anywhere `<ol>` is allowed, even though `<ol>` is in its ancestry, because the two specializations share `<task>` in their ancestry as well, and we need to maintain the validity of `<task>` as a generalization target. If we allowed `<mytask>` to transplant specializations of `<steps>` anywhere they wanted in the `<mytask>` model, then `<mytask>` would cease to be a valid specialization of `<task>`.

For example, suppose that `<yourtask>` is a valid specialization of `<task>` and that the `<yoursteps>` element in `<yourtask>` is a specialization of the `<steps>` element from `<task>`. The `<yoursteps>` ancestry is: `topic/ol task/steps yourtask/yoursteps`. Consequently, `<yoursteps>` must only appear in a context in `<yourtask>` that can be generalized back to `<steps>` in `<task>`. Now, suppose that another specializer wants to create a second specialization of `<task>`, called `<mytask>`. The specializer borrows the `<yoursteps>` element from the `<yourtask>` specialization for use in `<mytask>`, because `<yoursteps>` is exactly what is needed. As was the case in `<yourtask>`, the placement of `<yoursteps>` in `<mytask>` must also appear in a context that can be generalized back to `<steps>` in `<task>`. This is because the ancestry for `<yoursteps>` is still: `topic/ol task/steps yourtask/yoursteps`, even though `<yoursteps>` is being used in the `<mytask>` specialization.

Although a well-designed structural specialization hierarchy with controlled use of domains is still the primary means of sharing and reusing elements in DITA, the ability to also share elements declared elsewhere in the hierarchy allows for situations where relevant markup comes from multiple sources and would otherwise be developed redundantly.

### 3.4.3.9 Attribute generalization

DITA provides a syntax to generalize attributes that have been specialized from the `@props` or `@base` attribute. Specialization-aware processors _SHOULD_ process both the specialized and generalized forms of an attribute as equivalent in their values.

Both Eliot and I think that the above normative statement should be _MUST_, as it is a fundamental part of interoperability. However, the 1.2 spec shipped with a _should_.

When a specialized attribute is generalized to an ancestor attribute, the value of the ancestor attribute consists of the name of the specialized attribute followed by its specialized value in parentheses. For example, if `@jobrole` is an attribute specialized from `@person`, which in turn is specialized from `@props`:

- `jobrole="programmer"` can be generalized to `person="jobrole(programmer)"` or to `props="jobrole(programmer)"
- `props="jobrole(programmer)"` can be respecialized to `person="jobrole(programmer)"` or to `jobrole="programmer"

In this example, processors performing generalization and respecialization can use the `@domains` attribute to determine the ancestry of the specialized `@jobrole` attribute, and therefore the validity of the specialized `@person` attribute as an intermediate target for generalization.

If more than one attribute is generalized, the value of each is separately represented in this way in the value of the ancestor attribute.

Generalized attributes are typically not expected to be authored or edited directly. They are used by processors to preserve the values of the specialized attributes during the time or in the circumstances in which the document is in a generalized form.
A single element **MUST NOT** contain both generalized and specialized values for the same attribute. For example, the following `<p>` element provides two values for the `@jobrole` attribute, one in a generalized syntax and the other in a specialized syntax:

```xml
<p person="jobrole(programmer)" jobrole="admin">
  <!-- ... -->
</p>
```

This is an error condition, since it means the document has been only partially generalized, or that the document has been generalized and then edited using a specialized document type.

### 3.4.3.10 Specializing foreign or unknown content

Specializing the `<foreign>` or `<unknown>` element is an open extension to DITA for the purpose of incorporating standard vocabularies for non-textual content, such as MathML and SVG, as markup within DITA documents. These elements should not be used to include textual content or metadata in DITA documents, except where such content acts as an example or display, rather than as the primary content of a topic.

**Note:** Both MathML and SVG are built-in domains starting with DITA 1.3 and serve as working examples of `<foreign>` specializations.

#### Incorporating foreign or unknown content

There are three methods of incorporating foreign content into DITA.

- A domain specialization of the `<foreign>` or `<unknown>` element. This is the usual implementation.
- A structural specialization using the `<foreign>` or `<unknown>` element. This affords more control over the content.
- Do nothing: simply embed the foreign content within `<foreign>` or `<unknown>`.

#### Foreign or unknown content and the architectural `@class` attribute

Foreign content that is incorporated in DITA by one of these methods is not specialized. Specialization depends upon the architectural `@class` attribute found in every DITA element. If the foreign content has interoperability or vocabulary naming issues such as those that are addressed by specialization in DITA, they must be addressed by means that are appropriate to the foreign content.

#### Example of specializing foreign or unknown content using DTDs

The sample below shows how to create a domain declaration for SVG markup and integrate it into a document type shell.

From the `svgDomain.ent` file:

```xml
<!-- ---------------------------------------------------------- -->
<!-- SVG DOMAIN ENTITIES -->
<!-- ---------------------------------------------------------- -->

<!--[.--- SVG elements must be prefixed, otherwise they conflict with existing DITA elements (e.g., <desc> and <title>).]-->

<!ENTITY % NS.prefixed "INCLUDE" >
<!ENTITY % SVG.prefix "svg" >

<!ENTITY % svg-d-foreign 
"svg-container"
>

<!ENTITY % svg-d-att 
"(topic svg-d)"
>

DITA Version 1.3 Specification
Copyright © OASIS Open 2005, 2014. All Rights Reserved.
Note that the SVG-specific `%SVG.prefix;` parameter entity is declared. This establishes the namespace prefix to be used for the SVG content. It could be overridden in a document type shell by declaring the parameter entity before the reference to the `svgDomain.ent` file.

From the `svgDomain.mod` file:

```xml
<!ENTITY % svg-container        "svg-container" >
<!ENTITY % svgref               "svgref" >
<!ENTITY % svg11.dtd
SYSTEM "svg11/svg11.dtd">
<!ENTITY % svg-container.content (%data; | %data-about; | %SVG.pfx;svg)* >
<!ENTITY % svg-container.attributes "id-atts; %localization-atts; base CDATA #IMPLIED %base-attribute-extensions; outputclass CDATA #IMPLIED >
<!ENTITY % svgref.content "EMPTY" >
<!ENTITY % svgref.attributes "href CDATA #IMPLIED keyref CDATA #IMPLIED type CDATA #IMPLIED format CDATA #IMPLIED scope (external | local | peer | -dita-use-conref-target) %uni-atts; outputclass CDATA #IMPLIED" >
```
Example of SVG within a `<p>` element

```xml
<p>This is an ellipse:
<svg-container>
<svg:svg width="100%" height="100%" version="1.1"
xmlns="http://www.w3.org/2000/svg">
<ellipse cx="300" cy="150" rx="200" ry="80"
style="fill:rgb(200,100,50);
stroke:rgb(0,0,100);stroke-width:2"/>
</svg:svg>
</svg-container>.</p>
```

Example of specializing foreign content using RELAX NG

The code sample below illustrates how to create a domain declaration for the RELAX NG `<grammar>` element and integrate it into a document type shell. Domain modules that include foreign grammars should use the `<externalRef>` element to refer to the foreign grammar when the grammar specifies a `<start>` element. When using the `<externalRef>` element, it is possible to safely include foreign vocabularies that are not in a namespace, because the externally-referenced grammar patterns are not merged with those of the referencing grammar.
The following domain specialization specializes `<foreign>` in order to contain RELAX NG `<grammar>` elements, for example, to document document types by rendering the grammar markup as diagrams.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
xmlns="http://relaxng.org/ns/structure/1.0">
  <moduleDesc xmlns="http://dita.oasis-open.org/architecture/2005/">
    <moduleTitle>RelaxNG Foreign Domain</moduleTitle>
    <headerComment>
      Define the Relax NG XML syntax as a DITA foreign vocabulary
      ---------------------------------------------------------------
    </headerComment>
    <moduleMetadata>
      <moduleType>elementdomain</moduleType>
      <moduleShortName>rnggrammar-d</moduleShortName>
      <modulePublicIds>
        <dtdMod>urn:oasis.org:elements:dita:dtd:rnggrammarDomain.mod</dtdMod>
        <dtdEnt>urn:oasis.org:entities:dita:dtd:rnggrammarDomain.ent</dtdEnt>
        <xsdMod>urn:oasis.org:names:dita:xsd:rnggrammarDomain.xsd</xsdMod>
        <rncMod>urn:oasis.org:names:dita:rnc:rnggrammarDomain.rnc</rncMod>
        <rngMod>urn:oasis.org:names:dita:rng:rnggrammarDomain.rng</rngMod>
      </modulePublicIds>
      <domainsContribution>(topic rnggrammar-d)</domainsContribution>
    </moduleMetadata>
  </moduleDesc>
  <div>
    <a:documentation>DOMAIN EXTENSION PATTERNS</a:documentation>
    <define name="rnggrammar-d-foreign">
      <choice>
        <ref name="rng_grammar"/>
      </choice>
    </define>
    <define name="foreign" combine="choice">
      <ref name="rnggrammar-d-foreign"/>
    </define>
  </div>
  <div>
    <a:documentation>ELEMENT TYPE NAME PATTERNS</a:documentation>
    <define name="rng_grammar">
      <ref name="rng_grammar.element"/>
    </define>
  </div>
  <div>
    <a:documentation>ELEMENT TYPE DECLARATIONS</a:documentation>
    <define name="rng_grammar.content">
      <zeroOrMore>
        <choice>
          <externalRef href="rngGrammarProxy.rng"
            ditaarch:dtPublicId="-//OASIS//DTD DITA RNG Grammar//EN"
            ditaarch:dtSystemId="rngGrammar.dtd"
            ditaarch:xsdURI="rngGrammar.xsd"
            dita:xsdTargetNamespace="http://relaxng.org/ns/structure/1.0"/>
          <ref name="data"/>
          <ref name="data-about"/>
        </choice>
      </zeroOrMore>
    </define>
    <define name="rng_grammar.attributes">
      <ref name="univ-atts"/>
      <optional>
        <attribute name="outputclass"/>
      </optional>
    </define>
    <define name="rng_grammar.element">
      <element name="rng_grammar" ditaarch:longName="RNG grammar">
        <ref name="rng_grammar.attlist"/>
        <ref name="rng_grammar.content"/>
      </element>
    </define>
  </div>
</grammar>
```
Because the RELAX NG grammar defines patterns that match any element with any content (patterns named "other" and "any" in the RELAX NG grammar), the domain requires overriding these patterns to exclude the element names that define an attribute with an ID type in order to satisfy @id attribute declaration consistency requirements that are imposed by the RELAX NG DTD compatibility facility, elements that are referred to explicitly in these patterns with the reference to the "idElements" pattern. This is done through an intermediate file, rngGrammarProxy.rng, which then references the main grammar document and overrides the two patterns:

Comment by Kristen Eberlein, 8 April 2014

From review #1:

Eberlein: "We need to edit and unpack [the first] sentence, Eliot. It is too dense and difficult to read and parse."

Kimber: "Need to say a bit more about "proxy" grammars for foreign vocabs anyway. In particular, that the proxy may need to be locally modified in order to reflect all the shell topic or map types into which the foreign vocabulary is integrated. So we need to do a bit more writing around this whole subject."

<?xml version="1.0" encoding="UTF-8"?>
<grammar datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"
  ns="http://relaxng.org/ns/structure/1.0"
  xmlns="http://relaxng.org/ns/structure/1.0">
  <include href="rngGrammar.rng">
    <define name="other">
      <zeroOrMore>
        <choice>
          <parentRef name="idElements"/>
          <element>
            <anyName/>
            <except>
              <name ns=""></name>
            </except>
          </element>
        </choice>
      </zeroOrMore>
    </define>
    <define name="any">
      <choice>
        <parentRef name="idElements"/>
        <element>
          <anyName/>
          <except>
            <name ns=""></name>
          </except>
        </element>
      </choice>
    </define>
  </include>
</grammar>
Example of specializing foreign content using XML Schemas

The sample below describes how to create a domain declaration of the mathML element, but does not show how to integrate that declaration in a DITA document-type shell. For more specific information on creating document-type shells, see XML Schema: Coding requirements for DITA specialization modules (see page 164)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:m="http://www.w3.org/1998/Math/MathML"
   elementFormDefault="qualified">
  <xs:import schemaLocation="mathml3/mathml3.xsd"
             namespace="http://www.w3.org/1998/Math/MathML"/>
  <xs:group name="mathml-d-foreign">
    <xs:sequence>
      <xs:choice>
        <xs:element ref="mathml_container"/>
      </xs:choice>
    </xs:sequence>
  </xs:group>
  <xs:group name="mathml_container.content">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="m:math"/>
      <xs:group ref="data.elements.incl" minOccurs="0"/>
    </xs:choice>
  </xs:group>
  <xs:attributeGroup name="mathml_container.attributes">
    <xs:attribute name="outputclass" type="xs:string"/>
    <xs:attributeGroup ref="global-atts"/>
    <xs:attributeGroup ref="univ-atts"/>
  </xs:attributeGroup>
  <xs:complexType name="mathml_container.class" mixed="false">
    <xs:sequence>
      <xs:group ref="mathml_container.content"/>
    </xs:sequence>
    <xs:attributeGroup ref="mathml_container.attributes"/>
  </xs:complexType>
  <xs:element name="mathml_container">
    <xs:annotation>
      <xs:documentation>
        The mathml_container (&lt;keyword>mathml_container</keyword>&gt;) element contains zero or more MathML equations, along with optional &lt;keyword&gt;data</keyword>&gt; or &lt;keyword&gt;data-about</keyword>&gt; elements, which act as metadata for the equations.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType mixed="false">
      <xs:complexContent>
        <xs:extension base="mathml_container.class">
          <xs:attribute ref="class" default=""/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
</xs:schema>
```
Example of MathML within an <equation-block> element

<p>
<equation-block>
<!-- sum 4 + x -->
<mathML>
<m:math display="block">
  <m:mrow>
    <m:mo>sum</m:mo>
    <m:mn>4</m:mn>
    <m:mo>+</m:mo>
    <m:mi>x</m:mi>
  </m:mrow>
</m:math>
</mathML>
</equation-block>.
</p>

3.4.3.11 Specialization module coding requirements

The base DITA element and attribute types may be extended through the creation of new vocabulary modules that define specializations of more-general types.

3.4.3.11.1 DTD syntax specialization module coding requirements

To be extensible and backward compatible, DITA requires that a DTD implementation of structural and domain specialization modules conform to well-defined implementation (coding) requirements.

These coding requirements implement the specialization architecture with the capabilities and within the limitations of the DTD grammar. They are the coding requirements for structural specializations, element domain specializations, and attribute domain specializations.

3.4.3.11.1.1 General element type declaration coding requirements

Structural and element domain vocabulary modules MUST reflect the same coding requirements for element type declarations.

Module names

Each vocabulary module has a short name that is used to construct file names, entity names, and other names used in associated declarations. Modules MAY also have abbreviated names that further shorten the short name, for example "sw" for the "software" domain, where "software" is the short name and "sw" is the abbreviated name.

For structural modules, the module name MUST be the element type name of the top-level topic or map type defined by the module, such as "concept", "bookmap".

For element domain modules, the module name MUST be a name that reflects the subject domain to which the domain applies, such as "highlight", "software". Domain module names should be sufficiently unique that they are unlikely to conflict with any other domains.

Module files

A structural or element domain vocabulary module MUST have two files:

- A module entity declaration file, which declares the entities used to integrate the module into a shell DTD.
For structural modules, the file name is the module name plus the `ent` extension, e.g. `concept.ent`. For domain modules, the file name is the domain name plus `Domain` plus the `ent` extension, e.g. `highlightDomain.ent`, `newAttDomain.ent`.

- A definition module, which contains the element type and/or attribute list declarations for the module.
  - For structural modules, the file name is the module name plus the `mod` extension, e.g., `concept.mod`.
  - For domain modules, the file name is the domain name plus "Domain" and the `mod` extension, e.g., `highlightDomain.mod`, `newAttDomain.mod`.

**Domain declaration entity**

The domain declaration entity *MUST* conform to the following implementation pattern:

The declaration file *MUST* define an entity that associates the domain with a module. The name of the entity is the structure type name or domain abbreviation plus the `-att` suffix, e.g. "concept-att", "hi-d-att".

The value of the entity *MUST* list the dependencies of the domain module in order of dependency from left to right within enclosing parentheses, starting with the topic module. Domain abbreviations are used in the list, and the defining domain is the last item in the list. The following example declares the dependency of the highlight domain on the base topic module.

```
<!ENTITY hi-d-att "(topic hi-d)">
```

The domain declaration entity is used to construct the effective value of the `@domains` attribute for a map or topic type as configured in a shell DTD.

**Element definitions**

A structural or domain vocabulary module *MUST* contain a declaration for each specialized element type named by the module. While the XML standard allows content models to refer to undeclared element types, all element types named in content models or attribute list declarations within a vocabulary module *MUST* have an `ELEMENT` declaration, in one of:

- The vocabulary module
- A base module of which the vocabulary module is a direct or indirect specialization
- A required domain module (if the vocabulary module is a structural module).

The specialized elements *MUST* follow the rules of the architecture in defining content models and attributes.

For each element type declared in the vocabulary module there must be an element name parameter entity whose default value is the name of the element, e.g.:

```
<!ENTITY % conbody "conbody">
```

The element name entity provides a layer of abstraction that facilitates redefinition. A document type shell can redefine an element entity to add domain-specialized elements or replace a base element type with one or more specializations of that type. Because declarations use the entity rather than the element type name to include the element in a content model, the redefinition given in a shell is propagated to every context in which the base element occurs.

The element name parameter entities *MUST* be grouped together at the top of the vocabulary module before any other declarations to ensure they are declared before any use in content models declared in the same module. The declarations can occur in any order. By convention, they are usually ordered alphabetically or grouped logically.

For each element type, the content model and attribute list declarations *SHOULD* start with a descriptive comment. For example:

```
<!--                    LONG NAME: Topic Head                      -->
```
Each element type **MUST** have a corresponding content model parameter entity named `%tagname.content`. The value of the entity **MUST** be the complete content model definition. For example:

```xml
<!ENTITY % topichead.content
"((%topicmeta;)?,
  (%anchor; | %data.elements.incl; | %navref; | %topicref;)*)">
```

The content model parameter entity can be overridden in shell DTDs or constraint modules to further constrain the content model for the element type.

Each element type **MUST** have a corresponding attribute list parameter entity named `%tagname.attributes`. The parameter entity must declare all attributes used by the element type (except for the attributes provided by the `%global-atts;` parameter entity, which is always referenced as part of the attribute list declaration for an element's class attribute). For example:

```xml
<!ENTITY % topichead.attributes
"navtitle
 CDATA
  #IMPLIED
outputclass
 CDATA
  #IMPLIED
keys
 CDATA
  #IMPLIED
%topicref-atts;
%univ-atts;">
```

The ELEMENT declaration for each element type **MUST** consist entirely of a reference to the corresponding content model parameter entity:

```xml
<!ELEMENT topichead    %topichead.content;>
```

The ATTLIST declaration for each element type **MUST** consist entirely of a reference to the corresponding attribute list parameter entity:

```xml
<!ATTLIST topichead    %topichead.attributes;>
```

The content model parameter entity, attribute list parameter entity, ELEMENT declaration, and ATTLIST declaration **SHOULD** be grouped together within the module. Each such group of declarations may occur in any order within the module. For example:

```xml
<!--                    LONG NAME: Topic Head                      -->
<!ENTITY % topichead.content
"((%topicmeta;)?,
  (%anchor; | %data.elements.incl; | %navref; | %topicref;)*)">
<!ENTITY % topichead.attributes
"navtitle
 CDATA
  #IMPLIED
outputclass
 CDATA
  #IMPLIED
keys
 CDATA
  #IMPLIED
%topicref-atts;
%univ-atts;">
<!ELEMENT topichead    %topichead.content;>
<!ATTLIST topichead    %topichead.attributes;>
```
Attributes

Comment by sethpark

[RDAnderson, 4 Jan 2010] In the final section about attributes, we should provide a pointer to the topic in the specialization section, "Element type class hierarchy declaration (the class attribute)". This section as written leaves out some details, without mentioning that more is required; we should indicate that complete details of constructing the @class attribute are in the other topic.

The attributes of an element type MUST restrict or conserve those of the element type it specializes. Specialized element types may not add new attributes. New global attributes may be defined via attribute domain modules. Structural modules might require the use of attribute domain modules.

A vocabulary module MUST define a @class attribute for every specialized element declared in the module. The @class attribute MUST include the value of the @class attribute of the base element, and append to it the element name qualified by the topic element name with at least one leading and trailing space. The @class attribute for an element introduced by a structural specialization MUST start with a minus sign ("-"). The @class attribute for a domain specialization MUST start with a plus sign ("+"). The initial minus or plus sign MUST be followed by one or more spaces. The attribute value MUST end with one or more trailing spaces.

The ATTLIST declaration for the @class attribute MUST also include a reference to the %global-atts parameter entity.

For example, the ATTLIST definition for the <conbody> element (a specialization of the <body> element in the <topic> base type) includes global attributes with an entity, then the definition of the @class attribute, as follows:

```xml
<!ATTLIST conbody %global-atts; class CDATA "- topic/body concept/conbody">
```

The @class attribute declarations for a module MUST be grouped together at the end of the module after any other declarations. The declarations may occur in any order. By convention they are often ordered alphabetically or grouped logically.

See class attribute (see page 144) for complete details on the @class attribute.

3.4.3.11.1.2 Structural module coding requirements

A structural vocabulary module defines a new topic or map type as a specialization of a base topic or map type. The purpose is usually to enhance the user's interaction by adapting the topic or map type to its particular purposes.

A structural type module must conform to the following coding requirements in addition to the general module coding requirements:

**Default included domains entity**

The module must define the included-domains entity with a default value, as in the following example:

```xml
<!ENTITY included-domains "">
```

A document type shell can predefine the included-domains entity to list domains to be added to the document type.

Structural vocabulary modules may require the use of specific domains. In that case, the default value of the included-domains entity must include the appropriate domain use declaration, for example:

```xml
<!ENTITY included-domains "{topic myDomain}">
```

The list of included domains must declare the domains from most generic (on the left) to most specialized, current domain (on the right). See Domain usage declaration (the @domains attribute).
**Topic and map element attributes**

The topic or map element type must set the @DITAArchVersion attribute to the DITAArchVersion entity and the @domains attribute to the included-domains entity. These attributes give processors a reliable way to check the architecture version and look up the list of domains available in the document type.

```xml
<!ATTLIST concept
 %concept.attributes;
 %arch-atts;
 domains CDATA
 "&included-domains;"
>
```

### 3.4.3.11.1.3 Topic type module coding requirements

Topic type vocabulary modules must conform to additional coding requirements for defining default topic nesting.

**Default nested topics entity**

A topic type module must define an entity to specify default subordinate topics. The entity name must be the topic element name plus the -info-types suffix. For example, the info-types entity for the concept topic is concept-info-types. If the topic has default subordinate topics, this entity can default to a list of element entities. If not, the entity can default to the value of the info-types entity as in the following example:

```xml
<!ENTITY % concept-info-types "%info-types;">
```

A document type shell can then control how topics are allowed to nest by redefining the topic-type-info-types entity for each topic type, or it can efficiently create common nesting rules by redefining the main info-types entity.

In the declaration of the root element of a topic type, the last position in the content model must be the topic-type-info-types nested topics entity, as in the following example:

```xml
<!ENTITY % concept.content "(%title;),
 (%titlealts;)?,
 (%abstract; | %shortdesc;)?,
 (%prolog;)?,
 (%conbody;)?,
 (%related-links;)?,
 (%concept-info-types;)*">
```

### 3.4.3.11.1.4 Element domain module coding requirements

An element domain vocabulary module defines element types that are appropriate for the subject-matter or application domain for which they are designed. The purpose is usually to enhance the user's interaction by providing semantic elements whose names more accurately denote their content, making that content easier to search and retrieve.

**Domain entity file**

In addition to the domain declaration entity, the entity declaration file for element domain modules must include the following components:

**Element extension entity**

The declaration (.ent) file must define an entity for each element extended by the domain. The base of the entity name is the abbreviation for the domain and the extension of the entity name is the name of the extended element. For example, the highlight domain (abbreviated as hi-d) extends the ph element, so the entity for the extended element is named hi-d-ph.
The value of the entity is a list of the specialized elements that are intended to occur in the same locations as the extended element. Each element must be separated by the vertical line ( | ) symbol.

For example, the hi-d-ph entity is defined as follows:

```
<!ENTITY % hi-d-ph "b | u | i | line-through | overline | tt | sup | sub">
```

### 3.4.3.11.1.5 Attribute domain module coding requirements

An attribute domain vocabulary module declares a new attribute specialized from either the @props or @base attribute. An attribute domain module defines exactly one new attribute type.

An attribute domain's name is the name of the attribute plus "Att" to distinguish the domain from any element domains with the same name. For example, for an attribute named "new" the attribute domain name would be "newAtt". The attribute domain name is used to construct filenames and entity names for the domain.

An attribute domain must consist of one file, whose name consists of the module name plus Domain plus the ent extension. For example: newAttDomain.ent for an attribute named "new".

The file must have two parts:

**Attribute extension entity**

The attribute declaration is in an entity. This entity can then be used in document type shells to add the new attribute. The attribute declaration entity name consists of the attribute name plus "-d-attribute". For example:

```
<!ENTITY % newAtt-d-attribute "new   CDATA #IMPLIED">
```

For an attribute named "new".

**Domain declaration entity**

The attribute domain is declared in @domains attribute values through a general text entity that contains the attribute domain's domain declaration fragment. The entity name consists of the module name plus "-d-att". For example, "newAtt-d-att" for an attribute named "new". See domains attribute (see page 146) for details on attribute domain @domains values.

For example:

```
<!ENTITY newAtt-d-att       "a(props new)"  >
```

Attribute domains do not have domain module declaration (.mod) files.

### 3.4.3.11.2 XML Schema: Coding requirements for DITA specialization modules

To be extensible and backward compatible, DITA requires that an XSD implementation of structural and domain specialization modules conform to well-defined implementation (coding) requirements.

These design patterns implement the specialization architecture with the capabilities and within the limitations of the XML Schema grammar. They are the coding requirements for structural specializations, domain specializations, and attribute domain specializations.

#### 3.4.3.11.2.1 General element type declaration coding requirements

Structural and element domain vocabulary modules MUST reflect the same coding requirements for element type declarations.

**Comment by sethpark**

[Jeff, 3 Jan 2010] I don't think it makes sense to have both a section for "General element type declaration coding requirements" and another for "Element domain module coding requirements". We should either combine them or give them better titles and otherwise reword them to make it clear what they are about what how they
differ from each other. I'd favor combining them. I'm none too sure about the differences between the other sections on "Structural" and "Topic type" module coding requirements either. There seems as if there is a good deal over overlap between the sections, which I find confusing.

Comment by Kristen Eberlein, 5 April 2014
Comment from Eliot Kimber during review #1: "Looks like the normative terms have not been updated in this topic."

Module names

Each vocabulary module has a short name that is used to construct file names, entity names, and other names used in associated declarations.

For structural modules, the module name must be the element type name of the top-level topic or map type defined by the module, such as "concept", "bookmark".

For element domain modules, the module name must be a name that reflects the subject domain to which the domain applies, such as "highlight", "software". Domain module names should be sufficiently unique that they are unlikely to conflict with any other domains.

For attribute domain modules, the module name must be the name of the attribute defined by the module plus "Att" (to avoid conflict with similarly-named structural types or element domains).

Element definitions

A structural or domain vocabulary module must contain a declaration for each specialized element type named by the module. While the XSD standard allows content models to refer to undeclared element types, all element types named in content models within a vocabulary module must have an xs:element declaration, either in the vocabulary module, in a base module of which the vocabulary module is a direct or indirect specialization, or, for structural modules, in a required domain module. The specialized elements must follow the rules of the architecture in defining content models and attributes.

Domain modules must consist of a single XSD document named modulenameMod.xsd. Structural modules must consist of two modules, modulenameGrp.xsd, which contains all element name groups, and modulenameMod.xsd, which contains all other declarations for the module.

For each element type declared in the vocabulary module there must be an xs:group whose name is the element type name and whose one member is a reference to the element type, e.g.:

```xml
<xs:group name="codeph">
<xs:sequence>
  <xs:choice>
    <xs:element ref="codeph"/>
  </xs:choice>
</xs:sequence>
</xs:group>
```

The element name group provides a layer of abstraction that facilitates redefinition. A document type shell can redefine an element group to add domain-specialized elements or replace a base element type with one or more specializations of that type.

For domain modules, the group definitions should be grouped together at the start of the domain's XSD document. The definitions may occur in any order.

Each element type must have a corresponding content model group named tagname.content. The value of the group must be the complete content model definition. For example:

```xml
<xs:group name="codeph.content">
<xs:sequence>
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:group ref="basic.ph.notm" minOccurs="0"/>  
    <xs:group ref="data.elements.incl" minOccurs="0"/>  
    <xs:group ref="foreign.unknown.incl" minOccurs="0"/>
  </xs:choice>
</xs:sequence>
</xs:group>
```
The content model group may be overridden in constraint modules to further constrain the content model for the element type.

Each element type must have a corresponding attribute group named tagname.attributes. The group must declare all attributes used by the element type except for the @class attribute. For example:

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

Each element type must have a complex type definition named tagname.class, which references the tagname.content and tagname.attributes groups. For example:

```xml
<xs:complexType name="codeph.class" mixed="true">
  <xs:sequence>
    <xs:group ref="codeph.content"/>
  </xs:sequence>
  <xs:attributeGroup ref="codeph.attributes"/>
</xs:complexType>
```

Each element type must have an xs:element declaration named tagname, that uses as its type the tagname.class complex type and extends that complex type to add the @class attribute for the element. For example:

```xml
<xs:element name="codeph">
  <xs:annotation>
    <xs:documentation>
      The code phrase (&lt;keyword>codeph</keyword>&gt;) element represents a snippet of code within the main flow of text. The code phrase may be displayed in a monospaced font for emphasis. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts and reference information.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent>
      <xs:extension base="codeph.class">
        <xs:attribute ref="class" default="+ topic/ph pr-d/codeph "/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

The content model group, attribute group, complex type, and element type definition for an element SHOULD be grouped together within the module. Each such group of declarations may occur in any order within the module. It is recommended to sort the element type definitions alphabetically or group them into categories. Here is an example declaration for the <codeblock> element:

```xml
<xs:element name="codeblock">
  <xs:annotation>
    <xs:documentation>
      The &lt;keyword>codeblock</keyword>&gt; element represents lines of program code. Like the <ph>element, content of this element has preserved line endings and is output in a monospaced font. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts and reference information.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent>
      <xs:extension base="codeblock.class">
        <xs:attribute ref="class" default="+ topic/pre pr-d/codeblock "/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```
Each xs:element declaration should include descriptive documentation as in the examples above.

### 3.4.3.11.2.2 Structural specialization coding requirements

An XSD structural module declares a top-level map or topic type, implemented as a pair of XSD documents, one that defines groups used to integrate and override the type and one that defines the element types specific to the type.

A structural type module must conform to the following coding requirements in addition to the general module coding requirements:

**Module files**

A structural vocabulary module must have two files:

- A module schema document. The file name is the name of the root structural element plus `Mod` plus the `.xsd` extension. For example, `conceptMod.xsd` is the module schema document for the concept topic type.
- A module group definition schema document. The file name is the name of the root structural element plus `Grp` plus the `.xsd` extension. For example, `conceptGrp.xsd` is the module group definition schema document for the concept topic type.

**Structural module schema document**

The root element must reference the `@DITAArchVersion` attribute and the `@domains` attribute. These attributes give processors a reliable way to check the architecture version and look up the list of domains available in the document type. The `@DITAArchVersion` attribute is referenced as in the following example:

```xml
<xs:attribute name="id" type="xs:ID" use="required"/>
<xs:attribute ref="ditaarch:DITAArchVersion" />
```

See XSD document-type shell: Coding requirements (see page 124) for information on how to set the values for the `domains` attribute for XSD shells.
For topic modules, the last position in the content model **MUST** be the `topictype-info-types` nested topics group as in the following example of the root element of the concept topic:

```xml
<xs:complexType name="concept.class">
  <xs:sequence>
    <xs:group ref="title"/>
    <xs:group ref="titlealts" minOccurs="0"/>
    <xs:choice minOccurs="0">
      <xs:group ref="shortdesc" />
      <xs:group ref="abstract" />
    </xs:choice>
    <xs:group ref="prolog" minOccurs="0"/>
    <xs:group ref="conbody" minOccurs="0"/>
    <xs:group ref="related-links" minOccurs="0"/>
    <xs:group ref="concept-info-types" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  ...
</xs:complexType>
```

**Topic module schema document**

For topic modules, the module schema document must define an info-type model group. The name of this group is the topic element name plus `info-types`. Thus, the info-type model group for the concept topic type is `concept-info-types`. The following example shows how this group is defined in `conceptMod.xsd`:

```xml
<xs:group name="concept-info-types">
  <xs:choice>
    <xs:group ref="concept" minOccurs="0"/>
    <xs:group ref="info-types" minOccurs="0"/>
  </xs:choice>
</xs:group>
```

**Domains attribute information for the XML Schema document-type shell integration**

A structural vocabulary module **SHOULD** include the a value for the `domain usage declarations` that's used in the `@domains` attribute in the XML Schema document-type shell in the modules. It **MUST** be defined using the element `<dita:domainsModule>`. For the XML Schema's the `<dita:domainsModule>` element, in the structural vocabulary module, is equivalent to `@domains` attribute definition in the `.ent` files for the DTDs.

**Comment by ekimber**

A structural vocabulary module **SHOULD** include the a value for the domain usage declaration that's used in the `@domains` attribute in the XML Schema document-type shell in the modules. It **MUST** be defined using the element `<dita:domainsModule>` . For the XML Schema's the `<dita:domainsModule>` element, in the structural vocabulary module, is equivalent to `@domains` attribute definition in the `.ent` files for the DTDs.

For example:

```xml
<xs:annotation>
  <xs:appinfo>
  </xs:appinfo>
  <xs:documentation></xs:documentation>
</xs:annotation>
```

**3.4.3.11.2.3 Attribute domain coding requirements**

An attribute domain **must** consist of one file, whose name consists of the module name plus Domain plus the `xsd` extension. For example: `newAttDomain.xsd` for an attribute named "new". The file must have a single attribute group definition that contains the definition of the attribute itself, where the attribute group is named `attnameAtt-d-attribute`. 
For example, for an attribute named "new":

```
<xs:attributeGroup name="newAtt-d-attribute">
  <xs:attribute name="new" type="xs:string"/>
</xs:attributeGroup>
```

The attribute domain must be reflected in a shell document type XSD that integrates it. See `domains attribute` (see page 146) for details of attribute domain `@domains` values.

For example, if the attribute named "new" is a specialization of the `@props` attribute, the `@domains` value would be "a({props new})".

### 3.4.3.11.3 RELAX NG specialization modules: Coding requirements

To be extensible and backward compatible, DITA requires that RELAX NG-based structural and domain specialization modules conform to well-defined coding requirements.

These coding requirements implement the specialization architecture with the capabilities and within the limitations of the RELAX NG grammar. Structural specializations, element domain specializations, and attribute domain specializations **MUST** conform to these requirements, as well as the requirements defined in *Using RELAX NG for document type shells, vocabulary modules, and constraint modules* (see page 113).

Unlike DTD and XSD-based specialization modules, RELAX NG modules are self integrating. This means that document type shells only need to include the module, include the contribution to the default value for the `@domains` attribute, and list the element types that define the `@id` attribute as type "ID" in the override of the "any" pattern.

All vocabulary and constraint modules **MUST** document their `@domains` attribute contribution. This **MAY** be done by using the `<domainsContribution>` element within a `<moduleDesc>` element or by using a normal XML comment or `<a:documentation>` element. Grammars that are intended to be used for automatic generation of DTD- and XSD-based modules **MUST** use the `<domainsContribution>` element.

#### 3.4.3.11.3.1 RELAX NG: Coding requirements for element type declarations

Structural and domain vocabulary modules have the same coding requirements for element type declarations.

**Module names**

Each vocabulary module has a short name that is used to construct file names, pattern names, and other names that are used in associated declarations. Modules also can have abbreviated names that further shorten the short name, for example "sw" for the "software" domain, where "software" is the short name and "sw" is the abbreviated name.

For structural modules, the module name **MUST** be the element type name of the top-level topic or map type that is defined by the module, for example, "concept" or "bookmap".

For element domain modules, the module name **SHOULD** be a name that reflects the subject domain to which the domain applies, such as "highlight" or "software". Domain module names should be sufficiently unique that they are unlikely to conflict with any other domains.

---

**Comment by Kristen Eberlein, 15 February 2014**

This is the first instance of the term "element domain modules." Do we want to make this clear distinction between element domain and attribute domain modules? If so, we should do so consistently throughout the spec.

**Comment by Kristen Eberlein, 15 February 2014**

Doesn't the material in this section apply to DTD and XSD as well as RNG?

**Module files**

A RELAX NG vocabulary module consists of a single module file.
For modules, the file name takes the following format:

<table>
<thead>
<tr>
<th>Module type</th>
<th>File name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structural</td>
<td>moduleNameMod.extension</td>
</tr>
<tr>
<td>Domain</td>
<td>moduleNameDomainMod.extension</td>
</tr>
</tbody>
</table>

where `moduleName` is the module name, and `extension` is either "rng" or "rnc". Examples of file names for structural modules include `conceptMod.rng` and `glossentryMod.rnc`; examples of file names for domain modules include `highlightDomainMod.rng` and `programmingDomainMod.rnc`.

**Element definitions**

A structural or element-domain vocabulary module must contain a declaration for each specialized element type that is named in the module. While the XML standard allows content models to refer to undeclared element types, all element types that are named in content models or attribute list declarations within a vocabulary module must have a RELAX NG element declaration. The RELAX NG element declaration can occur in one of the following:

- The vocabulary module
- A base module of which the vocabulary module is a direct or indirect specialization
- (If the vocabulary module is a structural module) A required domain or structural module

The element type patterns are organized into the following sections:

**Element type name patterns**

This section *SHOULD* have the section heading "ELEMENT TYPE NAME PATTERNS"

```xml
<documentation>ELEMENT TYPE NAME PATTERNS</documentation>
<define name="b">
  <ref name="b.element"/>
</define>
```

For each element type that is declared in the vocabulary module, there must be a pattern whose name is the element type name and whose content is a reference to the element type "tagname.element" pattern. For example:

```xml
<documentation>ELEMENT TYPE NAME PATTERNS</documentation>
<define name="b">
  <ref name="b.element"/>
</define>
```

The element-type name pattern provides a layer of abstraction that facilitates redefinition. The element-type name patterns are referenced from content model and domain extension patterns. Specialization modules can redefine the patterns to include specializations of the type, allowing the specialized types in all contexts where the base type is allowed.

The declarations can occur in any order. By convention, they are usually ordered alphabetically or grouped logically.

**Common content-model patterns**

Structural and element-domain modules can include a section that defines the patterns that contribute to the content models of the element types that are defined in the module. This section *SHOULD* use the section header "COMMON CONTENT MODEL PATTERNS"

```xml
<documentation>COMMON CONTENT MODEL PATTERNS</documentation>
```

The declaration of the patterns can occur in any order. By convention, they are usually ordered alphabetically or grouped logically.
Common content-model patterns SHOULD use names that end with ".cnt", for example, "body.cnt".

**Common attribute sets**

Structural and element-domain modules can include a section that defines patterns for attribute sets that are common to one or more of the element types that are defined in the module. This section SHOULD use the section header "COMMON ATTRIBUTE SETS":

```
<div>
  <a:documentation>COMMON ATTRIBUTE SETS</a:documentation>
</div>
```

Common attribute set patterns SHOULD use names that end with "-atts", for example, "topicref-atts".

**Element type declarations**

For each element type that is declared in the vocabulary module, there must be a set of patterns that define the content model and attributes for the element type. Each set of patterns SHOULD be grouped within a `<div>` element and should start with a descriptive comment of the form:

```
<div>
  <a:documentation>LONG NAME: Long Name</a:documentation>
</div>
```

Each element type MUST have a content model pattern named `<tagname>.content`. The value of the pattern MUST be the complete content-model definition. For example:

```
<define name="topichead.content">
  <optional>
    <ref name="topicmeta"/>
  </optional>
  <zeroOrMore>
    <choice>
      <ref name="anchor"/>
      <ref name="data.elements.incl"/>
      <ref name="navref"/>
      <ref name="topicref"/>
    </choice>
  </zeroOrMore>
</define>
```

The content model pattern MAY be overridden in constraint modules to further constrain the content model for the element type.

Each element type MUST have an attribute list pattern named `<tagname>.attributes`. The pattern entity MUST declare all attributes that are used by the element type (except for the attributes provided by the "global-atts" pattern, which is always referenced as part of the attribute list declaration for the `@class` attribute for an element). For example:

```
<define name="topichead.attributes">
  <optional>
    <attribute name="navtitle"/>
  </optional>
  <optional>
    <attribute name="outputclass"/>
  </optional>
  <optional>
    <attribute name="keys"/>
  </optional>
  <optional>
    <attribute name="copy-to"/>
  </optional>
  <ref name="topicref-atts"/>
  <ref name="univ-atts"/>
</define>
```

The attribute list declaration MAY be overridden in constraint modules to further constrain the attribute list for the element type.
Comment by Kristen Eberlein, 15 February 2014
Should the wording be "can" here rather than "MAY"? I think we need to express possibility -- that the attribute list declaration can be overridden -- rather than MAY, which might suggest that constraint modules are optional features which processors can choose to implement or not as they please.

Each element type **MUST** have a pattern named `<tagname.element>` that contains an `<element>` element whose `@name` value is the element type name and whose content is a reference to the `<tagname.attlist>` and `<tagname.content>` patterns. The `<element>` element **SHOULD** include an `<a:documentation>` element that contains a short description of the element type. The description should not include any unescaped XML markup. For grammars intended for use in generating DTD- and XSD-based modules, the `<element>` element **SHOULD** specify the attribute "longName" in the DITA architecture namespace (http://dita.oasis-open.org/architecture/2005/, by convention bound to the prefix "ditaarch"). For example:

```xml
<define name="topichead.element">
  <element name="topichead" ditaarch:longName="Topic Head">
    <a:documentation>The &lt;topichead> element provides a title-only entry in a navigation map, as an alternative to the fully-linked title provided by the &lt;topicref> element. Category: Mapgroup elements</a:documentation>
    <ref name="topichead.attlist"/>
    <ref name="topichead.content"/>
  </element>
</define>
```

Each element type **MUST** have a pattern named `<tagname.attlist>` with a `@combine` value of "interleave" that contains only a reference to the `<tagname.attributes>` pattern. For example:

```xml
<define name="topichead.attlist" combine="interleave">
  <ref name="topichead.attributes"/>
</define>
```

Comment by Kristen Eberlein, 15 February 2014
We need a sentence or more explaining what this code sample illustrates or is an example of.

Comment by Kristen Eberlein, 5 April 2014
From a comment by Eliot Kimber in review #1: ". . . an example of a complete set of patterns for a single element type declaration minus the class attribute declaration."

```xml
<define name="topichead.element">
  <element name="topichead" ditaarch:longName="Topic Head">
    <a:documentation>LONG NAME: Topic Head</a:documentation>
    <optional>
      <ref name="topicmeta"/>
    </optional>
    <zeroOrMore>
      <choice>
        <ref name="anchor"/>
        <ref name="data.elements.incl"/>
        <ref name="navref"/>
        <ref name="topicref"/>
      </choice>
    </zeroOrMore>
  </element>
</define>
```
Element types that declare the @id attribute as type "ID", including all topic and map element types, MUST be included in the idElements pattern in order to correctly configure the "any" pattern overrides that are defined in document type shells. Each such element type MUST have a pattern declaration named "idElements" with a @combine value of "choice" and a reference to the element-type tagname.element pattern. The pattern declaration SHOULD follow the "tagname.attlist" pattern:

For each element type that is defined in the module there MUST be a pattern named tagname.attlist that contains a reference to the attribute list pattern "global-atts" and defines an optional attribute named "class". The default value for the @class attribute MUST include the value of the @class attribute of the base element; the element name qualified by the topic element name with at least one leading and trailing space must be appended. The @class attribute for an element that is introduced by a structural specialization MUST start with a minus sign ("-"). The @class attribute for a domain specialization MUST start with a plus sign ("+"). The initial minus or plus sign MUST be followed by one or more spaces. The attribute value MUST end with one or more trailing spaces.

The default value for the attribute is declared using the @a:defaultValue attribute.

For example:
The `@class` attribute declarations for a module **MUST** be grouped together at the end of the module after all other declarations. The `@class` attribute declarations can occur in any order. By convention, they are often ordered alphabetically or grouped logically.

See **class attribute** (see page 144) for more information about the `@class` attribute.

### 3.4.3.11.3.2 RELAX NG: Coding requirements for structural modules

A structural vocabulary module defines a new topic or map type as a specialization of a base topic or map type. A structural type module **MUST** conform to the following coding requirements in addition to the general coding requirements for modules and the rules defined in *Using RELAX NG for document type shells, vocabulary modules, and constraint modules* (see page 113).

**Architecture attributes**

The topic or map element type **MUST** declare the `@DITAArchVersion` attribute and set its value to "1.3". This attribute gives processors a reliable way to check the architecture version. In addition, because the `@DITAArchVersion` attribute is in a DITA-defined namespace, it serves as an unambiguous signal that the element is a DITA element.

The architectural attributes section **SHOULD** use the section header "ARCHITECTURE ATTRIBUTES":

```xml
<define name="arch-atts">
  <optional>
    <attribute name="ditaarch:DITAArchVersion" a:defaultValue="1.3"/>
  </optional>
</define>
```

**3.4.3.11.3.3 RELAX NG: Coding requirements for topic type modules**

Topic-type vocabulary modules **MUST** conform to additional coding requirements for defining *default topic nesting*. Default topic nesting defines whether topics are permitted to nest and what topic types are permitted to nest.

A topic type module **MUST** define a pattern to specify default topic nesting. The pattern name **MUST** be the topic element name plus the suffix -info-types. For example, the info-types pattern for the concept topic type is `concept-info-types`. The info type pattern section should use the section header "INFO TYPES PATTERNS":

```xml
<define name="mytopic-info-types">
  <ref name="subtopictype-01.element"/>
  <ref name="subtopictype-02.element"/>
</define>
```

If the topic has default topic nesting, this pattern **MAY** refer to a list of name patterns for topic element types, as in this example:

```xml
<define name="mytopic-info-types">
  <ref name="subtopictype-01.element"/>
  <ref name="subtopictype-02.element"/>
</define>
```
If the topic does not permit nested topics, the pattern SHOULD refer to the info-types pattern as in the following example:

```xml
<define name="mytopic-info-types">
  <ref name="info-types"/>
</define>
```

A document type shell then can control how topics are allowed to nest by redefining the "topictype-info-types" pattern for each topic type, or it can efficiently create common nesting rules by redefining the main info-types pattern.

In the declaration of the root element of a topic type, the last position in the content model MUST be the "topictype-info-types" pattern, as in the following example:

```xml
<define name="concept.content">
  <ref name="title"/>
  <optional>
    <ref name="titlealts"/>
  </optional>
  <optional>
    <choice>
      <ref name="abstract"/>
      <ref name="shortdesc"/>
    </choice>
  </optional>
  <optional>
    <ref name="prolog"/>
  </optional>
  <optional>
    <ref name="conbody"/>
  </optional>
  <optional>
    <ref name="related-links"/>
  </optional>
  <zeroOrMore>
    <ref name="concept-info-types"/>
  </zeroOrMore>
</define>
```

### 3.4.3.11.3.4 RELAX NG: Coding requirements for element domain modules

An element-domain vocabulary module defines element types that are applicable for a specific subject-matter or application domain. The purpose of an element domain is usually to provide semantic elements whose names more accurately denote their content, thus making content easier to search and retrieve.

**Comment by Kristen Eberlein , 15 February 2014**

The short description is quite generic and could be used across DTD- and XSD-specific topics as well.

**Domain extension patterns**

The domain extension patterns integrate domain modules into the base content models of the element types that the module extends, when the domain module is included in a document type shell. This section SHOULD use the section header "DOMAIN EXTENSION PATTERNS":

```xml
<define name="domain.extension">
  <ref name="domain-info-types"/>
</define>
```
For each element type that is extended by the element domain module, the module MUST define a domain extension pattern. The pattern consists of a choice group of references to the element-type name patterns for the domain-provided extensions of the extended element type.

The name of the pattern MUST use the following format:

```
shortName-d-elementName
```

where shortName is the short name for the domain, and elementName is the name of the element that is extended.

For example, the following pattern extends the `<ph>` element type by adding the specializations of `<ph>` that are defined in the highlighting domain:

```
<define name="hi-d-ph">
  <choice>
    <ref name="b.element"/>
    <ref name="i.element"/>
    <ref name="sup.element"/>
    <ref name="sub.element"/>
    <ref name="tt.element"/>
    <ref name="u.element"/>
  </choice>
</define>
```

For each element type that is extended by the element domain module, the module MUST extend the element type pattern with a `@combine` value of "choice" that contains a reference to the domain extension pattern.

For example, the following pattern adds the highlighting domain specializations of the `<ph>` element to the content model of the `<ph>` element:

```
<define name="ph" combine="choice">
  <ref name="hi-d-ph"/>
</define>
```

Because the extension of the element type pattern uses a `@combine` value of "choice", the effect is that the domain-provided elements are automatically added to the effective content model of the extended element in any grammar that includes the domain module.

### Inclusion of foreign vocabularies

In order to enable the generation of DTD- and XSD-based versions of the RELAX NG vocabulary, domain modules that include foreign vocabularies by using the `<externalRef>` element SHOULD specify the following attributes:

- `@ditaarch:dtidSystemId`
  (DTD; optional) Specifies the system ID URI as it should be specified on the external parameter-entity reference to the foreign vocabulary module

- `@ditaarch:dtidPublicId`
  (DTD) Specifies the public ID or URN as it should be specified on the external parameter-entity reference to the foreign vocabulary module

- `@ditaarch:xsdURI`
  (XSD) Specifies the URI reference as it should be specified on the `@schemalocation` attribute on the `<import>` element or `<include>` element

- `@ditaarch:xsdTargetNamespace`
  (XSD; required when the Schema targets a namespace) Specifies the namespace that the XML Schema targets
Example: RELAX NG code that includes a foreign vocabulary

The following code example illustrates markup in a RELAX NG vocabulary module that enables the generation of DTD and XSD-based modules.

```xml
<div><a:documentation>LONG NAME: RNG Grammar Container</a:documentation>
<define name="rng_grammar.content">
  <zeroOrMore>
    <choice>
      <externalRef href="rngGrammarProxy.rng"
        dita:namespacePrefix="rng"
        dita:dtdSystemId="rngGrammar.dtd"
        dita:xsdURI="rngGrammar.xsd"
        dita:xsdTargetNamespace="http://relaxng.org/ns/structure/1.0"
      />
      <ref name="data"/>
      <ref name="data-about"/>
    </choice>
  </zeroOrMore>
</define>
</div>
```

3.4.3.11.3.5 RELAX NG: Coding requirements for attribute-domain vocabulary modules

An attribute domain vocabulary module declares a new attribute specialized from either the \@props or \@base attribute. An attribute domain module defines exactly one new attribute type.

Comment by Kristen Eberlein, 5 April 2014

I missed incorporating this topic, so it was not present in review #1. It has not yet been edited. I did implement the XML mention domain and substituted `<varname>` elements for `<i>` elements.

An attribute domain's name is the name of the attribute plus "Att" to distinguish the attribute domain from any element domains with the same name. For example, for an attribute named "new" the attribute domain name would be "newAtt". The attribute domain name is used to construct file names and pattern names for the domain.

An attribute domain **MUST** consist of one file, whose name consists of the module name plus DomainMod plus the rng or rnc extension. For example: `newAttDomainMod.rng` for an attribute named "new".

The file **MUST** have three sections:

Domains attribute contribution

The `@domains` contribution for attribute domains uses the pattern "a(baseAttributeName specializedAttributeNames)" where `baseAttributeName` is either "base" or "props" and `specializedAttributeNames` is a space-separated list of attribute names reflecting the specialization hierarchy of the specialized attribute and ending with the name of the specialized attribute defined in the module.

**example:**

An attribute specialized directly from \@props:

```xml
<moduleDesc>
  <!-- ... -->
  <moduleMetadata>
    <!-- ... -->
    <domainsContribution>a(props deliveryTarget)</domainsContribution>
  </moduleMetadata>
</moduleDesc>
```

An attribute "myDeliveryTarget" specialized from \@deliveryTarget:

```xml
<moduleDesc>
  <!-- ... -->
  <moduleMetadata>
    <!-- ... -->
    <domainsContribution>a(props deliveryTarget myDeliveryTarget)</domainsContribution>
  </moduleMetadata>
</moduleDesc>
```
Attribute extension pattern

The attribute extension pattern extends either the @props or @base attribute set pattern to include the attribute specialization.

For specializations of @props the pattern MUST be named "props-attribute-extensions" and MUST specify a @combine value of "interleave". The content of the pattern MUST be a reference to the specialized attribute declaration pattern. For example:

```xml
<define name="props-attribute-extensions" combine="interleave">
  <ref name="deliveryTargetAtt-d-attribute"/>
</define>
```

For specializations of @base the pattern MUST be named "base-attribute-extensions" and MUST specify a @combine value of "interleave" The content of the pattern MUST be a reference to the specialized attribute declaration pattern. For example:

```xml
<define name="base-attribute-extensions" combine="interleave">
  <ref name="myBaseSpecializationAtt-d-attribute"/>
</define>
```

Attribute declaration pattern

The specialized attribute MUST be declared in a pattern named domainShortName-d-attribute. The attribute MUST be defined as optional. For example:

```xml
<define name="deliveryTargetAtt-d-attribute">
  <optional>
    <attribute name="deliveryTarget"/>
  </optional>
</define>
```

3.4.4 Constraints

Constraint modules define additional constraints for corresponding vocabulary modules in order to restrict content models or attribute lists for specific element types, remove extension elements from an integrated domain module, or replace base element types with domain-provided extension element types. Constraint modules do not and cannot change element semantics, only the details of how element types can be used in the context of a specific concrete document type. Because constraints can make optional elements required, documents that use the same vocabulary modules may still have incompatible constraints. Thus the use of constraints can affect the ability for content from one topic or map to be used directly in another topic or map.

Comment by Kristen Eberlein, 17 February 2014
Comment from Michael Priestley in e-mail to TC, 14 January 2014: "Says that constraints MAY be checked for validity during conref. Should be updated to explicitly set default as loose validation, with processor option to turn on strict, and make it clear that the rest of the topic only applies if strict validation is turned on."

Each constraint that is integrated into a DITA document type MUST be declared in the @domains attribute for each structural type that is integrated into the document type.

A constraint module can define any of the following types of constraint:

Restriction of content model for an element

Constraint modules can modify base content models by removing optional elements, making optional elements required, or requiring unordered elements to occur in a specific sequence. Constraint modules cannot make required elements optional or change the order of element occurrence for ordered elements.
For example, a constraint for `<topic>` could require `<shortdesc>`, could remove `<abstract>`, and could require that the first child of `<body>` be `<p>`. A constraint cannot allow `<shortdesc>` to follow `<prolog>`, because the base content model for `<topic>` declares `<shortdesc>` to precede `<prolog>`.

**Restriction of attributes for an element**

Constraint modules can restrict the attributes that are available on an element. They also can limit the set of permissible values for an attribute.

For example, a constraint for `<note>` could limit the set of allowed values for the @type attribute to "note" and "tip". It also could omit the @othertype attribute, since it is needed only when the value of the @type attribute is "other".

**Restriction of extension elements from a domain**

Constraint modules for element domains can restrict the set of extension elements that are provided by the element.

For example, a constraint on the programming domain could reduce the list of included extension elements to `<codeph>` and `<codeblock>`.

**Replacement of base elements by domain extensions**

Constraint modules can replace base element types with domain-provided extension elements.

For example, a constraint module could replace the `<ph>` element with the domain-provided elements, making `<ph>` unavailable.

When a domain module is integrated into a document type shell, the base domain element can be omitted from the domain extension group or parameter entity. In such a case, there is no separate constraint declaration, because the content model is configured directly in the document type shell. However, the constraint should be declared in the @domains attribute. For DTDs, there must be a domains module file that provides the constraint contribution to the @domains attribute. For XSDs and RELAX NG, the contribution to the @domains attribute is specified directly in the document type shell.

**Comment by Kristen Eberlein , 26 March 2014**

Eric Sirois, is the above statement correct for XSDs?

There can be at most one constraint module that defines the content model for a given element type that is included in a given document type. This means that constraints for the same element type that is defined in two different constraint modules cannot be aggregated together. In that case, a new constraint module must be created that reflects the aggregation of the two original constraints.

**Comment by Kristen Eberlein , 16 April 2014**

The above paragraph is difficult to read. It can -- and should -- be simplified.

**Constraint rules**

Constraint modules *MUST* conform to the following requirements:

- Designers must implement constrained content models for element types that are more restrictive than the unconstrained content models for the same element types.
- The content model and attributes of one element type can be constrained only by one constraint module included in a document type shell.
- The list of extension element types provided by a domain module can be constrained only by one constraint module included in a document type shell.
- Each constraint module may only constrain element types from one vocabulary module. This rule maintains granularity of reuse at the module level.
- Constraint modules that restrict different element types within the same vocabulary module can be combined with one another or with a constraint module that selects a subset of the extension element types for the
vocabulary. Such combinations of constraints on a single vocabulary module have no meaningful order or precedence.

- Designers have the option to declare a constraint module or combination of constraint modules to be more restrictive than another constraint module or combination of constraint modules on the same vocabulary module or a base vocabulary module. This option is particularly useful when a designer wants to constrain base and specialized element types in a consistent way. The advantage of declaring the consistency is that processors can take advantage of the consistency when converting document instances.

Comment by Kristen Eberlein, 8 April 2014
From review #1:
Kimber: "Where are the details of this constraint relationship declaration defined?"
Eberlein: "I have no earthly clue. This content is exactly as it was for 1.2 :(" 

For example, a constraint module for `<topic>` that requires both `<shortdesc>` and `<body>` is more restrictive than a constraint module that only requires `<body>`. By declaring this relationship, a designer can indicate that documents that use the first constraint also comply with the looser constraint.

**Content processing**

A document type with constraints allows a subset of the possible instances of a document type for the same vocabularies without constraints. To put it another way, all instances of the constrained document type are guaranteed to be valid instances of the unconstrained document type.

As a result, a constraint does not and cannot change basic or inherited element semantics. The constrained instances remain valid instances of the unconstrained element type, and the element type retains the same semantics and `@class` attribute declaration. Thus, a constraint never creates a new case to which content processing might need to react.

For example, a document type constrained to require the `<shortdesc>` element allows a subset of the possible instances of the unconstrained document type with an optional `<shortdesc>` element. Thus, the content processing for topic still works when `<topic>` is constrained to require a short description.

**Content interoperability**

A DITA document instance declares (by means of the `@domains` attribute and the `@class` attribute for the topic or map elements) the vocabularies that are available in its document type. A processor MAY examine these declarations to determine whether a document instance uses a subset of the vocabularies in another DITA document type and is thus compatible with the other DITA document type.

A constrained document type allows only a subset of the possible instances of the unconstrained document type. Thus, for a processor to determine whether a document instance is compatible with another document type, the document instance MUST declare any constraints on the document type.

For example, an unconstrained task is compatible with an unconstrained topic, because the `<task>` element can be generalized to `<topic>`. However, if the topic is constrained to require the `<shortdesc>` element, a document type with an unconstrained task is not compatible with the constrained document type, because some instances of the task might not have a `<shortdesc>` element. However, if the task document type also has been constrained to require the `<shortdesc>` element, it is compatible with the constrained topic document type.

3.4.4.1 DTD: Coding requirements for constraint modules

A structural constraint module defines the constraints for exactly one map or topic element type. A domain constraint module defines the constraints for exactly one element domain module.

Comment by Kristen Eberlein, 15 February 2014
The short description is very generic. If this is a unique topic about coding requirements for constraint modules that are implemented using DTDs, then the short description should say so.

Comment by Kristen Eberlein, 2 March 2014
This topic is a HEAVILY revised and corrected version of what we shipped for DITA 1.2. It is organized in a similar fashion as the new RELAX NG topic.

Robert Anderson has also suggested that we further revise this topic to clarify the two sorts of things that one does with constraints:

- Restricting the content model of an element
- Using a subset of a domain

Obviously, this would require redesigning and structuring the topic (and the XSD and RELAX NG topics), since "Restricting the content model of an element" could apply to either a structural element or a domain element.

**Structural constraint modules**

Structural constraint modules have the following requirements:

**Names**

Structural constraint modules SHOULD be named using the following format:

```
qualifierTagnameConstraint.mod
```

where:

- `qualifier` is a string that is specific to the constraints module and characterizes it, for example, "strict" or "requiredTitle".
- `Tagname` is the element type name with an initial capital, for example, “Taskbody.”

For example, the file name for the constraint that is applied to the general task to create the strict task type is `strictTaskbodyConstraint.mod`.

**Domain entity name and value**

The constraint module MUST contain a declaration for a general text entity with the following name:

```
"tagname-constraints"
```

where `tagname` is the name of the element type to which the constraints apply.

It also MUST contain the replacement text for the entity in the following format:

```
"(inheritance-hierachy qualifierTagname-c)"
```

where:

- `inheritance-hierachy` is the specialization hierarchy, for example, `topic task`.
- `qualifier` is a string that is specific to the constraints module and characterizes it, for example, "strict" or "requiredTitle".
- `Tagname` is the element type name with an initial capital, for example, "Taskbody".
- The literal "-c" indicates that the name is the name of a constraints domain.

For example, the following code provides the declaration for the strict task constraint that is shipped with the DITA standard.

```
<!ENTITY taskbody-constraints
  "(topic task strictTaskbody-c)"
>
```

**Declaration of the %tagname.content; parameter entity**

There also MUST be a declaration of the `%tagname.content;` parameter entity that defines the constrained content model.
The following parameter entity defines a more restricted content model for `<topic>`, in which either the `<abstract>` or `<shortdesc>` element is required.

```xml
<!ENTITY topic-constraints "(topic strictTopic-c)"

<!ENTITY % title         "title">
<!ENTITY % titlealts      "titlealts">
<!ENTITY % shortdesc      "shortdesc">
<!ENTITY % abstract       "abstract">
<!ENTITY % prolog          "prolog">
<!ENTITY % body            "body">
<!ENTITY % topic.content  "((%title;), (%titlealts;)?, (%abstract;| %shortdesc;), (%prolog;)?, (%body;)?, (%topic-info-types;)*)"
>```

**Domain constraint modules**

Domain constraint modules have the following requirements:

**Names**

Domain constraint modules **SHOULD** be named using the following format:

```
qualifierdomainDomainConstraint.mod
```

where:

- **qualifier** is a string that is specific to the constraints module and characterizes it, for example, "noSyntaxDiagram".
- **domain** is the name of the domain to which the constraints apply, for example, "Highlighting" or "Programming".

For example, the file name for a constraint module that removes the syntax diagram from the programming domain might be `noSyntaxDiagramProgrammingDomainConstraint.mod`.

**General text-entity declaration**

The constraint module **MUST** contain a declaration for a general text entity with the following name:

```
"tagname-constraints"
```

where **tagname** is the name of the element type to which the constraints apply.

**Replacement text for the general text-entity declaration**

The replacement text for the entity **MUST** be of the following form:

```
"(tagname qualifierTagname-c)"
```

where:

- **tagname** is the name of the element type to which the constraints apply.
- **qualifier** is a string that is specific to the constraints module and characterizes it, for example, "strict" or "requiredTitle".
- **Tagname** is the element type name with an initial capital, for example, "Topic".
- The literal "-c" indicates that the name is the name of a constraints domain.

**Declaration of the %tagname.content; parameter entity**

There also **MUST** be a declaration of the `%tagname.content;` parameter entity that defines the constrained content model.
The following parameter entity defines a more restricted content model for the highlighting domain, in which the elements are reduced to just `<b>` and `<i>`.

```
<!ENTITY % basicHighlight-c-ph  "b | i">
<!ENTITY   basicHighlight-c-att   "((topic hi-d basicHighlight-c))">
```

**Requirements for document type shells**

Information on how to integrate a constraint module into a DTD-based, document-type shell can be found in *DTD document-type shell: Coding requirements* (see page 119).

### 3.4.4.2 XML Schema: Coding requirements for constraint modules

A given constraint module corresponds to exactly one topic, map, or domain vocabulary module.

**Structural constraint modules**

Structural constraint modules have the following requirements:

**Names**

Structural constraint modules *SHOULD* be named using the following format:

```
qualifierTagnameConstraintMod.xsd
```

where:

- *qualifier* is a string that is specific to the constraints module and characterizes it, for example, "strict" or "requiredTitle".
- *Tagname* is the element type name with an initial capital, for example, "Taskbody".

**Intermediate names**

Because of restrictions on the redefine feature of XML Schema, it sometimes is necessary to use an intermediate level of redefinition, which requires a separate XSD document. The intermediate XSD document *SHOULD* be named using the following format:

```
qualifierTagnameConstraintIntMod.xsd
```

where:

- *qualifier* is a string that is specific to the constraints module and characterizes it, for example, "strict" or "requiredTitle".
- *Tagname* is the element type name with an initial capital, for example, "Taskbody".

---

**Comment by Kristen Eberlein, 26 March 2014**

Does the format need to be reworked to cover domain constraints as well as structural constraints? For example:

- Should *Tagname* be changed to *Object*?
- Should "*Tagname* is the element type name with an initial capital, for example, "Taskbody"" be changed to "*Object* is the element type name or domain name with an initial capital, for example, "Taskbody" or "Highlighting."

The code fragment shows how the `<topic>` element can be constrained to restrict the content model. In this example, the `<body>` element is disallowed. This `<xs:redefine>` element would be placed in a file named `<noBodyTopicConstraint.xsd>`.

```xml
<!-- ... -->
xsi:redefine schemaLocation="urn:oasis:names:tc:dita:xsd:topicMod:xsd:1.2">
xsi:group name="topic.content">
xsi:sequence>
  <xsi:sequence>
    <xsi:sequence>
      <xsi:group ref="title"/>
```
Domain constraint modules

Domain constraint modules have the following requirements:

Names

Domain constraint modules SHOULD be named using the following format:

\[
\text{qualifier}\text{domain}\text{Constraints.xsd}
\]

where:

- \textit{qualifier} is a string that is specific to the constraints module and characterizes it, for example, "strict" or "requiredTitle".
- \textit{domain} is the name of the domain to which the constraints apply, for example, "hi-d", "pr-d", or "mydomain-d".

Comment by Kristen Eberlein, 16 February 2014

A couple of things here. The above content is from the DITA 1.2 spec, simply with a new information design to improve clarity. But it does not match the file name in the example below. Note the following differences:

- The file name in the example uses the domain long name, not the short name.
- The literal in the example is "Constraint.xsd", not "Constraints.xsd".

Intermediate names

Because of restrictions on the redefine feature of XML Schema, it sometimes is necessary to use an intermediate level of redefinition, which requires a separate XSD document. The intermediate XSD document SHOULD be named using the following format:

\[
\text{qualifier}\text{domain}\text{DomainConstraintsInt.xsd}
\]

where:

- \textit{qualifier} is a string that is specific to the constraints module and characterizes it, for example, "strict" or "requiredTitle".
- \textit{domain} is the name of the domain to which the constraints apply, for example, "hi-d", "pr-d", or "mydomain-d".

Comment by Kristen Eberlein, 16 February 2014

A couple of things here. The above content is from the DITA 1.2 spec, simply with a new information design to improve clarity.

The following code fragment shows how the highlighting domain can be constrained to include only <b> and <i>. These declarations would be placed in a file named basicHighlightConstraint.xsd.

```
...<xs:group name="basicHighlight-c-ph">
  <xs:choice>
    <xs:element ref="b"/>
  </xs:choice>
</xs:group>
```
For each extension element type in the base vocabulary module whose content model or attributes are to be constrained in the constraint module, there **MUST** be a `<xs:redefine>` element that defines the restricted content model for the base element. Attributes for an element type **MAY** be constrained as part of the redefinition of the complex type.

**Comment by Kristen Eberlein, 2 March 2014**
I have a call scheduled with Eric Sirois to talk through the questions on Monday, 3 March 2014.

**Comment by Kristen Eberlein, 16 February 2014**
The above content is from the DITA 1.2 spec, simply with XML mention domain implemented, and correct markup for normative terminology. My questions:

- Does this apply to both structural constraint modules and domain constraint modules?
- What do we mean by “extension element type”? By “base vocabulary module”?
- Should this material be added to the “Structural constraint module” section, in a `<dlentry>` element named “Defining the restricted content model”?

---

**Comment by Kristen Eberlein, 16 February 2014**
My questions:
- Why is the note non-normative? Why is it a note?

For selective restriction, there **MUST** be a group with a subset list of extension elements for a domain in a reusable constraints module. The group name **SHOULD** be named using the following format:

```
"qualifierdomain-c-tagname"
```

where:
- **qualifier** is a description for the constraint vocabulary constraint module file.
- **domain** is the name of the domain, map, or topic being constrained.
- **tagname** is the name of the extension element being restricted.

---

**Comment by Kristen Eberlein, 16 February 2014**
The above content is from the DITA 1.2 spec, simply with a new information design to improve clarity. My questions:
Why do we mean by "selective restriction"?
Does this apply to both structural and domain constraints? The explanation of the domain variable suggests that it does, while the "Example of a domain constraint module" suggests that this applies only to an (element or attribute) domain module.
What do we mean by "with a subset list of extension elements for a domain in a reusable constraints module"?

Document type shells that integrate constraint modules

XSD-based document-type shells that integrate constraint modules MUST reflect the following requirements:

- (For content model constraints) Document type shells MUST include the constraint module instead of the vocabulary module that it constrains.
- (For selective extension) Document type shells MUST include the extension subset constraint module and use that group for domain or topic type extension.
- Document type shells MUST declare the constraints in the @domains attribute.

The code fragment demonstrates the markup that is used to constrain the standard <topic> element. These declarations would be placed in a document type shell named strictTopic.xsd.

```xml
<!-- ... -->
<xs:include schemaLocation="basicHighlightConstraint.xsd"/>
<!-- ... -->
<xs:redefine schemaLocation="commonElementGrp.xsd">
  <xs:group name="ph">
    <!-- drop base <ph> as well as apply basic subset of highlight domain -->
    <xs:choice>
      <xs:group ref="basicHighlight-c-ph"/>
    </xs:choice>
  </xs:group>
<!-- ... -->
</xs:redefine>
<!-- ... -->
<xs:redefine schemaLocation="strictTopicConstraint.xsd">
  <xs:complexType name="topic.class">
    <xs:complexContent>
      <xs:extension base="topic.class">
        <!-- declare the constraint of topic and highlight vocabulary modules and compatibility of constrained highlight with subset of topic constraints -->
        <xs:attribute name="domains" type="xs:string"
          default="(topic noBasePhrase-c)
          (topic strictTopic-c)
          (topic strictTopic-c hi-d basicHighlight-c)/">
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
<!-- ... -->
</xs:redefine>
```

Figure 35: Example of a document type shell that integrates a constraint module

Comment by Kristen Eberlein, 16 February 2014
There is a lot going on in the above example. Can we explicate it more thoroughly in the <desc> element? Clarify how it relates to the paragraph that preceded it?

3.4.4.3 RELAX NG: Coding requirements for constraint modules

A structural constraint module defines the constraints for exactly one map or topic element type. A domain constraint module defines the constraints for exactly one element or attribute domain.

Comment by Kristen Eberlein, 15 February 2014
The short description is very generic. If this is a unique topic about coding requirements for constraint modules that are implemented using RELAX NG, then the short description should say so.
**Structural constraint modules**

Structural constraint modules have the following requirements:

**Names**

Constraint modules *SHOULD* be named using the following format:

```
qualifierTagnameConstraintMod.extension
```

where:

- *qualifier* is a string that is specific to the constraints module and characterizes it, for example, "strict" or "requiredTitle".
- *Tagname* is the element type name with an initial capital, for example, "Taskbody".
- *extension* is either "rng" or "rnc".

For example, the file name for the constraint that is applied to the general task to create the strict task type is `strictTaskbodyConstraintMod.rng`.

@domains contribution pattern

The constraint-module @domains contribution pattern *MUST* take the following form:

```
(elementTypeName qualifierTagname-c)
```

where:

- *elementTypeName* is the name of the element to which the constraints apply, preceded by the topic or map type specialization hierarchy to which the domain applies, for example, `topic task`.
- *qualifier* is a string that is specific to the constraints module and characterizes it, for example, "strict" or "requiredTitle".
- *Tagname* is the element type name with an initial capital, for example, "Taskbody".

For example, the following code fragment specifies the contribution to the @domains attribute as `(topic task strictTaskbody-c)`:

```
<moduleDesc>
  <!-- ... -->
  <moduleMetadata>
    <!-- ... -->
    <domainsContribution>(topic task strictTaskbody-c)</domainsContribution>
  </moduleMetadata>
  <!-- ... -->
</moduleDesc>
```

**Domain constraint modules**

A domain constraint module defines the constraints for exactly one element or attribute domain module. Domain constraint modules have the following requirements:

**Names**

Domain constraint modules *SHOULD* be named using the following format:

```
qualifierDomainDomainConstraintMod.extension
```

where:

- *qualifier* is a string that is specific to the constraints module and characterizes it, for example, strict" or "requiredTitle".
- *domain* is the name of the domain to which the constraints apply, for example, "Highlighting" or "Programming".
- *extension* is either "rng" or "rnc".
For example, the name of a domain constraint module that removes syntax diagrams from the programming domain is `noSyntaxDiagramProgrammingDomainConstraintMod.rng`.

The constraint-module @domains contribution pattern MUST take the following form:

```
(domain qualifierDomainName-c)
```

where:

- **domain** is the short name of the domain to which the constraints apply (for example, hi-d", "pr-d", "mydomain-d", or "deliveryTargetAtt-d"), preceded by the topic or map type specialization hierarchy to which the domain applies (for example, "topic task").
- **qualifier** is a string that is specific to the constraints module and characterizes it, for example, strict" or "required" or "requiredReqconds".
- **DomainName** is the domain name with an initial capital, for example, "Taskreq".

The following code fragment illustrates the domains contribution for a constraint module that makes the `<reqconds>` element required:

```xml
<moduleDesc>
  <!-- ... -->
  <moduleMetadata>
    <!-- ... -->
    <domainsContribution>(topic task taskreq-d requiredReqcondsTaskreq-c)</domainsContribution>
  </moduleMetadata>
</moduleDesc>
```

Comment by Kristen Eberlein , 15 February 2014

Is this section needed, or is the material covered elsewhere?

Requirements for document type shells

Constraint modules are integrated into document type shells simply by integrating the constraint module in the document type shell. See RELAX NG document-type shell: Coding requirements (see page 129). The @domains contribution of the constraint module MUST be reflected in the @domains value for the root map or topic type that is configured by the document type shell.

3.4.4.4 Conref and generalization for constraint modules

When documents use different constraints, conref and generalization processors can examine the @domains attribute to verify compatibility between the document instances.

Comment by Kristen Eberlein , 17 February 2014

Comment from Michael Priestley in e-mail to TC, 14 January 2014: "Says that constraints MAY be checked for validity during conref. Should be updated to explicitly set default as loose validation, with processor option to turn on strict, and make it clear that the rest of the topic only applies if strict validation is turned on."

Conref compatibility with constraints

To determine compatibility between two document instances, a conref processor can check the @domains attribute to confirm the following:

- The referencing document has a superset of the vocabulary modules in the referenced document.
- For each vocabulary module in the referenced document, the referencing document qualifies the common module with a subset of the constraints in the referenced document.

Some examples:
### Generalization and constraints

Similarly, to determine compatibility between a document instance and a target document type, a generalization processor can use the `@domains` and `@class` attributes for the document instance and the `@domains` attribute for the target document type to determine how to rename elements in the document instance. For each element instance, the generalization processor:

- Iterates over the `@class` attribute on the element instance from specific to general, inspecting the vocabulary modules.
- Looks for the first vocabulary module that is both present in the target document type and that has a subset of the constraints in the document instance.

If a module is found in the target document type, that module becomes the minimum threshold for the generalization of contained element instances.

If a module is not found, the document instance cannot be generalized to the target document type and, instead, can only be generalized to a less constrained document type.

Note that a document instance can always be converted from a constrained document type to an unconstrained document type merely by switching the binding of the document instance to the less restricted schema (which would also have a different `@domains` attribute declaration). No renaming of elements is needed to remove constraints.

#### Related Links

3.3.6 Content inclusion (conref) (see page 84)

The DITA `@conref`, `@conkeyref`, `@conrefend`, and related attributes provide a mechanism for reuse of content fragments within DITA topics or maps.

<table>
<thead>
<tr>
<th>Referencing</th>
<th>Referenced</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>(topic)</td>
<td>(topic shortdescReq-c)</td>
<td>Allowed - content model of referenced topic is more constrained</td>
</tr>
<tr>
<td>(topic shortdescReq-c)</td>
<td>(topic)</td>
<td>Prevented - content model of referenced topic is less constrained</td>
</tr>
<tr>
<td>(topic hi-d)</td>
<td>(topic hi-d basicHighlight-c)</td>
<td>Allowed - domain extension list of referenced document type shell is more constrained</td>
</tr>
<tr>
<td>(topic hi-d basicHighlight-c)</td>
<td>(topic hi-d)</td>
<td>Prevented - domain extension list of referenced document type shell is less constrained.</td>
</tr>
<tr>
<td>(topic hi-d)</td>
<td>(topic noBasePhrase-c) (topic hi-d)</td>
<td>Allowed - referencing document type shell doesn't replace base element with domain extensions.</td>
</tr>
<tr>
<td>(topic task) (topic hi-d basicHighlight-c)</td>
<td>(topic simpleSection-c task simpleTaskSection-c)</td>
<td>Allowed - referencing shell has a subset of the constraints of the referenced shell on the common vocabulary modules.</td>
</tr>
<tr>
<td>(topic shortdescReq-c task shortdescTaskReq-c) (topic hi-d basicHighlight-c)</td>
<td>(topic simpleSection-c task simpleTaskSection-c)</td>
<td>Prevented - referencing shell has constraints on common vocabulary modules that aren't in the referenced shell.</td>
</tr>
</tbody>
</table>
3.4.4.5 Examples of constraint declaration modules

This section provides examples of constraint declaration modules.

Constraining element content in a topic vocabulary module

A constraint module named shortdescReq redefines the content model of the <topic> element so that the <shortdesc> element is required, and the <abstract> element is disallowed. The DTD declarations for this module would be:

```xml
<!ENTITY shortdescReq.constraint
  "(topic shortdescReq-c)"
>
<!ENTITY % title         "title">
<!ENTITY % titlealts     "titlealts">
<!ENTITY % shortdesc     "shortdesc">
<!ENTITY % prolog        "prolog">
<!ENTITY % body          "body">
<!ENTITY % related-links "related-links">
<!ENTITY % topic.content
  "((%title;), (%titlealts;)?, (%shortdesc;), (%prolog;)?, (%body;)?, (%related-links;)?, (%topic-info-types;)*)"
>
Comment by jelovirta

<!ENTITY shortdescReq
 .-
constraint "((topic shortdescReq-c))" > <ENTITY % title "title"> <ENTITY % titlealts "titlealts"> <ENTITY % shortdesc "shortdesc"> <ENTITY % prolog "prolog"> <ENTITY % body "body"> <ENTITY % related-links "related-links"> <ENTITY % topic.content "((%title;), (%titlealts;)?, (%shortdesc;), (%prolog;)?, (%body;)?, (%related-links;)?, (%topic-info-types;)*)"
>

Constraining attributes in a topic vocabulary module

A constraint module named sectionAttributes redefines and restricts the attribute list for the <section> element so that the @id attribute is required and the <spectitle> attribute is removed. The DTD declarations for this module would be:

```xml
<!ENTITY sectionAttributes.constraint
  "(topic sectionAttributes-c)"
>
<!-- Original definition includes %univ-atts;, spectitle, and outputclass -->
<!-- Include individual pieces of univ-atts, to make ID required -->
<! ENTITY % section.attributes
  "id       CDATA   #REQUIRED
  %conref-atts;
  %select-atts;
  %localization-atts;
  outputclass CDATA   #IMPLIED"
>
Comment by jelovirta

<!ENTITY sectionAttributes
 .-
"(topic sectionAttributes-c)" >
```
Integrating a subset of the extension elements from a domain module

A constraint module named basicHighlight includes the `<b>` and `<i>` elements but not the `<line-through>`, `<overline>`, `<u>`, `<sub>`, `<sup>`, and `<tt>` elements from the highlighting domain. The DTD declarations for this module would be:

```xml
<!ENTITY basicHighlight-c-att "(topic hi-d basicHighlight-c)"

<!ENTITY % basicHighlight-c-ph "b | i">
```

The XSD declarations for this module would be:

```xml
<xsd:group name="basicHighlight-c-ph">
  <xsd:choice>
    <xsd:element ref="b"/>
    <xsd:element ref="i"/>
  </xsd:choice>
</xsd:group>
```

Applying multiple constraints to a single vocabulary module

A constraint module named simpleSection redefines the content models of the `<section>` and `<example>` elements to allow a single initial `<title>` element and to remove text and phrase elements. Because this constraint module redefines different elements than the shortdescReq constraint module shown above, both
modules can apply to the topic module. The order in which the constraint modules are listed is not significant. The DTD declarations for this module would be:

```xml
<!ENTITY simpleSection.constraints "(topic simpleSection-c)"
>
<!ENTITY % section.content "((%title;),
  (%basic.block; | %data.elements.incl; | %foreign.unknown.incl; | %div; | %sectiondiv;)*)
>
>
```

Note that this constraint module and the shortdescReq constraint module both constrain task but because they constrain different element types they do not conflict and can be used together. Each constraint module provides its own contribution to the @domains attribute, so that when integrated the effective value of the @domains attribute will include the declarations for both constraint modules, as well as the declarations for the other modules that are integrated by the shell document type, for example:

```xml
<!-- ... -->
(topic shortdescReq-c)
(topic simpleSection-c)
<!-- ... -->
```

A topic with elements replaced by domain extensions

A document type shell replaces the `<ph>` element with extension elements from the highlighting and programming domains. Because the highlighting and programming domains cannot be generalized to a topic without the `<ph>` element, the removal constraint must be declared on the topic module with a separate parenthetical expression.

The @domains attribute declaration:

```xml
(topic noBasePhrase-c)
(topic hi-d)
(topic pr-d)
```

3.5 Technical content specializations

3.5.1 Overview of the DITA 1.3 Specification: Technical Content

This section describes the DITA document types and specializations for technical content.

The document types and specializations included in the technical content package and described in this section were designed to meet the requirements of those authoring content for technically oriented products in which the concept, task, and reference information types provide the basis for the majority of the content. These information types are used by technical-communication and information-development organizations that provide procedures-oriented content to support the implementation and use of computer hardware, computer software, and machine-industry content. However, many other organizations producing policies and procedures, multi-component reports, and other business content also use the concept, task, and reference information types as essential to their information models.

The DITA technical content package includes the following document types and supporting structural specializations and constraints:

- Concept document type and structural specialization
- Reference document type and structural specialization
• Task document type (general task with the Strict Taskbody Constraint)
• General task document type and specialization
• Machinery task document type (general task with the Machinery Taskbody Constraint)
• Glossary entry (glossentry) document type and structural specialization
• Glossary group (glossgroup) document type and structural specialization
• Bookmap document type and structural specialization

The DITA technical content package includes the following domain specializations:

• Programming elements
• Software elements
• User interface elements
• Task requirements elements
• Extensible Name and Address Language (xNAL) elements
• Abbreviated form element
• Glossary reference (glossref) element

The DITA technical content package includes the following constraint modules:

• Strict Taskbody
• Machinery Taskbody

The technical content document type shells included in the DITA technical content package use information types included in other packages, and other packages use the information types from the technical content package.

The technical content package consists of the technicalContent, machineIndustry, and xnal directories, as well as the map document type but not the base map information types.

### 3.5.2 Technical content: Document types

The Technical Content package contains various document types: concept, reference, general task, strict task, machinery task, troubleshooting, glossary entry, and glossary group. These document types are designed specifically for information that describes how to use products and processes. The Technical Content package also includes the map document type.

#### 3.5.2.1 Technical-content document-type modules

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bookmap</td>
<td></td>
<td>bookmap.dtd, bookmap.ent, bookmap.mod (DTD)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bookmap.xsd, bookmapGrp.xsd, bookmapMod.xsd (XML Schema)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bookmap.rng, bookmapMod.rng (RELAX NG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bookmap.rnc, bookmapMod.rnc (RELAX NG compact)</td>
</tr>
<tr>
<td>Concept</td>
<td>The concept document type provides conceptual information designed to support the performance of tasks. Concepts might</td>
<td>concept.mod, concept.ent (DTD)</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Files</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>General task</td>
<td>The general-task document type provides a less strict task model for task-oriented information than that provided in the strict task. Some organizations might prefer the general task model; it can facilitate the migration of legacy task content that does not follow the strict task topic model. The general-task document type serves as the base for the strict task and machine-industry-task document types.</td>
<td>task.mod, task.ent (DTD) conceptMod.xsd, conceptGrp.xsd (XML Schema) conceptMod.rng (RELAX NG) conceptMod.rnc (RELAX NG compact)</td>
</tr>
<tr>
<td>Glossary entry</td>
<td>The glossary-entry document type enables authors to develop glossary topics that define terms, acronyms, and abbreviations. It also can contain terminology information.</td>
<td>glossentry.dtd, glossentry.ent, glossentry.mod (DTD) glossentryMod.xsd, glossentryGrp.xsd (XML Schema) glossentry.rng, glossentryMod.rng (RELAX NG) glossentry.rnc, (RELAX NG compact)</td>
</tr>
<tr>
<td>Glossary group</td>
<td>The glossary-group document type enables authors to incorporate multiple glossary entries in a single collection tile.</td>
<td>glossgroup.dtd, glossgroup.ent, glossgroup.mod (DTD) glossgroup.xsd (XML Schema) glossgroup.rng, glossgroupMod.rng (RELAX NG) glossgroup.rnc, glossgroupMod.rnc (RELAX NG compact)</td>
</tr>
<tr>
<td>Machinery task</td>
<td>The machinery-task document type provides procedural information, similar to other task types; it has a well-defined semantic structure to meet the special needs of machinery documentation.</td>
<td>machineryTask.dtd, machineryTaskbodyConstraint.mod (DTD)</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Files</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Files</td>
</tr>
<tr>
<td>Reference</td>
<td>The reference document type provides for the separation of fact-based information from concepts and tasks. Factual information might include tables and lists of specifications, parameters, parts, commands, and other information that the users are likely to look up. The reference document type enables fact-based content to be maintained by those responsible for its accuracy and consistency.</td>
<td>reference.mod, reference.ent (DTD) referenceMod.xsd, referenceGrp.xsd (XML Schema) referenceMod.rng (RELAX NG) referenceMod.rnc (RELAX NG compact)</td>
</tr>
<tr>
<td>Task (strict)</td>
<td>The task document type provides procedural information to support the performance of a task. The task document type is built by combining the general-task document type and the strict &lt;taskbody&gt; constraint. This information model provides detailed semantics to encourage authors to label standard parts of the task, including prerequisites, sufficient conceptual information required to perform the task, the commands that introduce each step in a procedure, additional support information required to understand a step, the result of performing the task, and examples that demonstrate the performance of the task.</td>
<td>task.mod, task.ent, strictTaskbodyConstraint.mod (DTD) taskMod.xsd, taskGrp.xsd, strictTaskbodyConstraintMod.xsd (XML Schema) taskMod.rng, strictTaskbodyConstraint.rng (RELAX NG) taskMod.rnc, strictTaskbodyConstraint.rnc (RELAX NG compact)</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>The troubleshooting document type provides markup for corrective action information such as troubleshooting and alarm clearing. The troubleshooting information type describes a condition or a symptom, after which it provides one or more possible causes and their associated remedies. Each remedy is a series of steps, associated with a cause, that</td>
<td>troubleshooting.mod, troubleshooting.ent (DTD) troubleshootingMod.xsd, troubleshootingGrp.xsd (XML Schema) (RELAX NG) (RELAX NG compact)</td>
</tr>
</tbody>
</table>
3.5.2.2 Concept topic

Concept topics are specialized from topic. They include the standard topic elements, including the short description, prolog, a body, and related links.

*The purpose of the concept information type*

Concepts provide background that helps readers understand essential information about a product, a task, a process, or any other conceptual or descriptive information. A concept may be an extended definition of a major abstraction such as a process or function. Conceptual information may explain the nature and components of a product and describe how it fits into a category of products. Conceptual information helps readers to map their knowledge and understanding to the tasks they need to perform and to provide other essential information about a product, process, or system.

*The structure of the concept topic*

The concept topic is specialized from the base topic information type. The top-level element for a DITA concept topic is the `<concept>` element. Every concept topic contains the standard topic elements, including title, short descriptions or abstract, prolog, a body, and related links.

The `<conbody>` element holds the main body-level elements of the concept topic. Like the `<body>` element of a base topic, the `<conbody>` allows paragraphs, lists, tables, figures and other general elements. It also provides two key elements that allow authors to subdivide the topic into parts, with or without titles. These subdivisions are called sections and examples. The `<conbody>` also allows `<div>`, `<bodydiv>`, and `<sectiondiv>` to facilitate grouping elements in the `<conbody>`.

*Limitations within `<conbody>`*

The `<conbody>` provides for an unlimited number of subdivisions in the form of sections and examples. However, once an author decides to incorporate a section or example in the `<conbody>`, only additional sections or examples are allowed. Sections and examples may not nest, meaning that only one level of subdivision is permitted in the concept topic.

*Concept body primary subdivisions*

`<section>`

Represents an organizational division in a concept topic. Sections organize subsets of information within a larger topic. You can only include a simple list of peer sections in a topic; sections cannot be nested. A section may have an optional title.

`<example>`

Provides examples that illustrate or support the current topic. The `<example>` element has the same content model as `<section>`.

Following is an example of a simple concept topic. Note that once an example is used, it may be followed only by another example or by a section.

```plaintext
<concept id="concept">
<title>Bird Songs</title>
<shortdesc>Bird songs are complex vocalizations used to attract mates or defend territories.</shortdesc>
<body>
<p>Bird songs vary widely among species, from simple songs that are genetically imprinted to complex songs that are learned over a lifetime.</p>
<example>
<p>Flycatchers know their songs from birth:</p>
</example>
</body>
</concept>
```
3.5.2.3 Reference topic

Reference topics are specialized from topic. They contain the standard topic elements, including title, short descriptions or abstract, prolog, a body, and related links.

Purpose of the reference topic

Reference topics provide data that supports users as they perform a task. Reference topics might provide lists and tables that include product specifications, parts lists, and other data that is often “looked up” rather than memorized. A reference topic also can describe commands in a programming language or required tools for a series of maintenance tasks.

Reference topics provide quick access to fact-based information. In technical information, reference topics are used to list product specifications and parameters, provide essential data, and provide detailed information on subjects such as the commands in a programming language. Reference topics can contain any subject matter that has regular content, such as ingredients for food in recipes, bibliographic lists, catalog items, and so on.

The structure of the reference topic

The top-level element for a reference topic is the `<reference>` element.

The `<refbody>` element contains the main body-level elements of the reference topic. Reference topics limit the body to tables (both simple and complex), property lists, syntax sections, generic sections, and examples.

All of the elements of `<refbody>` are optional; they may appear in any sequence and number.

Limitations on the reference body

The `<refbody>` provides for an unlimited number of subdivisions in the form of sections, examples, syntax sections, property lists, and tables. Simple and complex tables may appear within sections, examples, and syntax sections. They may not appear within the property list or simple or complex table sections. Sections, examples, syntax sections, table subdivisions, and property lists may not nest; only one level of subdivision is permitted in the reference topic.

The elements in the reference body

`<section>`

Represents an organizational division in a reference topic. Sections organize subsets of information within a larger topic. You can only include a simple list of peer sections in a topic; sections cannot be nested. A section may have an optional title.

`<refsyn>`

Contains syntax or signature content (for example, a command-line utility’s calling syntax or an API’s signature). The `<refsyn>` contains a brief, possibly diagrammatic description of the subject’s interface or high-level structure.

`<example>`

Provides examples that illustrate or support the current topic. The `<example>` element has the same content model as `<section>`.

`<table>`

Organizes information according into a rows and columns. Table markup also allows for more complex structures, including spanning rows and columns, as well as table captions.
<simpletable>
Holds information in regular rows and columns and does not allow a caption.
</simpletable>

<properties>
Lists properties of a subject and their types, values, and descriptions.
</properties>

Following is an example of a simple reference topic, including the <refsyn> element.

```
<reference id="boldproperty">
<title>Bold property</title>
<shortdesc>(Read-write) Whether to use a bold font for the specified text string.</shortdesc>
<refbody>
<refsyn>
  <var>object</var>.<delim>.<kwdfont><kwdbold>Font</kwdbold>.<delim>.<var>trueorfalse</var>
</synph>
</refsyn>
<properties>
  <property>
    <proptype>Data type</proptype>
    <propvalue>Boolean</propvalue>
  </property>
  <property>
    <proptype>Legal values</proptype>
    <propvalue>True (1) or False (0)</propvalue>
  </property>
</properties>
</refbody>
</reference>
```

Following is an example of a simple reference topic, including the <property> element.

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

### 3.5.2.4 General task topic

The general-task document type was introduced in DITA 1.2. It provides a less-strict content model for task-oriented information than was available previously. Some organizations might prefer the general-task content model over the strict task model, as it can facilitate the migration of legacy content that does not follow the strict-task topic model. The general-task document type serves as the base for the strict-task and machine-industry-task document types. It also can be used to create new document types or specializations.
**The purpose of the general task information type**

Like the DITA strict task document type, the general task document and information types contain the essential building blocks to provide procedural information. Both task information types answer the "How do I?" question by providing step-by-step instructions detailing the requirements that must be fulfilled, the actions that must be performed, and the order in which the actions must be performed. Both task topics include sections for describing the context, prerequisites, expected results, and other aspects of a task.

The general task information type is specifically designed to accommodate task specializations that differ from the DITA task information type. It may also be used for the conversion of loosely structured tasks from other sources into DITA before they are restructured to follow the more restrictive DITA task model.

**The structure of the general task topic**

The `<task>` element is the top-level element for the general task topic. The general task topic contains a `<title>` and a `<taskbody>` with optional alternative titles (`<titlealts>`), a short description or `<abstract>`, a `<prolog>`, and `<related-links>`.

The following elements are described here because they are introduced as part of the general task model. All other elements are described in the strict task topic.

- `<section>`
  Represents an organizational division in a task topic. Sections organize subsets of information within the larger topic. Sections may not be nested. A section may have an optional `<title>`.

- `<steps-informal>`
  Describes procedural task information that would not normally be ordered as `<steps>`, such as a group of general procedures that may all be applied in a particular situation. Instead of `<step>`, the `<steps-informal>` element uses `<ol>` and `<ul>` elements, which are less strictly defined than the `<step>` element. When converting legacy content, it may be simpler to convert numbered lists to `<ol>` elements than to `<step>` elements.

**Comparison of general and strict task**

The following table compares the structures of general and strict task:

<table>
<thead>
<tr>
<th>General taskbody</th>
<th>Strict taskbody constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>prerequisite (optional, in any order, any number)</td>
<td>prerequisite (optional, one only, must precede context)</td>
</tr>
<tr>
<td>context (optional, in any order, any number)</td>
<td>context (optional, one only, must follow prerequisite)</td>
</tr>
<tr>
<td>section (optional, in any order, any number)</td>
<td>(not defined for strict taskbody)</td>
</tr>
<tr>
<td>steps</td>
<td>steps</td>
</tr>
<tr>
<td>steps-unordered</td>
<td>steps-unordered</td>
</tr>
<tr>
<td>steps-informal</td>
<td>(not defined for strict taskbody)</td>
</tr>
<tr>
<td>result (optional, one only, precedes example)</td>
<td>result (optional, one only, precedes example)</td>
</tr>
<tr>
<td>example (optional, any number, precedes post-req)</td>
<td>example (optional, one only, precedes post-req)</td>
</tr>
<tr>
<td>post-requisite (optional, any number)</td>
<td>post-requisite (optional, one only)</td>
</tr>
</tbody>
</table>

**3.5.2.5 Task topic (strict task)**

The strict task document type supports the development of instructions for the completion of a procedure. The strict task document type is built using the general task information type combined with the Strict Taskbody
The purpose of the standard task information type

Tasks are the essential building blocks to provide procedural information. A task information type answers the "How do I?" question by providing precise step-by-step instructions detailing the requirements that must be fulfilled, the actions that must be performed, and the order in which the actions must be performed. The <task> topic includes sections for describing the context, prerequisites, expected results, and other aspects of a task.

The structure of the <task> topic

The <task> element is the top-level element for the strict task topic. The strict task document type contains a <title> and a <taskbody> with optional alternative titles (<titlealts>), a short description or <abstract>, a <prolog>, and <related-links>.

The <taskbody> element is the main body element inside a strict task document type. The strict task body has a constrained structure, with these optional elements in the following order:

- <prereq>
  Describes information that the user needs to know or do before starting the immediate task. This section may occur only once.

- <context>
  Provides background information for the task. This information helps the users understand the purpose of the task and what they will gain by completing the task correctly. This section should be brief and does not replace or recreate a concept topic on the same subject, although the context section may include some conceptual information. This section may occur only once.

- <steps>
  Provides the main content of the task topic. A <task> consists of a series of steps that accomplish the task. The <steps> element must have one or more <step> elements, which provide the specifics about each step in the task. The <steps> element may occur only once.

  The <step> element represents an action that a user must follow to accomplish a task. Each <step> in a <task> must contain a command <cmd> element which describes the particular action the user must perform to accomplish the overall task. The <step> element may also contain information <info>, substeps <substeps>, tutorial information <tutorialinfo>, a step example <stepxmp>, choices <choices>, a step result <stepresult>, or troubleshooting <steptroubleshooting>, although these are optional.

- <steps-unordered>
  Provides alternative content for the <task> topic, allowing for a single step in a procedure or a set of commands that need not be performed in a specific order.

- <result>
  Describes the expected outcome for the task as a whole.

- <example>
  Provides an example that illustrates or supports the task.

- <postreq>
  Describes steps or tasks that the user should do after the successful completion of the current task. It is often supported by links to the next task or tasks in the <related-links> section.

Here is an example of a task topic:

```xml
<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>
    <!-- Task body content -->
  </taskbody>
</task>
```
Birdhouses provide safe locations for birds to build nests and raise their young. They also provide shelter during cold and rainy spells.

To build a sound birdhouse, you will need a complete set of tools:

- hand saw
- hammer
- ... 

Lay out the dimensions for the birdhouse elements.

Cut the elements to size.

Drill a 1 1/2" diameter hole for the bird entrance on the front.

You need to look at the drawing for the correct placement of the hole.

You now have a beautiful new birdhouse!

Now find a good place to mount it.

Maintaining specializations using the strict task model

Organizations that have created specializations based on the DITA 1.0 and 1.1 strict task model should review the recommendations in Migrating from DITA 1.1 to 1.2 (see page 687) to maintain their specializations.

3.5.2.6 Machinery task topic

The machinery task document type supports the development of instructions for the completion of a procedure. The machinery task document type is built using the general task information type combined with the Machinery Taskbody Constraint.

The purpose of the machinery task information type

The machinery-task is designed to provide procedural information, similar to the strict task topic, and has a well-defined semantic structure that describes how to perform the steps required to accomplish a specific goal. Compared to the strict task information type, the machinery-task information type contains additional descriptive elements in the prelreqs section that add detail to the pre-requisites required to perform a task. The machinery-task topic is developed using the DITA constraint mechanism, in addition to specializations for new elements.

Machinery tasks are the essential building blocks to provide procedural information for machines, machinery equipment, assemblies, and apparatuses. A machinery-task information type answers the “How do I?” question by providing precise step-by-step instructions detailing the requirements that must be fulfilled, the actions that must be performed, and the order in which the actions must be performed. The machinery-task topic includes sections for describing the context, preliminary requirements, expected results, examples, closing requirements, and other aspects of a task.

The structure of the machinery-task topic

Similar to a strict DITA task, the <task> element is the top-level element for a machinery task topic. The machinery task document type contains a title and a taskbody with optional alternative titles (titlealts), a short description or abstract, a prolog, and related-links.

The <taskbody> element is the main body element inside a machinery-task topic. A machinery-task body has a very specific structure, with the following elements in this order: (<prelreqs> or <context> or <section>)*, <steps>, <result>, <example>, and <closereqs>. Each of the body sections is optional.

The machinery task includes two specialized element groups: <prelreqs> and <closereqs>. All other element groups are the same as the general task model.

The preliminary-requirements section of a task is used to describe what the user needs to know or do before starting the immediate task. The <prelreqs> element is similar to the prerequisites section of the general task model but contains a more descriptive content model. The <prelreqs> element contains required conditions, required personnel, required equipment, supplies, spares, and safety information.
<closereqs>
The close-requirements section is used to describe conditions that must be fulfilled after the successful completion of the current task. It is often supported by links to the next task or tasks in the related-links section. The <closereqs> element contains required conditions <reqconds>.

### 3.5.2.7 Troubleshooting topic

The troubleshooting topic type provides markup for corrective action information such as troubleshooting and alarm clearing.

**The troubleshooting information type**

In its simplest form, troubleshooting information follows this pattern:

1. A condition or symptom. Usually the condition or symptom is an undesirable state in a system, a product, or a service that a reader wants to correct.
2. A cause for the condition or symptom.
3. A remedy for the condition or symptom.

The troubleshooting topic provides sections for describing the condition, causes, and remedies needed to restore a system, a product, or a service to normal.

For some conditions there could be more than one cause-remedy pair. The troubleshooting topic accommodates this. Typically, a cause is immediately followed by its remedy. Multiple cause-remedy pairs can provide a series of successive fall-backs for resolving a condition.

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

The troubleshooting information type also can be used to document alarm clearing strategies.

**The structure of the troubleshooting topic**

The top-level element for troubleshooting topics is `<troubleshooting>`. The `<troubleshooting>` element contains a `<title>` with optional alternative titles ( `<titlealts>` ), a short description or `<abstract>`, a `<prolog>`, a `<troublebody>`, and `<related-links>`.

<troublebody> is the main body element in a troubleshooting topic. The `<troublebody>` element contains the following elements:

- `<condition>`
  
  This optional element is the first child of `<troublebody>`, and it describes a condition or symptom that is associated with an undesirable state in a system, a product, or a service. In cases where the topic title fully explains the condition, this element should not be used.

- `<troubleSolution>`
  
  One or more `<troubleSolution>` elements must appear after the element in `<troublebody>` element. `<troubleSolution>` is a wrapper element for `<cause>` and `<remedy>`, each of which are a cause-remedy pair.

The `<troubleSolution>` element contains the following elements:

- `<cause>`
  
  This optional, repeatable, first-child of `<condition>` `<troubleSolution>` describes a possible cause for the condition.

- `<remedy>`
This optional, repeatable, last-child of `<troubleSolution>` describes a possible remedy for the condition.

The `<remedy>` element begins with an optional `<title>` element followed by an optional `<responsibleParty>` element followed by either a `<steps>` element, a `<steps-unordered>` element, or a `<steps-informal>` element. The content models for `<steps>`, `<steps-unordered>`, and `<steps-informal>` are borrowed from `<task>`. This allows remedy to reuse steps from tasks.

**<responsibleParty>**

This optional first child of `<remedy>` indicates who is expected to perform the steps that are outlined in the `<remedy>` element.

Here is an example of a troubleshooting topic:

```xml
<troubleshooting id="nologon">
  <title>Cannot log on</title>
  <shortdesc>Login attempts have failed</shortdesc>
  <troublebody>
    <condition>
      <p>The system does not accept your login credentials.</p>
    </condition>
    <troubleSolution>
      <title>Caps lock key</title>
      <cause>
        <p>The CapsLock key might be on.</p>
      </cause>
      <remedy><steps-unordered>
        <step>
          <cmd>Verify that the CapsLock key is off.</cmd>
        </step>
      </steps-unordered>
    </remedy>
    <troubleSolution>
      <title>Wrong password</title>
      <cause>
        <p>The password that you are using does not match the one that is stored in the system.</p>
      </cause>
      <remedy conref="#nologon/gotoaccountmanagement"/>
    </troubleSolution>
    <troubleSolution>
      <title>Unknown account name</title>
      <cause>
        <p>The account name you are using does not match the one stored in the system.</p>
      </cause>
      <remedy conref="#nologon/gotoaccountmanagement"/>
    </troubleSolution>
    <troubleSolution>
      <title>Still cannot log on</title>
      <steps-informal>
        <p>If none of the previous solutions work, consider asking for help. Contact your system administrator if your organization has one; otherwise, contact our support team.</p>
      </steps-informal>
    </troubleSolution>
  </troublebody>
</troubleshooting>
```
3.5.2.8 Glossary entry topic

Each glossary entry <glossentry> topic defines a single sense of one term. Besides identifying the term and providing a definition, the topic accommodates basic terminology information, such as part of speech. A glossentry topic may also include acronyms and acronym expansions. Glossentry topics may be assembled by authors or processes to create glossaries for various purposes, including books, websites, or other projects.

The purpose of the glossary entry topic

Defining terminology in a glossary ensures that a team of writers uses the same term for the same concept. A glossary added to a book or available online in conjunction with other subject matter provides the reader with definitions of unfamiliar terms and expands acronyms.

The structure of the glossentry topic

The top-level element for a DITA glossentry topic is the <glossentry> element. Every glossentry topic contains a <glossterm> and a <glossdef> element and optional <related-links>.

Where a term has multiple definitions, the writer should create multiple glossentry topics with the same term in the <glossterm> element but different definitions in the <glossdef> element. A process can collate and group glossentry topics by term when generating formatted output. Note that definitions with the same term in one language can have different terms in other languages, so translations can result in different collation and grouping of the same set of glossentry topics.

Here is an example of a simple glossentry topic:

```xml
<glossentry id="ddl">
  <glossterm>Data Definition Language</glossterm>
  <glossdef>A language used for defining database schemas.</glossdef>
</glossentry>
```

To create a glossary, authors can group multiple entries together by

- authoring in a single document using the Glossary group document type
- authoring in a single document under a container topic using the ditabase document type
- referencing the glossentry topics from a map
- using an automated process

For example, an automated process may assemble glossentry topics from a repository based on the <term> markup in a particular collection of topics.

Acronyms defined within glossentry topics

The glossentry topic may be used to provide expansions of acronyms in online text and assist in the proper translation of acronyms into multiple languages. The acronym elements of the glossentry topic include the following:

- <glossterm> to enter the full text to which the acronym refers
- <glossSurfaceForm> to provide the appropriate rendering of the full text plus the acronym in each language
- <glossAcronym> to provide the acronym text itself

Here is an example of an acronym used in the glossentry topic:

```xml
<glossentry id="wmd" xml:lang="en">
  <glossterm>Weapons of Mass Destruction</glossterm>
  <glossBody>
    <glossSurfaceForm>Weapons of Mass Destruction (WMD)</glossSurfaceForm>
    <glossAlt>
      <glossAcronym>WMD</glossAcronym>
    </glossAlt>
</glossentry>
```
Here is an example of how the glossentry topic would be translated into Spanish:

```xml
<glossentry id="wmd" xml:lang="es">
  <glossterm>armas de destrucción masiva</glossterm>
  <glossBody>
    <glossSurfaceForm></glossSurfaceForm>
    <glossAlt>
      <glossAcronym></glossAcronym>
    </glossAlt>
  </glossBody>
</glossentry>
```

Note that because no acronym exists for the term in Spanish, the `<glossSurfaceForm>` and `<glossAcronym>` elements are left blank.

In some languages, the surface form that expands the acronym in its first use handles the formatting differently than in English. For example, in Polish, the acronym precedes the expansion.

```xml
<glossentry id="eu" xml:lang="pl">
  <glossterm>Unia Europejska</glossterm>
  <glossBody>
    <glossSurfaceForm>UE (Unia Europejska)</glossSurfaceForm>
    <glossAlt>
      <glossAcronym>UE</glossAcronym>
    </glossAlt>
  </glossBody>
</glossentry>
```

For more information about the correct use of acronym expansions in multiple languages, see *Best Practice for Managing Acronyms and Abbreviations in DITA*, produced by the DITA Translation Subcommittee. [http://www.oasis-open.org/committees/download.php/29734/AcronymBestPractice_08112008.doc](http://www.oasis-open.org/committees/download.php/29734/AcronymBestPractice_08112008.doc)

### 3.5.2.9 Glossary group topic

The glossary group (`<glossgroup>`) topic enables authors to include one or more glossary entry (`<glossentry>`) topics in a single collection file, rather than authoring each glossary entry topic in a separate file. The glossary group topic is a specialization of concept.

### 3.5.2.10 Bookmap

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.

**The purpose of the bookmap specialization**

Books and other printed media are popular ways to present DITA content. By specializing the general DITA map structure into the general structure and subject areas used by most book-oriented DTDs, bookmaps enable users to organize their DITA information into front matter, parts, chapters, and so forth. A rich set of metadata allows for recording information about the book, such as its authors and owners, versions, and production history.

**The structure of the bookmap specialization**

The `<bookmap>` element is the top-level element for a DITA bookmap. Most of the content for a bookmap is optional, allowing for specializations that further restrict the bookmap model.

A bookmap allows the following parts:

- An initial title or booktitle (booktitle has more semantics)
- Book metadata (publisher, author, copyright holders and dates, etc.)
- Front matter (placement for Table of Contents and other preliminary information)
- Any number of chapters or parts (parts can group chapters, chapters can group topics)
A DITA domain defines a set of elements associated with a particular subject area or authoring requirement. DITA incorporates several domains into the Technical Content specializations. Other domains are incorporated into basic DITA.

The elements in a domain are defined in a domain module which can be integrated with a topic type to make the domain elements available within the topic type structure. Currently the following domains are provided as part of the Technical Content specializations:

Table 9: Technical content domains

*DITA includes domain specializations that are especially useful for authoring technical content. The classification, highlighting, utilities, hazard statement, and indexing domains associated with Base DITA are described under DITA Markup.*
<table>
<thead>
<tr>
<th>Domain</th>
<th>Description</th>
<th>Short name</th>
<th>Module name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbreviated form</td>
<td>For linking between a text reference to a <code>&lt;glossentry&gt;</code> topic. A specialization of <code>&lt;term&gt;</code> to provide an <code>&lt;abbreviated-form&gt;</code> element.</td>
<td>abbrev-d</td>
<td>abbreviateDomain.mod (DTD) abbreviateDomain.ent abbreviateDomain.xsd (Schema) abbreviateDomain.rng (RELAX NG) abbreviateDomain.rnc (RELAX NG Compact)</td>
</tr>
<tr>
<td>Equation</td>
<td>For containing equations and providing an integration point for MathML</td>
<td>equation-d</td>
<td>equationDomain.mod (DTD) equationDomain.ent equationDomain.xsd (Schema) equationDomain.rng (RELAX NG) equationDomain.rnc (RELAX NG Compact)</td>
</tr>
<tr>
<td>Glossary reference</td>
<td>For linking from a term to its glossary topic.</td>
<td>glossref-d</td>
<td>glossrefDomain.mod (DTD) glossrefDomain.ent glossrefDomain.xsd (Schema) glossrefDomain.rng (RELAX NG) glossrefDomain.rnc (RELAX NG Compact)</td>
</tr>
<tr>
<td>MathML</td>
<td>For containing MathML markup</td>
<td>mathml-d</td>
<td>mathmlDomain.mod (DTD) mathmlDomain.ent mathmlDomainMod.xsd (Schema) mathmlDomainMod.rng (RELAX NG) mathml3.rng mathmlDomainMod.rnc (RELAX NG Compact)</td>
</tr>
<tr>
<td>Programming</td>
<td>For describing programming and programming languages.</td>
<td>pr-d</td>
<td>programmingDomain.mod (DTD) programmingDomain.ent programmingDomain.xsd (Schema) programmingDomain.rng (RELAX NG) programmingDomain.rnc (RELAX NG Compact)</td>
</tr>
<tr>
<td>Release management</td>
<td>For describing release management metadata in a topic prolog</td>
<td>relmgmt-d</td>
<td>releaseManagementDomain.mod (DTD) releaseManagementDomain.net releaseManagementDomain.xsd (Schema) releaseManagementDomain.rng (RELAX NG)</td>
</tr>
<tr>
<td>Domain</td>
<td>Description</td>
<td>Short name</td>
<td>Module name</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------</td>
<td>------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Software</td>
<td>For describing software</td>
<td>sw-d</td>
<td>releaseManagementDomain.rnc (RELAX NG Compact)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>softwareDomain.mod (DTD)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>softwareDomain.ent (Schema)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>softwareDomain.xsd (Schema)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>softwareDomain.rng (RELAX NG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>softwareDomain.rnc (RELAX NG Compact)</td>
</tr>
<tr>
<td>User interface</td>
<td>For describing elements in a user interface</td>
<td>ui-d</td>
<td>uiDomain.mod (DTD)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>uiDomain.ent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>uiDomain.xsd (Schema)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>uiDomain.rng (RELAX NG)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>uiDomain.rnc (RELAX NG Compact)</td>
</tr>
</tbody>
</table>

The technical content domain specializations, like all domain specializations, may be included in the DITA document types beyond those in the technical content section. The DITA document types in the technical content section make use of other domain specializations in addition to those listed in the table.

The elements and attributes included in specific domain specializations are described in the domain elements of the DITA 1.2 Language Reference.

### 3.5.3.2 The xNAL domain

The DITA xNAL domain specialization defines a number of metadata elements and attributes that are useful in representing personal/organizational names and addresses. The metadata can be used to identify authors and content owners. The OASIS xNAL Standard (extensible Name and Address Language) was selected to represent close mappings from the DITA bookmap metadata content model to an existing standard. xNAL is included in the Bookmap and the LearningBookmap document types.

The OASIS Customer Information Quality (CIQ) standard for global-customer information management contains the definition of the **OASIS extensible Name and Address Language (xNAL)** metadata elements. Version 2 of the standard states:

The objective of xNAL is to describe a common structure for Personal/Organization Names and Addresses that would enable any applications that want to represent customer names and addresses in a common standard format. The applications could be CRM/e-CRM, Customer Information Systems, Data Quality (Parsing, Matching, Validation, Verification, etc.), Customer Data Warehouses, Postal services, etc.

However, any party for its own purposes and applications may use xNAL grammar or parts of it.

The DITA xNAL specialization is based on the OASIS extensible Name and Address Language metadata elements. Due to differences between the two processing architectures, the DITA xNAL domain does not incorporate all of the definitions from the OASIS xNAL standard directly. Instead, there is a transformational...
equivalence between the DITA and OASIS xNAL definitions for names and addresses. This equivalence enables XML-aware tools in workflow systems to capture and manipulate names and addresses in a standard manner.

The xNAL domain is available for use in the <bookmap> and <learningBookmap> document types, which are distributed as part of the DITA 1.2 specification. It can be included in specialized DITA document types that require metadata for names and addresses.

3.5.3.3 xNAL usage guidelines

Extended information and usage examples for DITA bookmap metadata elements associated with OASIS xNAL Standard (extensible Name and Address Language).

DITA bookmaps use a set of elements associated with a subset of the OASIS extensible Name and Address Language (xNAL) specification (Version 2) to denote name and address information related to persons and organizations.

While the elements share element names, and the expectation is that content written using this type of metadata should be straightforward to transform, the element name pairs do not share content models. The difference in content models reflects the different purposes of the two standards. The purpose of the name and address elements in DITA is to identify persons or organizations associated with the creation of a document; the purpose of the name and address elements in xNAL is to support customer resource management.

The examples shown after the table provide sample tagging methods for name and address information, using the DITA elements associated with xNAL.

**DITA elements associated with xNAL elements**

The set of bookmap elements associated with elements from the OASIS extensible Name and Address Language (xNAL) standard are listed in the table below.

<table>
<thead>
<tr>
<th>DITA Elements</th>
<th>xNAL Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;addressdetails&gt;</td>
<td>&lt;honorific&gt;</td>
</tr>
<tr>
<td>&lt;administrativearea&gt;</td>
<td>&lt;lastname&gt;</td>
</tr>
<tr>
<td>&lt;authorinformation&gt;</td>
<td>&lt;locality&gt;</td>
</tr>
<tr>
<td>&lt;contactnumber&gt;</td>
<td>&lt;localityname&gt;</td>
</tr>
<tr>
<td>&lt;contactnumbers&gt;</td>
<td>&lt;middlename&gt;</td>
</tr>
<tr>
<td>&lt;country&gt;</td>
<td>&lt;namedetails&gt;</td>
</tr>
<tr>
<td>&lt;emailaddress&gt;</td>
<td>&lt;organization&gt;</td>
</tr>
<tr>
<td>&lt;emailaddresses&gt;</td>
<td>&lt;organizationinfo&gt;</td>
</tr>
<tr>
<td>&lt;firstname&gt;</td>
<td>&lt;organizationname&gt;</td>
</tr>
<tr>
<td>&lt;generationidentifier&gt;</td>
<td>&lt;organizationnamedetails&gt;</td>
</tr>
</tbody>
</table>

**Example 1: Tagging personal information in DITA**

This example shows a way to tag the following personal name and description.

Mr. Ram V. Kumar Jr.
Chief Technologist
MSI Business Solutions

```
<authorinformation>
  <personinfo>
    <namedetails>
```

DITA Version 1.3 Specification

Copyright © OASIS Open 2005, 2014. All Rights Reserved.
Example 2: Tagging address information in DITA

This example shows a way to tag the following address.

23 Archer St.
Chatsworth
NSW 2067
Australia

Example 3: Tagging complex name and address information in DITA

This example shows two ways to tag a fairly complex collection of personal, organizational, and address information.

Mr. Samuel L. Johnson Jr.
Chief Technologist
c/o XYZ Corporation
52 New Main St.
Carrboro, NC 27510 USA
email: johnson@example.com
phone: 919-555-7987

This method tags all the organizational information as associated with the identified person.
The following method separates the person and organization information. It might be used if it were necessary to associate address information with organizations rather than persons.

3.5.3.4 MathML and equation domains

The MathML and equation domains provide general containers for equations (inline, block, and display) and an integration with the MathML standard. The equation containers identify equations separate from the data that provides the equation content, which might take many forms, including MathML markup, images, and non-XML formats like TeX and ASCII equations.

These two domains provide support for mathematics elements:

MathML domain

Provides the `<mathml>` element, which is a specialization of `<foreign>`. The `<mathml>` element can contain direct MathML markup, or it can use the `<mathmlref>` element to include MathML markup that is stored in a separate, non-DITA document. When this domain is integrated without additional constraints, the `<mathml>` element is allowed wherever `<foreign>` is allowed.
While a `<mathml>` element can only be an equation, the equation domain elements can be used to distinguish equations semantically. While authors can include the `<mathml>` element directly in content, the recommended best practice is to always include `<mathml>` within one of the equation domain elements or their semantic equivalent in a non-OASIS-defined DITA vocabulary.

Comment by Kristen Eberlein, 26 January 2014
What does the first sentence mean?

Equation domain

Provides the `<equation-inline>`, `<equation-block>`, and `<equation-figure>` elements. This domain can be used independently of the MathML domain. The elements in the equation domain enable authors to clearly distinguish equations from other type of content; these markup distinctions can enable formatting distinctions, numbering of equations, and more.

3.5.3.5 Release management domain

The release management domain provides markup that enables content developers to log information about changes that have been made to a DITA topic or map. This information can be used to generate release notes, which can help users locate significant changes in revisions of large documents.

Processors can gather the information and assemble it into an appendix or separate document, thus eliminating the need for separate release-note topics or external tracking mechanisms such as spread sheets or databases. Regulated industries can use the release management domain to provide the records that are required by regulatory agencies.

This figure shows the structure of the release management elements.

```
Release management  topic  prolog  change-historylist  change-item*
```

Figure 36: Release management elements

The following list provides a brief description of the element:

- `<change-historylist>`
  Contains `<change-item>` elements, each of which represents a release note, which is information about a single change
- `<change-item>`
  Contains information about a release note
- `<change-person>`
  Contains the name of a person who made the change
- `<change-organization>`
  Contains the name of the organization that requested the change
- `<change-revisionid>`
  Contains an identifier that is associated with the change
<change-request-reference>
Contains information that provides traceability to an external change request or other ticketing system
</change-request-reference>

<change-request-system>
Contains the name of the external system that is referenced
</change-request-system>

<change-request-id>
Contains an identifier that is associated with the change request, such as an issue ID or ticket number
</change-request-id>

<change-started>
Contains a string that holds date and time information about when the change was started
</change-started>

<change-completed>
Contains a string that holds date and time information about when the change was finished
</change-completed>

<change-summary>
Contains a descriptive summary of the change
</change-summary>

The following DITA modules are provided for the release management domain:

releaseManagementDomain.mod, releaseManagementDomain.ent
releaseManagementDomainMod.xsd, releaseManagementDomainGrp.xsd

Comment by Kristen Eberlein , 7 March 2014
Need names of RELAX NG files

This figure shows three simple change-items added to a single topic. This topic is used in documentation for two products, A and B.

<prolog>
<!--...-->
<changehistory-list>
<change-item product="productA productB">
 <change-person>Joan Smith</change-person>
 <change-completed>2013-03-23</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>Bill Carter</change-person>
 <change-completed>2013-06-07</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>Richard Smith</change-person>
 <change-completed>2013-07-20</change-completed>
 <change-summary>Made change 3 to both products</change-summary>
 <data>Details of change 3</data>
</change-item>
</changehistory-list>
<!--...-->
</prolog>

Figure 37: Excerpt from topic prolog

Comment by Kristen Eberlein , 8 April 2014

From review #1:

Cihak: "I don't see the optional multiple <data> elements that should be children of <change-item>. Should I be concerned? (I can't tell from looking at <data> whether they're in or not.)"

Eberlein: "Hi, Tom, I think you are asking two separate questions here:

• What is in the content model supported by the OASIS grammar file? Let's bring Eliot into this discussion.
Whether the code example needs to be modified to illustrate multiple `<data>` elements.

Eberlein: "E-mailed Tom and Eliot to look at my response. Haven't heard back, so moving the exchange in the DITA source as a draft comment."

### 3.5.4 Troubleshooting information

Troubleshooting information is invaluable to technical content users because it provides advice to correct problems with task performance.

User studies show that users find troubleshooting information invaluable because they frequently access technical information only when they have encountered a problem in performing a task.

The DITA troubleshooting elements added in DITA 1.3 encourage authors to provide troubleshooting information directly within a task, at the point of need, and to label the troubleshooting information appropriately. Minimalism research indicates that users are most likely to read troubleshooting information if it is visibly labeled. We recommend that implementations use stylesheets that add symbols or text to draw attention to the troubleshooting elements in the topics.

Troubleshooting information can be added to an individual step in a DITA task; it also can be added after the steps are completed. Troubleshooting information can be added in any topic type in a `<note>` element with the `@type` attribute set to "trouble". However, we recommend that users use either `steptroubleshooting` or `tasktroubleshooting` when working on a task topic, rather than inserting an `<note>` element with the `@type` attribute set to "trouble".

The examples below illustrate the three options.

#### Step troubleshooting

Troubleshooting information can be added following a step in a procedure if it is likely that the user will encounter a problem in performing the step.

The `<steptroubleshooting>` element can occur following the `<cmd>` element in the `<step>` or `<substep>` element. The `<steptroubleshooting>` element ends the `<step>` or `<substep>` element. Another element, such as `<info>` or `<stepxmp>`, cannot be added after the `<steptroubleshooting>` element.

The first example shows troubleshooting information added to a step. The second example shows the troubleshooting information following a step result.

```xml
<step>
  <cmd>Selected <uicontrol>Shut Down</uicontrol> from the <uicontrol>File</uicontrol> Menu.</cmd>
  <steptroubleshooting> 
    If a problem with the computer prevents you from choosing Shut Down—for example, if the computer “freezes” so that the pointer does not respond to the trackpad—you can turn off the computer by holding down the Control, Option, Command, and Power On keys at the same time.
  </steptroubleshooting>
</step>

<step>
  <cmd>Selected the element for which you want to assign a conditional-processing attribute, and, in the <wintitle>Attributes</wintitle> window, select the conditional-processing attribute.</cmd>
  <stepresult>The permissible values for the attribute are displayed in a drop-down list.</stepresult>
  <steptroubleshooting> 
    If the list of controlled values is not displayed, ensure that the root map is opened in the <wintitle>DITA Maps Manager</wintitle> window.
  </steptroubleshooting>
</step>
```
Task troubleshooting

Troubleshooting information can be added after the procedural component to assist the user in correcting a problem that might have occurred. The <tasktroubleshooting> element is one of four optional elements that can follow the <steps> element. When these optional element are used, they must appear in the following order: <results>, <tasktroubleshooting>, <example>, and <postreq>.

<steps>
<step>
<cmd>When the fast blinking stops, press <uicontrol>small espresso</uicontrol>.</cmd>
</step>
</steps>
<tasktroubleshooting>If the <uicontrol>small espresso</uicontrol> button is not lit, recycle the unit by turning the external <uicontrol>Power</uicontrol> off and on.
</tasktroubleshooting>

Troubleshooting note

A <note> element can include type="troub" to indicate that the note deals with a potential problem that the user might encounter. We recommend that DITA implementations configure their CSS or stylesheets to include appropriate text or an icon in the generated output.

Best practices for authoring DITA content mandate that the troubleshooting note should not be used in place of <steptroubleshooting> or <tasktroubleshooting> in a task topic.

<concept>
<p>If you expose your camera to sudden changes in temperature or humidity, you may experience some condensation in the camera. You should avoid the possibility of condensation because it may result in soil on the lens or the monitor, cause mold, or damage the camera.</p>
<note type="trouble">If you do get condensation, turn off the camera and wait about two hours before using it. Once the camera adjusts to the surrounding temperature, the fogging will clear naturally.</note>
</concept>

3.6 Learning and training specializations

3.6.1 Overview

The DITA Learning and Training specialization addresses several key problems facing developers and consumers of instructional content.

Today’s learners confront a complex world with many inter-related bits and pieces of information, many different ways to access that information, and a strong need to identify the connection points, the objectives, and the context for what to know and what to learn.

In this environment, developers of learning and training content face many challenges, including:

• How to find the context for developing and delivering the right content to the right person at the right time?
• How to identify the learning goals and objectives?
• Who and how many are the audiences?
• How to pull together and integrate content from many different sources and content providers?
• How to enable customers and partners to add, integrate, assemble, and deliver their own content?

These key challenges and issues for delivery of learning and training content mirror long-standing pain points and requirements for content delivery in general.

• Content consumers value consistency of content and learning experiences.
• They desire management of content to make it shareable across and within teams.
• They seek to simplify the information needed to support the complex environments.
• Finally, they want a content development process that can enable the assembly and delivery of custom content that addresses specific learning contexts and use cases.

3.6.2 Objectives of the DITA Learning and Training Specialization

The DITA Learning and Training specialization builds on best practices for modular content design, following DITA principles.

The objectives of the specialization include the following:

1. Provide a general top-level design for authoring of education content with good learning architecture, following DITA principles and best practices.

Some specifics of good DITA design for learning content include:

1. offers a starter set of specialized topic types that support structured, intent-based authoring of content for learning and training, including assessments
2. provides a map domain for structuring and storing the specialized learning topics as reusable learning objects, and for managing the linking and relationships among them
3. offers basic map-driven processing to support topic linking, relationships, and simple sequencing
4. includes a starter set of commonly-used learning interactions, for use in testing and assessment
5. provides support for learning metadata based on the IEEE standard for learning objects metadata (LOM), for use in both topics and maps

2. Establish guidelines that promote best practices for applying standard DITA approaches to learning content, which include:

   a. separation of presentation and content (as much as possible)
   b. separation of content and context
   c. single sourcing, repurposing, and reuse

3. Provide basic support for processing DITA content for delivery as learning and training in a variety of forms, including print and presentation delivery to support instructor-led training (ILT) and web delivery for distance learning.

4. Provide a framework for developing targeted support for processing DITA learning content for delivery with standards-based learning, specifically targeting SCORM. Extend DITA processing to support basic SCORM packaging and required SCORM LMS runtime behaviors. Build on best practices for behaviors to drive and present the interactions.

5. Build on existing DITA infrastructures as much as possible, so learning content developers do not have to start from scratch because with minimal adaptation they can use standard approaches for DITA content and reuse content previously developed for other purposes.

Note: Simply using the content models described in this specification, of course, does not ensure quality learning content. Quality learning content only results from good instructional design and in-depth learning needs analysis.

3.6.3 A learning objects approach to learning and training content

The DITA Learning and Training specialization applies DITA principles and best practices for using topic-based and modular content to plan, develop, and deliver learning and training content.

The reusable learning objects, or RLO, approach to learning content derives from the pioneering work of learning content designers at several companies, including Autodesk®, Oracle®, and Cisco®. Author Peder Jacobsen defines an RLO as "a discrete reusable collection of content used to present and support a single learning objective." With this approach, it is possible to gather a pool of information objects and make them available for reuse and repurposing in a variety of learning delivery contexts.

There is a strong affinity between the DITA topic-based, modular approach to content in general, and the learning objects approach to learning content in particular.
Working assumptions about learning content and how to support authoring and delivering it with DITA include the following:

- The DITA Learning and Training specialization builds on a reusable learning objects (RLO) approach to learning content.
- **DITA topic types** are the basic building blocks for learning objects and specify the meaning and intention of content provided in instructional and information objects.
- **DITA domains** provide the mechanism for defining *interactions*, which can be used across the learning topic types.
- **DITA domains** also provide the mechanism for defining learning *metadata*, which can be assigned either in topics or in maps.
- **DITA maps** arrange the DITA learning topics into a hierarchy of learning objects and group such content for delivery as lessons, modules, and courses.
- **DITA specialization** provides the mechanism for creating the learning-based topic types, domains, and maps needed for instructional and information object content requirements.

This figure shows the composition of learning objects as a) instructional objects, b) information objects, and c) the specialized DITA topic types to support them.

**Opportunity: Learning objects and specialized DITA learning types**

In this approach, a learning object comprises a "discrete reusable collection of content used to present and support a single learning objective," and consists of two primary information components:

- **Instructional objects**, which provide the structured framework for a learning experience. The `<learningOverview>`, `<learningSummary>`, and `<learningAssessment>` topic types provide content for instructional objects.
- **Information objects**, which provide the source learning content - the topic-based learning content and other supplemental content that supports the learning goals identified in the instructional objects. The `<learningContent>` topic type provides content for information objects.
- **Instructional plans**, which identify the learning goals, needs, and objectives. The `<learningPlan>` topic type provides content for instructional plans.

**Figure 38: Learning objects and specialized DITA learning and training topic types**

This picture shows the end-to-end process for designing, authoring, and delivering specialized learning content with DITA.
In this approach, a learning content developer:

1. Uses **learning map elements** to identify the learning objects and the supporting content needed to address specific learning goals and objectives.

2. Uses **learning topic elements** to structure the learning content.

3. Applies **learning metadata elements** to describe specific characteristics of the learning content, following a sub-set of the IEEE LOM standard.

4. Constructs **specific build maps and relationship tables** to organize learning objects for delivery as a course with specific output and delivery needs.

5. Invokes **processing** to generate specific learning deliverables, based on the default processing available with DITA content and specialized as needed for learning-specific purposes and delivery formats.

**Figure 39: Learning content design, authoring, and delivery through DITA specialization**

### 3.6.4 Use cases

Several use cases inform the design and development of the DITA Learning and Training Specialization.

**Enable indexing, searching, and retrieval of learning content**

By structuring content with DITA topics and maps as self-contained learning objects matched with appropriate DITA metadata, it is possible to enable fast indexing, search, and retrieval of learning content that meets specific learning goals and objectives.

**Creating custom courses quickly**

A company has a large inventory of topic-based content that is used to provide technical and troubleshooting information about a set of componentized software products. It desires to enable field engineers to quickly identify technical content that is suitable for providing on-site training. The DITA Learning and Training Specialization enables field engineers to draw on their inventory of topics and quickly assemble learning content to meet specific customer needs.

**Making technical content available for direct sharing and reuse in learning and training**

A DITA learning specialization makes it possible to define a context for and directly assemble and use existing technical content for delivery as learning and training. The DITA approach identifies consistent structures and patterns and leverages them to enable a consistent approach for sharing content across teams. The result is much more opportunity to share content between different providers and across areas of expertise, to learn from each other, and to deliver content and the learning experience consistently. As a
result, instead of copy, paste, and make unique as the norm, we have write once and share with others as the new norm.

Related Links

"Recycling Knowledge with Learning Objects," by Ruth Clark.
"DITA in Education," by Dan Schneider, EduTech Wiki.

3.6.5 Summary of learning topic, map, and domain designs

The DITA Learning and Training specialization provides a set of specialized DITA topics, a learning interactions domain, a learning metadata domain, and a learning map domain to support creating and delivering structured learning content.

Learning topic types

The following specialized DITA topic types provide support for creating learning and training content.

Learning Plan topic type
Describes learning needs and goals, instructional design models, task analyses, learning taxonomies, and other information necessary to the lesson planning process.

Learning Overview topic type
Identifies the learning objectives and includes other information helpful to the learner, such prerequisites, duration, and intended audience.

Learning Content topic type
Provides the learning content itself and enables direct use of content from DITA task, concept, and reference topics, as well as additional content of any topic type that supports specific objectives declared in the Learning Overview topic type.

Learning Summary topic type
Recaps and provides context for the learning objectives and provides guidance to reinforce learning and long-term memory.

Learning Assessment topic type
Presents instruments that measure progress, encourage retrieval, and stimulate reinforcement of the learning content and can be presented before the content as a pre-assessment or after the content as a post-assessment checkpoint or test.

Learning map types

<learningGroupMap>
A map container to enable <learningGroup> elements to be stored as separate objects and referenced by maps.
<learningObjectMap>
A map container to enable <learningObject> elements to be stored as separate objects and referenced by maps.

Learning map domain

Use the learning map domain to organize groups of topics as learning objects.

Note: The learning map domain is part of the <learningMap> and the <learningBookmap> document types. As these learning map structures are delivered as a domain specialization rather than as a structural specialization, it is possible to extend any type of DITA map to include these structures.

<learningGroup>
A map container and optional topic reference to introduce and group learning objects into higher-level organizations, such as course-level, module-level, or lesson-level. A `<learningGroup>` can contain other `<learningGroup>` elements, allowing you to organize learning content at course, module, or other higher levels of hierarchy.

`<learningObject>`
A map container and optional topic reference to introduce and group the topic references for a learning object.

`<learningPlanRef>`
A topic reference to a learning plan or other topic that provides the learning plan.

`<learningOverviewRef>`
A topic reference to a learning overview or other topic that introduces the learning object.

`<learningContentRef>`
A topic reference to a learning content topic, or a topic, task, concept, reference or other specialized topic.

`<learningContentComponentRef>`
A topic reference to a learning content topic, or a topic, task, concept, reference or other specialized topic.

`<learningSummaryRef>`
A topic reference to a learning summary or other topic that provides the summary.

`<learningPreAssessmentRef>`
A topic reference to a learning assessment or other topic that is used as a pre-assessment.

`<learningPostAssessmentRef>`
A topic reference to a learning assessment or other topic that is used as a post-assessment.

`<learningObjectMapRef>`
A topic reference to a learning object.

`<learningGroupMapRef>`
A topic reference to a learning group map

**Learning interactions domain**

The learning interactions domain defines a set of basic learning interaction elements as a DITA domain. Although there are two sets of elements for interaction support, the recommended set is defined in the learningDomain2 support provided in DITA 1.3, which supports block elements. This support deprecates original DITA 1.2 elements, which failed to support including block elements where needed. See [learningDomain1 and learningDomain2 Support](see page 221).

`<lcOpenQuestion2>`
Poses an open-ended question in an assessment interaction.

`<lcTrueFalse2>`
Presents the learner with two choices, one correct, the other incorrect, often presented as true/false or yes/no responses.

`<lcSingleSelect2>`
Presents three or more choices, only one of which is correct.

`<lcMultipleSelect2>`
Presents two or more choices, two or more of which are correct.

`<lcMatching2>`
In a list of paired choices, the learner identifies the correct choice that matches another choice.

`<lcHotspot2>`
Presents an image, and the learner clicks on one or more regions to indicate a choice.

`<lcSequencing2>`
Presents choices in a list that the learner must arrange in a correct ordered sequence.

**Learning metadata domain**

The learning metadata domain defines a set of basic learning metadata elements as a DITA domain and available for use in the learning topic types, as specialized `<prolog>` metadata and in the learning map domain, as specialized `<topicmeta>`.

`<lcLom>` makes the learning metadata elements available in the learning topics and learning map domain.

Elements in `<lcLom>` include:

- `<lomAggregationLevel>`
- `<lomContext>`
- `<lomCoverage>`
- `<lomDifficulty>`
- `<lomInstallationRemarks>`
- `<lomIntendedUserRole>`
- `<lomInteractivityLevel>`
- `<lomInteractivityType>`
- `<lomLearningResourceType>`
- `<lomOtherPlatformRequirements>`
- `<lomSemanticDensity>`
- `<lomStructure>`
- `<lomTechRequirement>`
- `<lomTypicalAgeRange>`
- `<lomTypicalLearningTime>`

**Instructor notes**

The learning interactions domain also makes available an `<lcInstructornote>` element for providing instructor-specific information.

### 3.6.6 learningDomain1 and learningDomain2 Support

Many questions require multiple block elements for the question statement ("prompt" or "stimulus" in instructional design parlance) and for feedback. The DITA 1.2 interaction, learningDomain1, design only allows for a single paragraph question and single-paragraph feedback. The learningDomain2 design uses the `<div>` element as the base type for interaction elements to provide block element support in question statements, answer content and feedback.

The DITA 1.3 learningDomain2 support deprecates the DITA 1.2 learningDomain1, but enables backward compatibility by redefining the `<learningAssessment>` topic with the option of using either the DITA 1.2 support or the DITA 1.3 support or both.

**learningDomain2**

DITA 1.3 defines two new domains for the Learning and Training vocabulary: interactionBase2Domain and learning2Domain. These two new domains mirror the existing interactionBase and learning domains, but provide a base model that allows multi-paragraph questions and multi-paragraph feedback. The new domains are otherwise semantically identical to the original domains.

**learningInteractionBase2Domain**

The `<lcInteractionBase2>` element differs from `<lcInteractionBase>` as follows:
• Is a specialization of <div> rather than <fig>. It removes the content limitations that were imposed by <fig>, such as the inability to include <table>. The 1.3 specialization allows all content allowed in <div>.
• Uses <lcQuestionBase2> in place of <lcQuestionBase> to allow multiple paragraphs within its content.
• Makes @id attribute optional for all interactions.

<lcQuestionBase2>
The <lcQuestionBase2> element is also a specialization of <div> and allows the same content as <div>. <div> allows all content that occurs in <body> except <section> and <bodydiv>.

<lcInteractionLabel2>
The <lcInteractionLabel2> is a specialization of <p> and replaces the use of <title>. This element is necessary because <div> does not allow <title>.

learning2Domain
The learning2Domain is closely modeled on the 1.2 learningDomain. It defines new versions of each of the interaction types, each of which are specialized from <lcInteractionBase2>. The learning2Domain is not dependent on the original learningDomain. The learning2Domain can be used by itself or integrated along with the learningDomain.

3.6.7 Learning and Training specialization usage
The DITA 1.3 learning and training support includes new domains that deprecate the DITA 1.2 interactions, but provide a path for backward compatibility.

The key changes in the DITA 1.3 support are:
• Support block elements in question content, answer content, and feedback.
• Allow multiple assets.
• Make the @id attribute for interactions optional.

New implementations
If you are implementing learning and training support in DITA for the first time, use the learningDomain2 domain for document types that need assessments. This domain is integrated into the learningAssessment-learning2 document-type shell.

Existing implementations to continue DITA 1.2 assessment support
If you have already implemented learning and training support with DITA 1.2 and none of the limitations adversely affected your support, then you can continue to use the learningDomain in document types that need assessments, as demonstrated by the learningAssessment.dtd and the learningAssessment-learning2.dtd document type shells.
4 Language reference: Learning and training

Elements in the Learning and Training section include specialized topic types designed for learning and training content, as well as specialized domain elements for organizing learning content within maps, specifying learning metadata, and describing learning interactions.

4.1 Element quick reference

4.1.1 Base DITA elements, A to Z

This section provides links to all of the base DITA elements in alphabetical order.

abstract (see page 254)
alt (see page 258)
anchor (see page 305)
anchorid (see page 346)
anchorkey (see page 350)
anchorref (see page 313)
area (see page 360)
attributedef (see page 384)
audience (see page 321)
author (see page 322)
b (see page 357)
body (see page 255)
bodydiv (see page 256)
boolean (see page 406)
brand (see page 323)
category (see page 323)
cite (see page 258)
colspec (see page 292)
component (see page 324)
consequence (see page 353)
coords (see page 360)
copyrholder (see page 325)
copyright (see page 324)
copyryear (see page 325)
created (see page 326)
critdates (see page 326)
data-about (see page 409)
data (see page 407)
dd (see page 258)
ddhd (see page 259)
defaultSubject (see page 385)
desc (see page 259)
dita (see page 257)
ditavalmeta (see page 367)
ditavalref (see page 366)
div (see page 260)
dl (see page 261)
liq (see page 273)
map (see page 302)
mapref (see page 315)
messagepanel (see page 356)
metadata (see page 328)
navref (see page 306)
navtitle (see page 250)
no-topic-nesting (see page 411)
note (see page 274)
object (see page 274)
of (see page 278)
othermeta (see page 328)
overline (see page 359)
p (see page 278)
param (see page 279)
permissions (see page 329)
ph (see page 280)
platform (see page 330)
pre (see page 280)
prodinfo (see page 330)
prodname (see page 330)
prognum (see page 331)
prolog (see page 320)
publisher (see page 331)
q (see page 281)
related-links (see page 256)
relatedSubjects (see page 388)
relcell (see page 309)
relcolspec (see page 309)
relheader (see page 309)
relrow (see page 308)
reltable (see page 306)
required-cleanup (see page 413)
resourceid (see page 331)
revised (see page 333)
row (see page 293)
schemeref (see page 376)
searchtitle (see page 249)
section (see page 281)
sectiondiv (see page 282)
series (see page 333)
shape (see page 362)
shortdesc (see page 251)
simpletable (see page 295)
si (see page 282)
sli (see page 283)
sort-as (see page 363)
source (see page 334)
state (see page 412)
stentry (see page 297)
4.1.2 Technical content elements, A to Z

This section provides an alphabetized list of links to all elements in the technical content subject area.

abbreviated-form (see page 457)
abbreviated-form (see page 457)
*abbrevlist* (see page 462)
*addressdetails* (see page 537)
*administrativearea* (see page 537)
*amendments* (see page 462)
*apiname* (see page 495)
*appendices* (see page 463)
*appendix* (see page 463)
*approved* (see page 476)
*authorinformation* (see page 536)
*backmatter* (see page 464)
*bibliolist* (see page 464)
*bookabstract* (see page 465)
*bookchangehistory* (see page 477)
*bookevent* (see page 477)
*bookeventtype* (see page 478)
*bookid* (see page 478)
*booklibrary* (see page 465)
*booklist* (see page 466)
*booklists* (see page 466)
*bookmap* (see page 461)
*bookmeta* (see page 475)
*booknumber* (see page 478)
*bookowner* (see page 479)
*bookpartno* (see page 479)
*bookrestriction* (see page 480)
*bookrights* (see page 480)
*booktitle* (see page 467)
*booktitlealt* (see page 467)
*cause* (see page 446)
*change-completed* (see page 509)
*change-historylist* (see page 509)
*change-item* (see page 510)
*change-organization* (see page 510)
*change-person* (see page 511)
*change-request-id* (see page 511)
*change-request-reference* (see page 511)
*change-request-system* (see page 512)
*change-revisionid* (see page 512)
*change-started* (see page 512)
*change-summary* (see page 513)
*chapter* (see page 467)
*chdesc* (see page 433)
*chdeschd* (see page 431)
*chhead* (see page 430)
*choice* (see page 434)
*choices* (see page 434)
*choicetable* (see page 430)
*choption* (see page 433)
*choptionhd* (see page 431)
*chrow* (see page 432)
closereqs (see page 521)
cmd (see page 428)
cmdname (see page 514)
codeblock (see page 496)
codeph (see page 497)
coderef (see page 496)
colophon (see page 468)
completed (see page 480)
conbody (see page 422)
conbodydiv (see page 422)
concept (see page 421)
condition (see page 445)
contactnumber (see page 538)
contactnumbers (see page 538)
context (see page 425)
copyrfirst (see page 481)
copyrlast (see page 481)
country (see page 539)
day (see page 481)
dedication (see page 468)
delim (see page 507)
draftintro (see page 469)
edited (see page 482)
edition (see page 482)
emailaddress (see page 539)
emailaddresses (see page 540)
equation-block (see page 489)
equation-figure (see page 490)
equation-inline (see page 491)
esttime (see page 524)
figurelist (see page 469)
filepath (see page 515)
firstname (see page 540)
fragment (see page 502)
fragref (see page 503)
frontmatter (see page 470)
generationidentifier (see page 541)
glossAbbreviation (see page 449)
glossAcronym (see page 449)
glossAlt (see page 450)
glossAlternateFor (see page 450)
glossBody (see page 451)
glossPartOfSpeech (see page 451)
glossProperty (see page 451)
glossScopeNote (see page 452)
glossShortForm (see page 453)
glossStatus (see page 453)
glossSurfaceForm (see page 454)
glossSymbol (see page 454)
glossSynonym (see page 456)
perskill (see page 524)
person (see page 484)
personinfo (see page 546)
personname (see page 546)
personnel (see page 523)
plentry (see page 498)
postalcode (see page 547)
postreq (see page 437)
preface (see page 473)
prefreqs (see page 520)
prereq (see page 424)
printlocation (see page 484)
propdesc (see page 443)
propdescd (see page 442)
properties (see page 440)
property (see page 442)
prophead (see page 441)
proptype (see page 443)
proptypehd (see page 441)
propvalue (see page 443)
propvaluehd (see page 442)
pt (see page 499)
published (see page 485)
publisherinformation (see page 485)
publishtype (see page 485)
refbody (see page 439)
refbodydiv (see page 439)
reference (see page 438)
refsyn (see page 440)
remedy (see page 447)
repsep (see page 508)
reqcond (see page 522)
reqconds (see page 521)
reqcontp (see page 522)
reppers (see page 523)
responsibleParty (see page 447)
result (see page 437)
reviewed (see page 486)
revisionid (see page 486)
safecond (see page 529)
safety (see page 530)
screen (see page 532)
sep (see page 508)
shortcut (see page 532)
spare (see page 528)
spares (see page 528)
sparesli (see page 528)
started (see page 486)
step (see page 427)
stepresult (see page 435)
4.1.3 Learning and training elements, A to Z

This topic provides an alphabetical listing of all elements in the Learning and Training group.

- `lcAge` (see page 618)
- `lcAnswerContent` (see page 605)
- `lcAnswerContent2` (see page 569)
- `lcAnswerOption` (see page 605)
- `lcAnswerOption2` (see page 570)
- `lcAnswerOptionGroup` (see page 606)
- `lcAnswerOptionGroup2` (see page 570)
- `lcAreaCoords` (see page 607)
- `lcAreaCoords2` (see page 572)
- `lcArea` (see page 606)
- `lcArea2` (see page 571)
- `lcAreaShape` (see page 608)
- `lcAreaShape2` (see page 573)
- `lcAssessment` (see page 618)
- `lcAsset` (see page 609)
- `lcAttitude` (see page 619)
- `lcAudience` (see page 619)
- `lcBackground` (see page 619)
- `lcCIN` (see page 620)
- `lcChallenge` (see page 620)
- `lcClassroom` (see page 620)
- `lcClient` (see page 621)
- `lcConstraints` (see page 621)
- `lcCorrectResponse` (see page 609)
- `lcCorrectResponse2` (see page 574)
- `lcDelivDate` (see page 622)
- `lcDelivery` (see page 622)
- `lcDownloadTime` (see page 622)
- `lcDuration` (see page 623)
- `lcEdLevel` (see page 623)
- `lcFeedback` (see page 610)
- `lcFeedback2` (see page 575)
- `lcFeedbackCorrect` (see page 610)
- `lcFeedbackCorrect2` (see page 576)
- `lcFeedbackIncorrect` (see page 610)
- `lcFeedbackIncorrect2` (see page 576)
- `lcFileSizeLimitations` (see page 623)
- `lcGapAnalysis` (see page 624)
- `lcGapItem` (see page 624)
- `lcGapItemDelta` (see page 625)
- `lcGeneralDescription` (see page 625)
- `lcGoals` (see page 625)
- `lcGraphics` (see page 626)
- `lcHandouts` (see page 626)
- `lcHotspot` (see page 595)
- `lcHotspot2` (see page 577)
- `lcHotspotMap` (see page 611)
lcPlanPrereqs (see page 639)
lcPlanResources (see page 640)
lcPlanSubject (see page 640)
lcPlanTitle (see page 640)
lcPlayers (see page 641)
lcPrereqs (see page 641)
lcProcesses (see page 642)
lcProject (see page 642)
lcQuestion (see page 615)
lcQuestion2 (see page 584)
lcQuestionBase (see page 590)
lcQuestionBase2 (see page 584)
lcResolution (see page 643)
lcResources (see page 643)
lcReview (see page 644)
lcSecurity (see page 644)
lcSequence (see page 616)
lcSequence2 (see page 584)
lcSequenceOption (see page 616)
lcSequenceOption2 (see page 585)
lcSequenceOptionGroup (see page 617)
lcSequenceOptionGroup2 (see page 586)
lcSequencing (see page 593)
lcSequencing2 (see page 587)
lcSingleSelect (see page 591)
lcSingleSelect2 (see page 587)
lcSkills (see page 645)
lcSpecChars (see page 645)
lcSummary (see page 646)
lcTask (see page 646)
lcTaskItem (see page 647)
lcTechnical (see page 647)
lcTime (see page 648)
lcTrueFalse (see page 591)
lcTrueFalse2 (see page 588)
lcValues (see page 648)
lcViewers (see page 649)
lcW3C (see page 649)
lcWorkEnv (see page 650)
lcWorkEnvDescription (see page 650)
learningAssessment (see page 552)
learningAssessmentbody (see page 553)
learningBase (see page 556)
learningBasebody (see page 557)
learningContent (see page 550)
learningContentComponentRef (see page 567)
learningContentRef (see page 566)
learningContentbody (see page 551)
learningGroup (see page 559)
learningGroupMap (see page 557)
4.1.4 All DITA elements, A to Z

This section provides links to all of the DITA elements in alphabetical order.

abbreviated-form (see page 457)
abbrevlist (see page 462)
abstract (see page 254)
addressdetails (see page 537)
administrativearea (see page 537)
alt (see page 258)
amendments (see page 462)
anchor (see page 305)
anchorid (see page 346)
anchorkey (see page 350)
anchorref (see page 313)
apiname (see page 495)
appendices (see page 463)
appendix (see page 463)
approved (see page 476)
area (see page 360)
chrow (see page 432)
cite (see page 258)
closereqs (see page 521)
cmd (see page 428)
cmdname (see page 514)
codeblock (see page 496)
codeph (see page 497)
coderef (see page 496)
colophon (see page 468)
colspec (see page 292)
completed (see page 480)
component (see page 324)
conbody (see page 422)
conbodydiv (see page 422)
concept (see page 421)
condition (see page 445)
consequence (see page 353)
contactnumber (see page 538)
contactnumbers (see page 538)
context (see page 425)
coords (see page 360)
copyrfirst (see page 481)
copyrholder (see page 325)
copyright (see page 324)
copyrlast (see page 481)
copyryear (see page 325)
country (see page 539)
created (see page 326)
critdates (see page 326)
data-about (see page 409)
data (see page 407)
day (see page 481)
dx (see page 258)
dd (see page 259)
ddhd (see page 259)
dedication (see page 468)
defaultSubject (see page 385)
delim (see page 507)
desc (see page 259)
dita (see page 257)
ditavalmeta (see page 367)
ditavalref (see page 366)
div (see page 260)
dl (see page 261)
dlentry (see page 262)
dlhead (see page 263)
draft-comment (see page 264)
draftintro (see page 469)
dl (see page 263)
dthd (see page 265)
drvKeyscopePrefix (see page 369)
dvrKeyscopeSuffix (see page 370)
dvrResourcePrefix (see page 368)
dvrResourceSuffix (see page 369)
edited (see page 482)
edition (see page 482)
<elementdef (see page 383)
emailaddress (see page 539)
emailaddresses (see page 540)
entry (see page 293)
enumerationdef (see page 382)
equation-block (see page 489)
equation-figure (see page 490)
equation-inline (see page 491)
esttime (see page 524)
example (see page 265)
exportanchors (see page 343)
featnum (see page 327)
fig (see page 266)
figgroup (see page 266)
figurelist (see page 469)
filepath (see page 515)
firstname (see page 540)
fn (see page 267)
foreign (see page 410)
fragment (see page 502)
fragref (see page 503)
frontmatter (see page 470)
generationidentifier (see page 541)
glossAbbreviation (see page 449)
glossAcronym (see page 449)
glossAlt (see page 450)
glossAlternateFor (see page 450)
glossBody (see page 451)
glossPartOfSpeech (see page 451)
glossProperty (see page 451)
glossScopeNote (see page 452)
glossShortForm (see page 453)
glossStatus (see page 453)
glossSurfaceForm (see page 454)
glossSymbol (see page 454)
glossSynonym (see page 456)
glossUsage (see page 456)
glossarylist (see page 470)
glossdef (see page 449)
glossentry (see page 447)
glossgroup (see page 460)
glossref (see page 458)
glossterm (see page 448)
groupchoice (see page 501)
groupcomp (see page 501)
groupseq (see page 500)
hasInstance (see page 378)
hasKind (see page 379)
hasNarrower (see page 379)
hasPart (see page 380)
hasRelated (see page 381)
hazardstatement (see page 352)
hazardsymbol (see page 354)
honorific (see page 541)
howtoavoid (see page 355)
i (see page 357)
image (see page 268)
imagemap (see page 361)
index-base (see page 411)
index-see-also (see page 341)
index-see (see page 340)
index-sort-as (see page 342)
indexlist (see page 471)
indexterm (see page 336)
indextermref (see page 340)
info (see page 428)
isbn (see page 482)
itemgroup (see page 411)
keydef (see page 315)
keyword (see page 270)
keywords (see page 327)
kwd (see page 505)
lastname (see page 541)
lcAge (see page 618)
lcAnswerContent (see page 605)
lcAnswerContent2 (see page 569)
lcAnswerOption (see page 605)
lcAnswerOption2 (see page 570)
lcAnswerOptionGroup (see page 606)
lcAnswerOptionGroup2 (see page 570)
lcAreaCoords (see page 607)
lcAreaCoords2 (see page 572)
lcArea (see page 606)
lcArea2 (see page 571)
lcAreaShape (see page 608)
lcAreaShape2 (see page 573)
lcAssessment (see page 618)
lcAsset (see page 609)
lcAttitude (see page 619)
lcAudience (see page 619)
lcBackground (see page 619)
lcCIN (see page 620)
lcChallenge (see page 620)
lcClassroom (see page 620)
lcClient (see page 621)
4.2 Topic elements

The base topic elements include elements that make up the core building blocks of the DITA topic, such as topic, body, and related-links, as well as elements like `<p>` and `<ph>` that are used in many topic specializations. Some of these elements are also available inside the `<topicmeta>` map element.

4.2.1 Basic topic elements

The generic topic structure is used for untyped topics. While much of the DITA architecture is built on generic topics, it is generally better to use more specific information types (such as concept, task, or reference) when they are available.

For an answer to the question "What are topics?" and more details on when to use different information types, please refer to DITA topics (see page 24).
4.2.1.1 <topic>

The <topic> element is the top-level DITA element for a single-subject topic or article. Other top-level DITA elements that are more content-specific are <concept>, <task>, <reference>, and <glossentry>, all of which are specializations of the <topic> element.

Inheritance
- topic/topic

Example

```xml
<topic id="topic">
  <title>Some little topic</title>
  <body>
    <p>Here's a little topic.</p>
    <ul>
      <li>Some item</li>
      <li>Another item</li>
    </ul>
  </body>
</topic>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of @id, given below), *Architectural attribute group* (see page 654), and *outputclass* (see page 0).

@id (REQUIRED)

An anchor point. This ID is usually required as part of the @href or @conref syntax when cross referencing or reusing content within the topic; it also enables <topicref> elements in DITA maps to optionally reference a specific topic within a DITA document. This attribute is defined with the XML Data Type ID.

4.2.1.2 <title>

The <title> element contains a heading or label for the main parts of a topic: the topic as a whole, sections, examples, and labelled content such as figures and tables. The element also can be used to provide a title for a map or a relationship table; when used in a relationship table, the title typically is used as a authoring convenience and is not rendered for display to an end user.

Inheritance
- topic/title

Example

```xml
<title id="topicid">Monitoring the patient following the procedure</title>
</title>
<body>

/*!-- ... -->
  <fig id="fig-01">
    <title>Patent-monitoring procedure flow following procedures</title>
    <!-- ... -->
  </fig>

/*!-- ... -->
</body>
```
**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) (without the Metadata attribute group), `@base` and `@rev` from the *Metadata attribute group* (see page 651), and `outputclass` (see page 0).

**Comment by robander**

If somebody else would like to rephrase the above, feel free. We have Univ-atts minus metadata, plus `@base` and `@rev` from metadata. So could also say "ID, Debug, Localiation, @class, @base and @rev from Metadata, and @outputclass."

4.2.1.3 `<titlealts>`

The `<titlealts>` element allows the insertion of alternate titles, such as titles that should be used in creating a table of contents for navigation or a title specific to search results. When the `<titlealts>` element is absent, the `<title>` element is used for all title purposes.

**Inheritance**

- topic/titlealts

**Example**

```xml
<task id="progexample">
  <title>Example of Required Programming</title>
  <titlealts>
    <navtitle>Programming Example</navtitle>
  </titlealts>
  <taskbody> . . . </taskbody>
</task>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650).

**Comment by robander**

Proposal 13114 added `@rev` here, but it was already present in DITA 1.2 (it is part of univ-atts). I've removed the re-declaration.

4.2.1.4 `<searchtitle>`

The `<searchtitle>` element is used to specify a title that should be displayed by search tools that locate the topic. This is most useful when the topic has a title that makes sense in the context of a single information set, but may be too general in a list of search results; for example, a topic title of "Markup example" may make sense as part of a guide to DITA, but when found among thousands of unrelated topics, a search title of "DITA markup example" is more useful.

When a topic is rendered as XHTML, the contents of the `<searchtitle>` will typically appear in the XHTML's title element, which used in the result summary for many search engines. This element may not be supported for output formats that do not support distinct search titles for topics.

**Inheritance**

"- topic/searchtitle " when used in topics, and "- map/searchtitle " when used in maps."
Example

In the following example, the general title "Programming Example" is likely very useful in a set of information about XSLT basics; however, the same title is not helpful among a set of search results from the entire internet. In that case, "Example of basic programming in XSLT" will be much more helpful.

```
<task id="progexample">
  <title>Programming Example</title>
  <titlealts>
    <searchtitle>Example of basic programming in XSLT</searchtitle>
  </titlealts>
  <taskbody> . . . </taskbody>
</task>
```

When `<searchtitle>` is used in maps, the element provides a new search title for the topic when used in a specific context. For example, the if the following map includes information about programming in many languages, searches among that information set will be most useful when they return "Example of programming in XSLT":

```
<topicref href="progexample.dita">
  <topicmeta>
    <navtitle>Programming example</navtitle>
    <searchtitle>Example of programming in XSLT</searchtitle>
  </topicmeta>
</topicref>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0 ).

Comment by robander

Sent note to TC on Nov 27 2013: @outputclass has never been implemented in our DTD or Schema.

Comment by robander

Proposal 13114 added @rev here, but it was already present in DITA 1.2 (it is part of univ-atts). I've removed the re-declaration.

4.2.1.5 `<navtitle>`

The navigation title `<navtitle>` element is one of a set of alternate titles that can be included inside the `<titlealts>` element. This navigation title may differ from the first level heading that shows in the main browser window. Use `<navtitle>` when the actual title of the topic isn't appropriate for use in a table of contents, navigation pane, or online content (for example, because the actual title is too long). Beginning with DITA 1.2, the `<navtitle>` element is also available in the `<topicmeta>` element in a `<topicref>` in a map, and its use is preferred over the `<topicref>` element's @navtitle attribute.

When `<navtitle>` is used in a map, it functions in the same way as the @navtitle attribute; both are used to specify a navigation title for the target of the `<topicref>` element. That is, the title itself will only be used as an actual navigation title when the title is locked; the title is locked when the `<topicref>` element sets the attribute locktitle="yes". If the title is not locked, processing systems will typically retrieve the current title from the target topic, looking first for a `<navtitle>` element and second for the general `<title>`.

Comment by robander , 20 February 2014

In the previous paragraph, the 1.2 spec stated: "That is, the title itself will only be used as an actual navigation title when the title is locked; the title is locked when the closest ancestor topicref element sets or inherits the attribute locktitle="yes". However, the 1.2 spec itself does not list @locktitle as an inheritable attribute -- it does not cascade. I've changed the text above to indicate that locktitle must be specified on the topicref itself.

When both a `<navtitle>` element and a @navtitle attribute are specified, the `<navtitle>` element SHOULD be used.
Because the `<navtitle>` element is available within `<topicmeta>`, and `<topicmeta>` is used in many different contexts, it is possible that `<navtitle>` can be specified in contexts where a navigation title does not make sense (for example, on the `<topicgroup>` element). In those situations, the `<navtitle>` element has no defined purpose.

**Inheritance**
- topic/navtitle

**Example**

```dita
<task id="progexample">  
  <title>Publishing a DITA information set in PDF</title>  
  <titlealts><navtitle>Publishing in PDF</navtitle></titlealts>  
  <taskbody><context>Some text</context></taskbody>  
</task>
```

**Figure 40: `<navtitle>` sample in a topic**

In this sample, the first title is not locked, and will generally be replaced with a title retrieved from `a.dita`. The second title is locked, and will be displayed when this map is used as a basis for navigation.

```dita
<map xml:lang="en">  
  <title>This is a sample map</title>  
  <topicref href="a.dita">  
    <topicmeta>  
      <navtitle>Title of A</navtitle>  
    </topicmeta>  
  </topicref>  
  <topicref href="b.dita" locktitle="yes">  
    <topicmeta>  
      <navtitle>Short Title for B</navtitle>  
    </topicmeta>  
  </topicref>  
</map>
```

**Figure 41: `<navtitle>` samples in a map**

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650).

**Comment by robander**

Proposal 13114 added `@rev` here, but it was already present in DITA 1.2 (it is part of univ-atts). I've removed the re-declaration.

### 4.2.1.6 `<shortdesc>`

The `<shortdesc>` element is a short description of the topic that occurs between the topic title and the topic body, as the initial paragraph-like content of a topic, or it can be embedded in an `<abstract>` element. The short description, which represents the purpose or theme of the topic, is also intended to be used as a link preview and for searching. The `<shortdesc>` element also can be used in a DITA map.

Use the `<shortdesc>` element when the first paragraph of topic content is simple enough to be suitable for use as a link preview or for summaries. Otherwise use the `<abstract>` element to provide richer content around the `<shortdesc>`. See the *abstract* (see page 254) description for more details on the behavior of `<shortdesc>` in an abstract.

While inclusion of the `<shortdesc>` element is not mandated by DITA or the tools, it is recommended that topics contain this element. In cases where a topic contains only one paragraph, then it is preferable to include this text in the `<shortdesc>` and leave the topic body empty.
The short description should be a single, concise paragraph containing one or two sentences of no more than 50 words.

<table>
<thead>
<tr>
<th>Type</th>
<th>Recommended content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>The short description should explain what the task information helps users accomplish, the benefits of the task, or the purpose of the task. Do not simply repeat the title. Try to include information that will help users understand when the task is appropriate or why the task is necessary. Avoid stating the obvious, such as “You can use XYZ to do A” as the only statement in the short description for Task A. In some cases, add more information about why the task is beneficial. Do not use sentence fragments. Use complete sentences. Avoid starting short descriptions with phrases such as “This topic describes . . . .” or “This topic is about . . . .”.</td>
</tr>
<tr>
<td>Concept</td>
<td>Introduce the concept and provide a concise answer to the question &quot;What is this?&quot; and in some cases &quot;Why do I care about this?&quot; If the concept is unfamiliar, you can start with a brief definition. Avoid using the short description to lead in or build up to a topic. The short description paragraph should contain the main point of the concept topic. The concept short description should clearly apply to a concept. Avoid turning the concept topic into a task. Do not simply repeat the title. Do not use sentence fragments. Use complete sentences. Avoid starting short descriptions with phrases such as “This topic describes . . . .” or “This topic is about . . . .”.</td>
</tr>
<tr>
<td>Reference</td>
<td>Briefly describe what the reference item does, what it is, or what it is used for. In most cases, use a complete sentence. You can use a sentence fragment only for a topic that is very short, such as an API topic and each of its subtopics. Use consistent phrasing across libraries and information centers so that your information can be seamlessly integrated with another product’s information.</td>
</tr>
</tbody>
</table>

**Short descriptions in maps**

The `<shortdesc>` element is also available in maps within the `<topicmeta>` element. In a map, the element specifies that a topic has a short description that is specific to the context of that `<topicref>` in that map. When constructing link previews, links that are generated according to the context of the map should use the `<shortdesc>` content provided in the map rather than the `<shortdesc>` provided in the topic. The `<shortdesc>` element in the map allows authors to provide short descriptions for references to non-DITA objects.

The content of the `<shortdesc>` element also can be used to override the short description of the topic, when the `<copy-to>` attribute is specified.

**Note:** Processors may or may not implement this behavior.

**Inheritance**

"- topic/shortdesc " when used in topics, and "- map/shortdesc " when used in maps.

**Examples**

The following example demonstrates the use of a stand-alone `<shortdesc>` inside of a concept topic.

```
<concept id="concept">
  <title>Introduction to Bird Calling</title>
  <shortdesc> If you wish to attract more birds to your Acme Bird Feeder, learn the art of bird calling. Bird calling is an efficient way to alert more birds to the presence of your bird feeder. </shortdesc>
  <conbody>
    <p>Bird calling requires learning:</p>
  </conbody>
</concept>
```
Example: short description within a map

```xml
<topicref href="myThing.dita">
  <topicmeta>
    <navtitle>Navigation title for my topic</navtitle>
    <shortdesc>A description of myThing that is specific to this context.</shortdesc>
  </topicmeta>
</topicref>

<topicref href="http://www.example.org" scope="external">
  <topicmeta>
    <navtitle>Example website</navtitle>
    <shortdesc>The example.org address is often used in examples</shortdesc>
  </topicmeta>
</topicref>
```

**Example: `<abstract>` with phrase-level short description**

```xml
<abstract>The abstract is being used to provide more complex content.</abstract>
  <shortdesc>The shortdesc must be directly contained by the abstract.</shortdesc>
  The abstract can put text around the shortdesc.
```

**Topic output**

The abstract is being used to provide more complex content. The shortdesc must be directly contained by the abstract. The abstract can put text around the shortdesc.

**Preview/summary output**

The shortdesc must be directly contained by the abstract.

**Example: `<abstract>` with block-level short description**

```xml
<abstract>
  <p>The abstract is being used to provide more complex content.</p>
  <shortdesc>The shortdesc must be directly contained by the abstract.</shortdesc>
  <p>The abstract can put text around the shortdesc.</p>
</abstract>
```

**Topic output**

The abstract is being used to provide more complex content. The shortdesc must be directly contained by the abstract. The abstract can put text around the shortdesc.

**Preview/summary output**

The shortdesc must be directly contained by the abstract.

**Example: `<abstract>` with multiple short descriptions**

```xml
<abstract>The abstract is being used to provide more complex content.</abstract>
  <shortdesc>The shortdesc must be directly contained by the abstract.</shortdesc>
  <p>The abstract can put text around the shortdesc.</p>
  <shortdesc>There can be more than one shortdesc.</shortdesc>
```

**Topic output**

The abstract is being used to provide more complex content. The shortdesc must be directly contained by the abstract. The abstract can put text around the shortdesc. There can be more than one shortdesc.
The abstract is being used to provide more complex content. The shortdesc must be directly contained by the abstract.

The abstract can put text around the shortdesc.

There can be more than one shortdesc.

**Preview/summary output**

The shortdesc must be directly contained by the abstract. There can be more than one shortdesc.

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

### 4.2.1.7 `<abstract>`

The `<abstract>` element occurs between the topic title and the topic body, as the initial content of a topic. It can contain paragraph-level content as well as one or more `<shortdesc>` elements which can be used for providing link previews or summaries. The `<abstract>` element cannot be overridden by maps, but its contained `<shortdesc>` elements can be, for the purpose of creating link summaries or previews.

Use the `<abstract>` element when the initial paragraph of a topic is unsuitable for use as a link preview or for summaries, because, for example, it contains lists or tables, or because only a portion of the paragraph is suitable. Note that when the initial paragraph is suitable as a summary, that content should be placed in a `<shortdesc>` element rather than in an `<abstract>` element. The `<abstract>` element allows for a wider range of content in your initial paragraph, such as lists and tables, and allows you to identify portions of the `<abstract>` content as useful for previews or summaries by embedding the `<shortdesc>` element within `<abstract>`.

When the contained `<shortdesc>` occurs within phrase-level content, it is treated as phrase-level content and should not create a separate paragraph on output of the topic. When the contained `<shortdesc>` occurs as a peer to paragraph-level content, it is treated as block-level content and should create a separate paragraph on output of the topic. When multiple `<shortdesc>` elements are included in an `<abstract>`, they are concatenated in output of link previews or summaries (separated by spaces).

When a `<shortdesc>` element occurs in a DITA map, it overrides the short description provided in the topic for the purpose of generating link previews, but does not replace the `<shortdesc>` in the rendered topic itself. This means that generated links to this topic will use the short description from the map for purposes any link previews provided with the link, while the rendered topic continues to use the short description inside the topic. If the `<topicref>` element in the DITA map also specifies the `@copy-to` attribute, the content of the `<shortdesc>` element in the DITA map also overrides the short description provided in the topic. In this case, the rendered topic itself will display the `<shortdesc>` contents from the map in place of the `<shortdesc>` originally specified in the topic.

**Note:** Processors may or may not implement this behavior.

**Inheritance**

- topic/abstract

**Example: `<abstract>` with phrase-level short description**

```xml
<abstract>
The abstract is being used to provide more complex content.
  <shortdesc>The shortdesc must be directly contained by the abstract.</shortdesc>
The abstract can put text around the shortdesc.
</abstract>
```

**Topic output**
The abstract is being used to provide more complex content. The shortdesc must be directly contained by the abstract. The abstract can put text around the shortdesc.

**Preview/summary output**

The shortdesc must be directly contained by the abstract.

**Example: `<abstract>` with block-level short description**

```xml
<abstract>
  <p>The abstract is being used to provide more complex content.</p>
  <shortdesc>The shortdesc must be directly contained by the abstract.</shortdesc>
  <p>The abstract can put text around the shortdesc.</p>
</abstract>
```

**Topic output**

The abstract is being used to provide more complex content.

The shortdesc must be directly contained by the abstract.

The abstract can put text around the shortdesc.

**Preview/summary output**

The shortdesc must be directly contained by the abstract.

**Example: `<abstract>` with multiple short descriptions**

```xml
<abstract>
  The abstract is being used to provide more complex content.
  <shortdesc>The shortdesc must be directly contained by the abstract.</shortdesc>
  <p>The abstract can put text around the shortdesc.</p>
  <shortdesc>There can be more than one shortdesc.</shortdesc>
</abstract>
```

**Topic output**

The abstract is being used to provide more complex content. The shortdesc must be directly contained by the abstract.

The abstract can put text around the shortdesc.

There can be more than one shortdesc.

**Preview/summary output**

The shortdesc must be directly contained by the abstract. There can be more than one shortdesc.

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

**4.2.1.8 `<body>`**

The `<body>` element is the container for the main content of a `<topic>`.

**Inheritance**

- topic/body

**Example**

```xml
<topic>
  <title>Sample title</title>
  <prolog>!-- metadata here --></prolog>
```
Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.2.1.9 <bodydiv>

The `<bodydiv>` element is used to contain informal blocks of information within the body of a topic. The `<bodydiv>` element is specifically designed to be a grouping element, without any explicit semantics, other than to organize subsets of content into logical groups that are not intended or should not be contained as a topic. As such, it does not contain an explicit title to avoid enabling the creation of deeply nested content that would otherwise be written as separate topics. Content that requires a title should use a `<section>` element or a nested topic.

The `<bodydiv>` element may nest itself, which means that it can be used as a specialization base to create structured information within a body. Another common use case for the `<bodydiv>` element is to group a sequence of related elements for reuse, so that another topic may reference the entire set with a single `@conref` attribute.

Because the `<bodydiv>` element allows `<section>`, it cannot be used within `<section>` elements. Use the `<div>` element to group content that might occur in both topic bodies and sections.

Inheritance
- topic/bodydiv

Example

```
<topic id="sample" xml:lang="en">
  <title>Sample for bodydiv</title>
  <body>
    <bodydiv id="div">
      <p>This set of information is reusable as a group.</p>
      <p>Lists of three contain three items.</p>
      <ul>
        <li>This is one item.</li>
        <li>This is another item.</li>
        <li>This is the third item.</li>
      </ul>
    </bodydiv>
    <p>This concludes my topic.</p>
  </body>
</topic>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.2.1.10 <related-links>

The related information links of a topic (<related-links> element) are stored in a special section following the body of the topic. After a topic is processed into its final output form, the related links are usually displayed at the end of the topic, although some Web-based help systems might display them in a separate navigation frame.

Links specified within the `<related-links>` element are typically displayed together with links generated based on a map context; see *DITA linking* (see page 58) for more information on map based linking.
Processing notes:

1. Links within a `<linklist>` element must appear in the order defined, while those outside of a `<linklist>` may be sorted and displayed in a different order or location (based upon their role, target, importance, or other qualifiers).

2. PDF output typically ignores hierarchical links such as those with roles of ancestor, parent, child, descendant, next, previous, or sibling, although this behavior is not required.

Inheritance

- topic/related-links

Example

The following indicates that the two external links are always applicable to this topic, regardless of how the topic is used.

```xml
<related-links scope="external" format="html">
  <link href="http://www.example.org">
    <linktext>Example 1</linktext>
  </link>
  <link href="http://www.example.com">
    <linktext>Example 2</linktext>
  </link>
</related-links>
```

Attributes

The following attributes are available on this element: `Universal attribute group` (see page 650), `Link relationship attribute group` (see page 661) (apart from `@href`), `The role and otherrole attributes` (see page 677), and `outputclass` (see page 0).

4.2.1.11 `<dita>`

The `<dita>` element provides a top-level container for multiple topics when you create documents using the ditabase document type. The `<dita>` element lets you create any sequence of concept, task, and reference topics, and the ditabase document type lets you nest these topic types inside each other. The `<dita>` element has no particular output implications; it simply allows you to create multiple topics of different types at the same level in a single document.

Inheritance

Not a specializable DITA element.

Example

```xml
<dita>
  <concept id="batintro"><title>Intro to bats</title><conbody><![endif>... }}</conbody></concept>
  <reference id="batparts"><title>Parts of bats</title><refbody><![endif>... }}</refbody></reference>
  <task id="batfeeding"><title>Feeding a bat</title><taskbody><![endif>... }}</taskbody></task>
  <task id="battraining"><title>Training a bat</title><taskbody><![endif>... }}</taskbody></task>
  <task id="batcleanup"><title>Cleaning a bat</title><taskbody><![endif>... }}</taskbody></task>
</dita>
```

Attributes

The following attributes are available on this element: `@xmlns:ditaarch` and `@DITAArchVersion` from `Architectural attribute group` (see page 654), `@xml:lang` from `Localization attribute group` (see page 653) and `Debug attribute group` (see page 654).
4.2.2 Body elements

The body elements support the most common types of content authoring for topics: paragraphs, lists, phrases, figures, and other common types of exhibits in a document.

4.2.2.1 <alt>

The <alt> element provides alternate text for an image. It is equivalent to the @alt attribute on the <image> element; since the attribute is deprecated, use the <alt> element instead. As an element, <alt> provides direct text entry within an XML editor and is more easily accessed than an attribute for translation.

**Inheritance**
- topic/alt

**Example**

The markup for alternate text within an image looks like this:

```xml
<image href="tip-ing.jpg">
  <alt>Here's a Tip!</alt>
</image>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.2.2.2 <cite>

The <cite> element is used when you need a bibliographic citation that refers to a book or article. It specifically identifies the title of the resource.

Though citations will often be set apart from surrounding text, such as through italics, rendering of the <cite> element is left up to implementations.

**Inheritance**
- topic/cite

**Example**

```xml
<p>The online article <cite>Specialization in the Darwin Information Typing Architecture</cite> provides a detailed explanation of how to define new topic types.</p>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and @keyref.

4.2.2.3 <dd>

The <dd> element contains the description of a term in a definition list entry.

**Inheritance**
- topic/dd
Example

```xml
<dl>
  <dlentry>
    <dt>Bytes returned</dt>
    <dd>The number of bytes of data returned.</dd>
  </dlentry>
  <dlentry>
    <dt>Bytes available</dt>
    <dd>The number of bytes of data available to be returned.</dd>
  </dlentry>
  <dlentry>
    <dt>Handle</dt>
    <dd>The returned handle value</dd>
  </dlentry>
</dl>
```

Figure 42: Simple definition list example

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 650).

4.2.2.4 `<desc>`

The `<desc>` element contains the description of the current element. In elements that also allow a title, such as `<table>` and `<fig>`, this is used to provide more information than is appropriate for the title. In the `<xref>` and `<link>` elements it contains a description of the target; processors MAY choose to display this as hover help for a link. In the `<object>` element, `<desc>` provides alternate content to use when the context does not permit displaying the object.

Inheritance

- topic/desc

Example

```xml
<fig>
  <title>The Handshake</title>
  <desc>This image shows two hands clasped in a formal, business-like handshake.</desc>
  <image href="handshake.jpg">
    <alt>The handshake</alt>
  </image>
</fig>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 650).

4.2.2.5 `<ddhd>`

The `<ddhd>` element provides an optional heading or title for a column of descriptions or definitions in a `<dl>`.

Inheritance

- topic/ddhd

Example

```xml
<dl>
  <dlhead>
```

DITA Version 1.3 Specification

Copyright © OASIS Open 2005, 2014. All Rights Reserved.
**Figure 43: Definition list with a heading**

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

**4.2.2.6 <div>**

The `<div>` element is used to organize subsets of content into logical groups that are not intended to be or should not be contained as a topic. The `<div>` element is designed to be a grouping element; it does not imply any explicit semantics or contain an explicit title. This avoids enabling the creation of deeply-nested content that would otherwise be written as separate topics. Content that requires a title should use a `<section>` element, a nested `<topic>`, or possibly a `<fig>` element.

**Inheritance**

- `topic/div`

**Example**

In the following example, a `<div>` element is used to organize several elements together so that they can be referenced by `@conref` or `@conkeyref`:

```xml
...  
<div id="div-01">
  <p>The first paragraph</p>
  <p>The second paragraph</p>
  <note>This is a note</note>
</div>
...  
```

Without using a `<div>` element, the content could not be grouped for content referencing since the start and end elements are of different types.

**Figure 44: Using `<div>` for grouping**

In the following example, `<div>` is used as the basis for specializing a new domain element, `<pullquote>`:

```xml
<!ENTITY % pullquote.content "(\div.cnt\)\*" >
<!ENTITY % pullquote.attributes "%univ-atts; outputclass CDATA" >
```
Instances of `<pullquote>` could then be used in both `<body>` and `<section>` contexts:

```xml
<topic id="article-01">
  <title>My Article</title>
  <body>
    <p>Something pithy someone said</p>
    <pullquote><p>Something Pithy</p></pullquote>
    <!-- ... -->
    <section spectitle="Deep Dive">
      <p>This is really really pithy</p>
      <pullquote><p>Really Pithy</p></pullquote>
      <!-- ... -->
    </section>
  </body>
</topic>
```

*Figure 45: Using `<div>` for specialization*

### Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

#### 4.2.2.7 `<dl>`

A `<dl>` element is a list of terms and corresponding definitions. The term `<dt>` is usually flush left. The description or definition `<dd>` is usually either indented and on the next line, or on the same line to the right of the term. However, actual rendering is up to the rendering engine.

An optional heading for the terms and definitions may be provided using the `<dlhead>` element, which contains header elements for those columns. The default formatting for the `<dlhead>` generally looks like a table with a heading row, but this is also up to the rendering engine.

### Inheritance

- topic/dl

### Examples

```xml
<dl>
  <dlentry>
    <dt>Bytes returned</dt>
    <dd>The number of bytes of data returned.</dd>
  </dlentry>
  <dlentry>
    <dt>Bytes available</dt>
    <dd>The number of bytes of data available to be returned.</dd>
  </dlentry>
  <dlentry>
    <dt>Handle</dt>
    <dd>The returned handle value</dd>
  </dlentry>
</dl>
```

*Figure 46: Simple definition list example*
<dlentry><dt>File Type</dt><dd>Image's file extension</dd></dlentry><dlentry><dt>Image Class</dt><dd>Image is raster, vector, metafile or 3D</dd></dlentry><dlentry><dt>Number of pages</dt><dd>Number of pages in the image</dd></dlentry><dlentry><dt>Fonts</dt><dd>Names of the fonts contained within a vector image</dd></dlentry>

**Figure 47: Definition list with a heading**

Rendering of definition lists will vary by application and by display format. Processors *MIGHT* render the second example as follows.

<table>
<thead>
<tr>
<th>Image File View Selection</th>
<th>Resulting Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Type</td>
<td>Image's file extension</td>
</tr>
<tr>
<td>Image Class</td>
<td>Image is raster, vector, metafile or 3D</td>
</tr>
<tr>
<td>Number of pages</td>
<td>Number of pages in the image</td>
</tr>
<tr>
<td>Fonts</td>
<td>Names of the fonts contained within a vector image</td>
</tr>
</tbody>
</table>

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), *compact* (see page 0), and *spectitle* (see page 0).

**4.2.2.8 <dlentry>**

The `<dlentry>` element groups a single entry in a definition list. The `<dlentry>` element includes a term `<dt>` and one or more definitions or descriptions `<dd>` of that term.

**Inheritance**

- topic/dlentry

**Example**

```xml
<dl>
  <dlentry>
    <dt>Bytes returned</dt>
    <dd>The number of bytes of data returned.</dd>
  </dlentry>
  <dlentry>
    <dt>Bytes available</dt>
    <dd>The number of bytes of data available to be returned.</dd>
  </dlentry>
  <dlentry>
    <dt>Handle</dt>
    <dd>The returned handle value</dd>
  </dlentry>
</dl>
```

**Figure 48: Simple definition list example**
Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.2.2.9 <dlhead>

The <dlhead> element contains optional headings for the term and description columns in a definition list. The definition list heading may contain a heading for the column of terms (<dthd>) and a heading for the column of descriptions (<ddhd>).

Inheritance

- topic/dlhead

Example

```
<dl>
  <dlhead>
    <dthd>Image File View Selection</dthd>
    <ddhd>Resulting Information</ddhd>
  </dlhead>
  <dlentry>
    <dt>File Type</dt>
    <dd>Image's file extension</dd>
  </dlentry>
  <dlentry>
    <dt>Image Class</dt>
    <dd>Image is raster, vector, metafile or 3D</dd>
  </dlentry>
  <dlentry>
    <dt>Number of pages</dt>
    <dd>Number of pages in the image</dd>
  </dlentry>
  <dlentry>
    <dt>Fonts</dt>
    <dd>Names of the fonts contained within a vector image</dd>
  </dlentry>
</dl>
```

Figure 49: Definition list with a heading

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.2.2.10 <dt>

The <dt> element contains a term in a definition list entry.

Inheritance

- topic/dt

Example

```
<dl>
  <dlentry>
    <dt>Bytes returned</dt>
    <dd>The number of bytes of data returned.</dd>
  </dlentry>
  <dlentry>
    <dt>Bytes available</dt>
    <dd>The number of bytes of data available to be returned.</dd>
  </dlentry>
```
<dlentry>
<dt>Handle</dt>
<dd>The returned handle value</dd>
</dlentry>

**Figure 50: Simple definition list example**

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and @keyref.

### 4.2.2.11 <draft-comment>

The `<draft-comment>` element is designed to facilitate review and discussion of topic contents within the marked-up content. Use the `<draft-comment>` element to ask a question or to make a comment that you want others to review. To indicate the source of the draft comment or the status of the comment, use the @author, @time, or @disposition attributes.

Processing systems should provide a run-time flag or parameter to cause the content of this element to be specially displayed for draft output only. By default, processors should strip them out to prevent publishing internal comments by mistake.

**Inheritance**

- topic/draft-comment

**Example**

```
<draft-comment author="EBP">Where's the usage information for this section?</draft-comment>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition for @translate, given below), *outputclass* (see page 0), and the attributes defined below.

- @translate
  Indicates whether the content of the element should be translated or not. For this element the default value is "no". Setting to "yes" will override the default. The DITA architectural specification contains a list of each OASIS DITA element and its common processing default for the translate value; because this element uses an actual default, it will always be treated as translate="no" unless overridden as described. Available values are:
  - no
    The content of this element is not translateable.
  - yes
    The content of this element is translateable.

- @author
  Designates the originator of the draft comment. This attribute is defined with the XML Data Type CDATA.

- @time
  Describes when the draft comment was created. This attribute is defined with the XML Data Type CDATA.

- @disposition
  TBD
Status of the draft comment. Prior to DITA 1.2, this attribute was limited to the following values: issue, open, accepted, rejected, deferred, duplicate, reopened, unassigned, or completed. This attribute is defined with the XML Data Type CDATA.

4.2.2.12 <dthd>

The <dthd> element provides an optional heading for the column of terms in a <dl>.

**Inheritance**

- topic/dthd

**Example**

```xml
<dl>
  <dlhead>
    <dthd>Image File View Selection</dthd>
    <dthd>Resulting Information</dthd>
  </dlhead>
  <dlentry>
    <dt>File Type</dt>
    <dd>Image's file extension</dd>
  </dlentry>
  <dlentry>
    <dt>Image Class</dt>
    <dd>Image is raster, vector, metafile or 3D</dd>
  </dlentry>
  <dlentry>
    <dt>Number of pages</dt>
    <dd>Number of pages in the image</dd>
  </dlentry>
  <dlentry>
    <dt>Fonts</dt>
    <dd>Names of the fonts contained within a vector image</dd>
  </dlentry>
</dl>
```

*Figure 51: Definition list with a heading*

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.2.2.13 <example>

The <example> element is a section that contains examples that illustrate or support the current topic.

DITA uses <example> to contain both sample code (or similar artifacts) and the discussion that illustrates the sample. For example, a DITA topic about programming code would use the <example> element to contain both the sample code and the text that describes the code. Similarly, a sample that included preformatted text might use the <pre> element along with a paragraph or list to elaborate upon sections of that text.

**Inheritance**

- topic/example

**Example**

```xml
<example id="example">
  <title>Example</title>
  <codeblock>&lt;p&gt;Example of the p element&lt;/p&gt;</codeblock>
</example>
```
Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and spectitle (see page 0).

4.2.2.14 <fig>

The <fig> element is a figure (sometimes called an "exhibit") with an optional title for a wide variety of content. A figure commonly contains an image or artwork, but it can contain several kinds of text objects as well. A title is placed inside the figure to provide a caption that describes the content.

Inheritance
- topic/fig

Example

```xml
<fig expanse="column">
  <title>The Handshake</title>
  <image href="handshake.jpg" placement="break">
    <alt>The Handshake</alt>
  </image>
</fig>
```

4.2.2.15 <figgroup>

The <figgroup> element is used primarily for specialization, in order to create segments within a figure. The element may nest itself, which allows it to create complex specialized structures (such as the nestable groups of syntax within a syntax diagram). Figure groups can be used to contain multiple cross-references, footnotes or keywords, but not multipart images. Multipart images in DITA should be represented by a suitable media type displayed by the <object> element.

Inheritance
- topic/figgroup

Example

```xml
<fig>
  <title>Sample complex figure</title>
  <figgroup>
    <title>First group</title>
    <ph>These elements</ph>
    <ph>are grouped together</ph>
    <ph>for some purpose</ph>
  </figgroup>
  <figgroup>
    <title>Second group</title>
    <data name="MetaItem" value="13"/>
    <data name="MetaThing" value="31"/>
    <ph>These elements</ph>
    <ph>are grouped with associated metadata</ph>
  </figgroup>
</fig>
```
4.2.2.16 <fn>

The <fn> element is a footnote used to annotate text with notes that are inappropriate for inline inclusion. It is also used to indicate the source for facts or other material used in the text.

Footnote content is skipped at the place where it was entered and rendered elsewhere, according to these rules:

- A footnote with no given @id attribute is a single-use footnote. Upon output, it generates a number as a superscript "callout" that is linked to the placement of the footnote, such as at the bottom of the immediate printed page or at the end of an online article. If a character is specified in the @callout attribute for the footnote, that character should be used as the superscript "callout" that is linked to the placement of the footnote.
- A footnote entered with an @id attribute is a use-by-reference footnote. Upon output, it does not appear anywhere unless it has been referenced using an <xref> with the @type attribute set to "fn".
- Ordinarily, a footnote in one topic can't be referenced in another topic. The previous behaviors are local to each topic. But by using the conref mechanism, you can create a new copy of another topic's footnote within the local topic where it will then follow these behaviors:
  - If you use <fn conref="file.dita#topic/thatid"></fn> all by itself, the result will be the same as the single-use footnote entered literally in the same location. That is, it creates a local copy of the footnote with no local @id attribute, so it uses the behavior from the first bullet above.
  - If you use <fn conref="file.dita#topic/thatid" id="thisid"></fn>, followed by <xref href="#thistopic/thisid" type="fn" />, the result will be the same as the use-by-reference model described in the second bullet. That is, the <fn> element creates a local copy of the footnote with an id of "thisid"; that local copy is then referenced by the <xref> element.

Note: The details of footnote processing and styling are implementation and/or stylesheet dependent. For example, a tool that renders DITA as PDF may lack support for the @callout attribute, or footnotes may be collected as endnotes for certain types of publications.

Inheritance
- topic/fn

Example

The first example is of a single-use footnote. It uses a simple <fn> element, with no @id and no @callout attribute. In that case, markup such as the following:

```
The memory storage capacity of the computer is 2 GB<fn>A GB (gigabyte) is equal to 1000 million bytes</fn> with error correcting support.
```

may produce output similar to the following:

```
The memory storage capacity of the computer is 2 GB\(^1\) with error correcting support.

\[\text{\ldots\ldots} \]

\(^1\) A GB (gigabyte) is equal to 1000 million bytes

----- [bottom of page] ----------------------------------------
```
The second example is a single-use footnote that uses a @callout attribute. It is marked up as follows:

The memory storage capacity of the computer is 2 GB<fn callout="#">A GB (gigabyte) is equal to 1000 million bytes</fn> with error correcting support.

That DITA markup may produce output similar to the following:

Table:  | The memory storage capacity of the computer is 2 GB<fn callout="#">A GB (gigabyte) is equal to 1000 million bytes</fn> with error correcting support.

The third example is a use-by-reference footnote. It uses an @id on a footnote, and then references that @id multiple times. The DITA markup looks like this:

I like pets. <fn id="reuse-fn">This is the name of an animal.</fn> At my house, I have a dog<xref href="#topic/reuse-fn" type="fn"/>, a cat<xref href="#topic/reuse-fn" type="fn"/> and a llama<xref href="#topic/reuse-fn" type="fn"/>.

and may produce output similar to the following:

Table: | I like pets. At my house, I have a dog<sup>1</sup>, a cat<sup>1</sup>, and a llama<sup>1</sup>. | 1This is the name of an animal.

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and the attribute defined below.

@callout

Specifies what character is used for the footnote link, for example a number or an alpha character. The attribute may also specify a short string of characters. When no @callout value is specified, footnotes are numbered. This attribute is defined with the XML Data Type CDATA.

4.2.2.17 <image>

Use the <image> element to include artwork or images in a DITA topic.

The <image> element has an optional @placement attribute that indicates whether the placement of the included graphic or artwork should be inline (a button or icon) or on a separate line for a larger image. There also are also attributes that indicate the size to which the included graphic or artwork should be scaled.

An image element specifies an @href attribute, a @keyref attribute, or both. When both @keyref and @href are specified, the @href is used as a fallback when the key reference cannot be resolved. The image addressed by the @keyref or @href is brought into the main flow of the content as rendered.

To make the intent of the image more accessible for users using screen readers or text-only readers, authors should include a description of the image content in the <alt> element.
**Inheritance**
- topic/image

**Example**

```xml
<image href="bike.gif" placement="break">
  <alt>Two-wheeled bicycle</alt>
</image>
```

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), @keyref, and the attributes defined below.

- **@href**
  Provides a reference to the image. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. This attribute is defined with the XML Data Type CDATA.

- **@scope**
  The @scope attribute identifies the closeness of the relationship between the current document and the target resource. Allowable values are local, peer, external, and -dita-use-conref-target. See *The scope attribute* (see page 677) for more information on values.

- **@height**
  Indicates the vertical dimension for the resulting image display. If necessary, the image is scaled to the specified size. The value of this attribute is a real number (expressed in decimal notation) optionally followed by a unit of measure from the set of pc, pt, px, in, cm, mm, em (picas, points, pixels, inches, centimeters, millimeters, and ems respectively). The default unit is px (pixels). Possible values include: "5", "5in", and "10.5cm". If a height value is specified and no width value is specified, the width will be scaled by the same factor as the height. If both a height value and width value are specified, some implementations may not be able to scale the two directions by a different factor and may therefore ignore one of the two values. This attribute is defined with the XML Data Type NMTOKEN.

- **@width**
  Indicates the horizontal dimension for the resulting image display. If necessary, the image is scaled to the specified size. The value of this attribute is a real number (expressed in decimal notation) optionally followed by a unit of measure from the set of pc, pt, px, in, cm, mm, em (picas, points, pixels, inches, centimeters, millimeters, and ems respectively). The default unit is px (pixels). Possible values include: "5", "5in", and "10.5cm". If a width value is specified and no height value is specified, the height will be scaled by the same factor as the width. If both a height value and width value are specified, some implementations may not be able to scale the two directions by a different factor and may therefore ignore one of the two values. This attribute is defined with the XML Data Type NMTOKEN.

- **@align**
  Controls the horizontal alignment of an image when placement is specified as "break". Common values include left, right, and center. This attribute is defined with the XML Data Type CDATA.

- **@scale**
  Specifies a percentage 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 this image's @height or @width attribute (or both), the @scale attribute is ignored. This attribute is defined with the XML Data Type NMTOKEN.

  It is an error if the value of this attribute is not an unsigned integer. In this case, the implementation may (but need not) give an error message and may (but need not) recover by ignoring this attribute.

- **@scalefit**
Allow an image to be scaled up or down to fit within available space. Allowable values are yes, no, and `-dita-use-conref-target`. If, for a given image, any one of @height, @width, or @scale is specified, those attributes determine the graphic size, and any setting of @scalefit is ignored. If none of those attributes are specified and scalefit="yes", then the image is scaled (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 the prevailing column (or table cell) width—that is, the width a paragraph of text would have if the graphic were a paragraph instead. 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.

@placement
Indicates whether an image should be displayed inline or separated from the surrounding text. The default is inline. Allowable values are: inline, break, or and `-dita-use-conref-target`.

---

**Comment by robander**

Nov 27 2013: DITA 1.2 said "The processing default is inline". But the "Default" column listed "inline" as a true default, and this has been the DTD and Schema implementation. I've removed the word "processing" from the previous definition.

---

@alt (DEPRECATED)
Alternative text that describes the image to provide accessibility to page readers or provides a text description when an image cannot be displayed by the user's software. The @alt attribute is deprecated; use the alt (see page 258) element instead. This attribute is defined with the XML Data Type CDATA.

@longdescref (DEPRECATED)
A reference to a textual description of the graphic or object. This attribute supports creating accessible content. See The href attribute (see page 663) for detailed information on supported values and processing implications. For examples of how this attribute is used in output, see this topic on long descriptions. NOTE: This attribute is deprecated in favor of the longdescref (see page 272) subelement to this element. This attribute is defined with the XML Data Type CDATA.

---

### 4.2.2.18 `<keyword>`

The `<keyword>` element identifies a keyword or token, such as a single value from an enumerated list, the name of a command or parameter, product name, or a lookup key for a message.

A keyword contains any text that has a unique or key-like value, such as a product name. Where there is an element that has a better meaning for what you are describing, use that element. The `<keyword>` element is a generic element; use it when no other element applies. The `<keyword>` element can also be used to contain reusable text.

With DITA 1.2, another option for reusable text is the `text` (see page 284) element, which is designed to be free of any extra semantic information. The `text` element is available within `<keyword>`, and at least one of those elements should be available to allow content reuse in any context.

Specific markup recommendations:

- Use `<apiname>` for API names and `<cmdname>` for command names.
- Use `<term>` to indicate what you are defining with inline paragraph definitions.
- Use `<ph>` for general phrases when `<keyword>` is not appropriate.
- Use `<kwd>` to indicate programming keywords in syntax diagrams and syntax phrases.

Specialized elements derived from `<keyword>` may also have extended processing, such as different formatting or automatic indexing.

All `<keyword>` or `<indexterm>` elements in the `<keywords>` metadata element are considered part of the topic's metadata and should be processed accordingly as appropriate for the given output medium.
Note: While the `<keyword>` element may be used inline, the `<keywords>` element is not an inline element. The `<keywords>` element only appears in the `<topicmeta>` or `<prolog>`, and is used to specify keywords that apply to the topic.

Inheritance
- topic/keyword

```
<p>The `<keyword>assert</keyword>` pragma statement allows messages to be passed to the emulator, pre-compiler, etc..</p>
<p>The `<keyword id="myProduct">AmazingProduct</keyword>` can make use of this feature to do really neat stuff.</p>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and @keyref.

4.2.2.19 `<li>`

An `<sl>` element is a single item in an ordered `<ol>` or unordered `<ul>` list. When a DITA topic is rendered, numbers and alpha characters are usually displayed with list items in ordered lists, while bullets and dashes are usually displayed with list items in unordered lists.

Inheritance
- topic/li

Example

```
<ul>
  <li>This is an item in an unordered list.</li>
  <li>This is another item in an unordered list.</li>
</ul>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0 ).

4.2.2.20 `<lines>`

The `<lines>` element may be used to represent dialogs or text fragments where line breaks are significant. The `<lines>` element is similar to `<pre>` in that line breaks are preserved, but the font style is not set to monospace, and extra spaces inside the lines are not preserved.

Inheritance
- topic/lines

Example

This is a sample of my favorite sonnet.
<lines>
Shall I compare thee to a summer's day?
Thou art more lovely and more temperate:
Rough winds do shake the darling buds of May,
and summer's lease hath all too short a date:
...
</lines>
Though exact formatting will vary, the previous sample will typically be rendered as follows.

This is a sample of my favorite sonnet

Shall I compare thee to a summer's day?
Thou art more lovely and more temperate:
Rough winds do shake the darling buds of May,
and summer's lease hath all too short a date:
...

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Display attribute group (see page 660), outputclass (see page 0), xml:space (see page 0), and spectitle (see page 0).

4.2.2.21 <longdescref>

The <longdescref> element supports a reference to a text description of the graphic or object. This element replaces the deprecated @longdescref attribute on <image> and <object> elements.

Inheritance

- topic/longdescref

Example

```xml
<image href="llama.jpg">
<alt>Llama picture</alt>
<longdescref href="my-pet-llama.dita"/>
</image>
```

Figure 52: <longdescref> which references a local DITA description

```xml
<image href="puffin.jpg">
<alt>Puffin picture</alt>
<longdescref href="http://www.example.org/birds/puffin.html" scope="external" format="html"/>
</image>
```

Figure 53: <longdescref> which references an external description

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661), outputclass (see page 0), and @keyref.

4.2.2.22 <longquoteref>

The <longquoteref> element provides a reference to the source of a long quote.

The following comment added based on a review request that we provide processing information.

Rendering of this element is left up to DITA processors. Depending on the presentation format, it may be appropriate to ignore the element, present it as a link, use it to turn the entire quote into a link, or do something else.

Inheritance

- topic/longquoteref
Example

<p>A great person once said the following thing.</p>

<q>Examples are the key to any specification.<longquoteref href="http://www.example.org/quotes" scope="external"/></q>

Attributes

The following attributes are available on this element: <em>Universal attribute group</em> (see page 650), <em>Link relationship attribute group</em> (see page 661), <em>outputclass</em> (see page <a>0</a>), and <em>@keyref</em>.

4.2.2.23 <q>

The <q> is used to provide extended content quoted from another source. Use the quote element <q> for short, inline quotations, and long quote <q> for quotations that are too long for inline use, following normal guidelines for quoting other sources. The @href attribute may be used to specify the source of the quotation; @href may reference another DITA topic or an external resource. The <longquoteref> (added in DITA 1.2) is available for more complex references to the source of a quote.

Most of the following comment was pulled from the HTML 4.01 specification's description of BLOCKQUOTE, after a review comment indicated that the spec should describe rendering expectations.

Although rendering is left up to implementations, processors generally render <q> as an indented block.

Inheritance

- topic/lq

Example

<p>This is the first line of the address that Abraham Lincoln delivered on November 19, 1863 for the dedication of the cemetery at Gettysburg, Pennsylvania.</p>

<q>Four score and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal.</q>

Attributes

The following attributes are available on this element: <em>Universal attribute group</em> (see page 650), <em>Link relationship attribute group</em> (see page 661) (with a narrowed definition for @type, given below), <em>outputclass</em> (see page <a>0</a>), and <em>@keyref</em>, and the attributes defined below.

@reftitle

The title of the document or topic being quoted. This attribute is defined with the XML Data Type CDATA.

@type

Indicates the location of the source of the quote. Note that this differs from the @type attribute on many other DITA elements. See <em>The type attribute</em> (see page 674) for detailed information on the usual supported values and processing implications. This attribute is defined with the XML Data Type CDATA. The following attribute values are allowed (but deprecated) for backward compatibility:

external

The @href is to a Web site. This value is deprecated in favor of use of the @scope and @format attributes.

internal

The @href is to a DITA topic. This value is deprecated in favor of use of the @scope and @format attributes.
4.2.2.24 <note>

A <note> element contains information which expands on or calls attention to a particular point. This information is typically differentiated from the main text.

Tip: Variant types of <note> (tip, caution, danger, restriction, etc.) can be indicated through values selected on the @type attribute. This <note> uses the attribute type="tip"

Inheritance

- topic/note

Example

This example:

```xml
<note type="tip">Thinking of a seashore, green meadow, or cool mountain overlook can help you to relax and be more patient.</note>
```

produces this result:

Tip: Thinking of a seashore, green meadow, or cool mountain overlook can help you to relax and be more patient.

Comment by robander, 20 February 2014

Seems like this should be cleaned up; it implies that the output from this note element is normative. May also want to clean up the sample "tip" that appears before the contains/contained-by info.

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), spectitle (see page 0), and the attributes defined below.

@type

Defines the type of a note. For example, if the note is a tip, the word Tip may be used to draw the reader's attention to it. Note that this differs from the @type attribute on many other DITA elements. See The type attribute (see page 674) for detailed information on supported values and processing implications. Available values are note, tip, fastpath, restriction, important, remember, attention, caution, notice, danger, warning, trouble, other, and -dita-use-conref-target.

@othertype

Indicates an alternate note type, when the type is not available in the @type attribute value list. This value is used as the user-provided note title when the @type attribute value is set to "other". This attribute is defined with the XML Data Type CDATA.

4.2.2.25 <object>

The DITA <object> element corresponds to the HTML <object> element, and attribute semantics derive from their HTML definitions. For example, the @type attribute differs from the @type attribute on many other DITA elements.

The <object> element enables authors to include animated images, applets, plug-ins, ActiveX controls, video clips, and other multimedia objects in a topic.

Inheritance

- topic/object
Output processors might need to modify data in order to enable compatible function across various browsers, so these examples are only representative:

```xml
<param name="movie" value="cutkey370.swf"/>
<param name="quality" value="high"/>
<param name="bgcolor" value="#FFFFFF"/>
```

What's EIM?

```xml
<param name="movie" value="eim.swf"/>
<param name="quality" value="high"/>
<param name="bgcolor" value="#FFFFFF"/>
<param name="pluginspace" value="http://www.macromedia.com/go/getflashplayer"/>
```

Where the keys could be:

```xml
<param name="poster: keyref="E5123_026_poster"/>
<param name="source" keyref="E5123_026_video"/>
```

**Figure 54: Object with reference to video using key reference on the `<param>` elements**

```xml
<object classidkeyref="video_classid" codebasekeyref="video_codebase" datakeyref="cutkey370" height="280" width="370" id="cutkey370">
<desc>A description of the task</desc>
</object>
```
Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and the attributes defined below.

@archive

Specifies a space-separated list of URIs indicating resources needed by the object. These resources may include those URIs specified by the @classid and @data attributes. Preloading these resources usually results in faster loadtimes for objects. The URIs in the list should be relative to the URI specified in the @codebase attribute. This attribute is defined with the XML Data Type CDATA.

@archivekeyrefs

Key references to one or more archives, as for @archive. The value is a space-separated list of key names. Each resolvable key reference is treated as a URI as though it had been specified on the @archive attribute. When specified, and at least one key name is resolvable, the key-provided archive list is used. If @archive is specified, it is used as a fallback when no key names can be resolved to a URI. This attribute is defined with the XML Data Type CDATA.

@classid

Contains a URI that specifies the location of an object’s implementation. It can be used together with the @data attribute which is specified relative to the value of the @codebase attribute. This attribute is defined with the XML Data Type CDATA.

@classidkeyref

Key reference to the URI that specifies the location of an object’s implementation, as for @classid. When specified, and the key is resolvable, the key-provided class ID URI is used. If @classid is specified, it is used
as a fallback when the key cannot be resolved to a URI. This attribute is defined with the XML Data Type CDATA.

@codebase
Specifies the base URI used for resolving the relative URI values given for @classid, @data, and @archive attributes. If @codebase is not set, the default is the base URI of the current element. This attribute is defined with the XML Data Type CDATA.

@codebasekeyref
Key reference to the base URI used for resolving other attributes, as for @codebase. When specified, and the key is resolvable, the key-provided code base URI is used. If @codebase is specified, it is used as a fallback if the key cannot be resolved to a URI. If no URI results from processing @codebasekeyref and @codebase is not specified, the default is the base URL of the current element. This attribute is defined with the XML Data Type CDATA.

@data
Contains a reference to the location of an object's data. If this attribute is a relative URL, it is specified relative to the value of the @codebase attribute. If this attribute is set, the @type attribute should also be set. This attribute is defined with the XML Data Type CDATA.

@datakeyref
Key reference to the location of an object's data, as for @data. When specified and the key is resolvable, the key-provided URI is used. A key that has no associated resource, only link text, is considered to be unresolved for the purpose of the @datakeyref attribute. If @data is specified, it is used as a fallback when the key cannot be resolved to a resource. This attribute is defined with the XML Data Type CDATA.

@declare
When this attribute is set to "declare", the current object definition is a declaration only. The object must be instantiated by a later nested object definition referring to this declaration. The only allowable value is "declare".

@type
Indicates the content type for the data specified by the @data or @datakeyref attribute. This attribute should be set when the @data attribute is set to avoid loading unsupported content types. Note that this differs from the @type attribute on many other DITA elements. If @type is not specified, the effective type value for the key named by the @datakeyref attribute is used as the this attribute's value.

@standby
Contains a message to be displayed while an object is loading. This attribute is defined with the XML Data Type CDATA.

@height
Indicates the vertical dimension for the resulting object display. If necessary, the object is scaled to the specified size. The value of this attribute is a real number (expressed in decimal notation) optionally followed by a unit of measure from the set of pc, pt, px, in, cm, mm, em (picas, points, pixels, inches, centimeters, millimeters, and ems respectively). The default unit is px (pixels). Possible values include: "5", "5in", and "10.5cm". If a height value is specified and no width value is specified, the width will be scaled by the same factor as the height. If both a height value and width value are specified, some implementations may not be able to scale the two directions by a different factor and may therefore ignore one of the two values. This attribute is defined with the XML Data Type NMTOKEN.

@width
Indicates the horizontal dimension for the resulting object display. If necessary, the object is scaled to the specified size. The value of this attribute is a real number (expressed in decimal notation) optionally followed by a unit of measure from the set of pc, pt, px, in, cm, mm, em (picas, points, pixels, inches, centimeters, millimeters, and ems respectively). The default unit is px (pixels). Possible values include: "5", "5in", and "10.5cm". If a width value is specified and no height value is specified, the height will be scaled by the same factor as the width. If both a height value and width value are specified, some implementations may not be
able to scale the two directions by a different factor and may therefore ignore one of the two values. This attribute is defined with the XML Data Type NMTOKEN.

@usemap
Indicates that a client-side image map is to be used. An image map specifies active geometric regions of an included object and assigns a link to each region. When a link is selected, a document may be retrieved or a program may run on the server. This attribute is defined with the XML Data Type CDATA.

@name
Defines a unique name for the object. This attribute is defined with the XML Data Type CDATA.

@tabindex
Position the object in tabbing order. This attribute is defined with the XML Data Type NMTOKEN.

@longdesc (DEPRECATED)
A reference to a textual description of the graphic or object. This attribute supports creating accessible content. See The href attribute (see page 663) for detailed information on supported values and processing implications. For examples of how this attribute is used in output, see this topic on long descriptions. NOTE: This attribute is deprecated in favor of the longdesc (see page 272) subelement to this element. This attribute is defined with the XML Data Type CDATA.

4.2.2.26 <ol>
The <ol> element includes a list of items sorted by sequence or order of importance.

Inheritance
- topic/ol

Example

Here are the colors of the rainbow in order of appearance from top to bottom:
<ol>
<li>Red</li>
<li>Orange</li>
<li>Yellow</li>
<li>Green</li>
<li>Blue</li>
<li>Indigo</li>
<li>Violet</li>
</ol>

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), compact (see page 0 ), and spectitle (see page 0 ).

4.2.2.27 <p>
The <p> element is a single paragraph containing a single main idea.

Inheritance
- topic/p

Example

This paragraph contains text that should be of interest. It might give details about my company, or an explanation of how my product is used, or it might be a short review of the book I read last week.
### Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

#### 4.2.2.28 `<param>`

The `<param>` (parameter) element specifies a set of values that may be required by an `<object>` at runtime. Any number of `<param>` elements may appear in the content of an `<object>` in any order, but must be placed at the start of the content of the enclosing object. This element is comparable to the XHTML `<param>` element, and its attributes' semantics derive from their HTML definitions. For example, the `@type` attribute differs from the `@type` attribute on many other DITA elements.

### Inheritance

- topic/param

#### Example

See *object* (see page 274).

### Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and the attributes defined below.

- **@name** (REQUIRED)
  
  The name of the parameter. This attribute is defined with the XML Data Type CDATA.

- **@value**
  
  Specifies the value of a run-time parameter that is specified by the `@name` attribute. This attribute is defined with the XML Data Type CDATA.

- **@valuetype**
  
  Specifies the type of the `@value` attribute. Allowed values are:

    - **data**
      
      A value of data means that the value will be evaluated and passed to the object's implementation as a string.

    - **ref**
      
      A value of ref indicates that the value of the `@value` attribute is a URL that designates a resource where run-time values are stored. This allows support tools to identify URLs that are given as parameters.

    - **object**
      
      A value of object indicates that the value of `@valuetype` is an identifier that refers to an object declaration in the document. The identifier must be the value of the ID attribute set for the declared object element.

- **-dita-use-conref-target**
  
  See *Using the -dita-use-conref-target value* (see page 666) for more information.

- **@type**
  
  This attribute specifies for a user agent the type of values that will be found at the URI designated by `@value`. Note that this differs from the `@type` attribute on many other DITA elements. This attribute is defined with the XML Data Type CDATA.

  1. When `@valuetype` is set to "ref", this attribute directly specifies the content type of the resource designated by `@value`.

  2. Otherwise, if `@type` is specified and `@keyref` is specified and resolvable, this attribute specifies the content type of the resource designated by `@keyref`. 
3. Otherwise, if @type is not specified and @keyref is specified and is resolvable, the effective type value specified for the key that is named by the @keyref attribute is used as the value of the @type attribute.

@keyref

Key reference to the thing the parameter references. If @valuetype is specified but is not set to "ref", this attribute is ignored. When @valuetype is not specified and @keyref is specified, it implies a setting of valuetype="ref". When @keyref is specified and the effective value of @valuetype is "ref":

1. When the key specified by @keyref is resolvable and has an associated URI, that URI is used as the value of this element (overriding @value, if that is specified).
2. When the key specified by @keyref is resolvable and has no associated resource (only link text), the <keyref> attribute is considered to be unresolvable for this element. If @value is specified, it is used as fallback.
3. When the key specified by @keyref is not resolvable, the value of the @value attribute is used as a fallback target for the <param> element.

This attribute is defined with the XML Data Type CDATA.

Comment by robander, 12 February 2014
Reminder that when the "keyref replacement text" topic is updated, it should match this (keyref with only link text is meaningless for param).

4.2.2.29 <ph>

The <ph> element is used to enclose a phrase for reuse or conditional processing (for example, when part of a paragraph applies to a particular audience). It can be used by specializations of DITA to create semantic markup for content at the phrase level, which then allows (but does not require) specific processing or formatting.

Inheritance
- topic/ph

Example

This was not changed. <ph rev="v5r2">This was updated.</ph> This was not.

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and @keyref.

4.2.2.30 <pre>

The <pre> element includes preformatted text that must preserve line breaks and spaces entered manually by the author in the content of the element. It typically presents the content in a monospaced type font (depending on your output formatting processor). Do not use <pre> when a more semantically specific element is appropriate, such as <codeblock>.

Inheritance
- topic/pre

Example

The following example will preserve all line breaks.

<p>MEMO: programming team fun day</p>
Remember to bring a kite, softball glove, or other favorite outdoor accessory to tomorrow's fun day outing at Zilker Park. Volunteers needed for the dunking booth.

The rendered result will differ depending on the processor that is rendering your DITA content. It will generally look something like this:

**MEMO: programming team fun day**

Remember to bring a kite, softball glove, or other favorite outdoor accessory to tomorrow’s fun day outing at Zilker Park. Volunteers needed for the dunking booth.

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Display attribute group* (see page 660), *outputclass* (see page 0), *xml:space* (see page 0), and *spectitle* (see page 0).

### 4.2.2.31 `<q>`

The `<q>` element includes content quoted from another source. This element is used for short quotes which are displayed inline. Use the long quote element (`<lq>`) for quotations that should be set off from the surrounding text.

Authors should not add quote punctuation manually when using the `<q>` element. Processors that render the `<q>` element should add appropriate styling, such as locale-specific quotation marks.

**Inheritance**

- topic/q

**Example**

George said, `<q>Disengage the power supply before servicing the unit.</q>`

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

### 4.2.2.32 `<section>`

The `<section>` element represents an organizational division in a topic. Sections are used to organize subsets of information that are directly related to the topic. For example, the titles *Reference Syntax*, *Example*, and *Properties* might represent section-level discourse within a topic about a command-line process—the content in each `<section>` relates uniquely to the subject of that topic. Multiple `<section>` elements within a single topic do not represent a hierarchy, but rather peer divisions of that topic. Sections cannot be nested. A `<section>` may have an optional title.

**Inheritance**

- topic/section

**Example**

```xml
<reference id="reference">
<title>Copy Command</title>
<refbody>
<section>
<title>Purpose</title>
<p>This little command copies</p>
</section>
</refbody>
</reference>
```
Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *suptitle* (see page 0).

4.2.2.33 <sectiondiv>

The `<sectiondiv>` element allows logical grouping of content within a section. There is no additional meaning associated with the `<sectiondiv>` element. The `<sectiondiv>` element does not contain a title; the lowest level of titled content within a topic is the section itself. If additional hierarchy is required, nested topics should be used in place of the section.

The `<sectiondiv>` element nests itself, so it can be specialized to create structured information within sections. Another common use case for the `<sectiondiv>` element is to group a sequence of related elements for reuse, so that another topic may reference the entire set with a single `<refconref>` attribute.

Because the `<bodydiv>` element allows `<section>`, it cannot be used within `<section>` elements. Use the `<div>` element to group content that might occur in both topic bodies and sections.

Inheritance

- `<topic/sectiondiv>`

Example

In the example below, the `<sectiondiv>` element is used to group content that can be reused elsewhere.

```
<section>
  <title>Nice pets</title>
  <sectiondiv id="smallpets">
    <p>Cats are nice.</p>
    <p>Dogs are nice.</p>
    <p>Friends of mine really love their hedgehogs.</p>
  </sectiondiv>
  <sectiondiv id="biggerpets">
    <p>lots of people want ponies when they grow up.</p>
    <p>Llamas are also popular.</p>
  </sectiondiv>
</section>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.2.2.34 <sl>

The `<sl>` element contains a simple list of items of short, phrase-like content, such as a list of materials in a kit or package.

On output, the list should have no bullets, on the assumption that each item is short enough to fit on one line, and needs no additional differentiation from its neighbors.

Inheritance

- `<topic/sl>`
Example

In a reference topic discussing related modules, the following sample markup could be used:

```xml
<section>
<title>Messages</title>
<p>Messages from the ags_open module are identical with messages from:</p>
<sl>
<sli>ags_read</sli>
<sli>ags_write</sli>
<sli>ags_close</sli>
</sl>
</section>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), compact (see page 0), and spectitle (see page 0).

### 4.2.2.35 `<sli>`

The `<sli>` element is an item in a simple list (`<sl>`). Simple list items have phrase or text content, adequate for describing package contents, for example. When a DITA topic is formatted for output, the items of a simple list should be placed each on its own line, with no other prefix such as a number (as in an ordered list) or bullet (as in an unordered list).

**Inheritance**

- topic/sli

**Example**

See `<sli>` (see page 282).

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

### 4.2.2.36 `<term>`

The `<term>` element identifies words that may have or require extended definitions or explanations.

**Inheritance**

- topic/term

**Example**

```xml
<p>The <term>reference implementation</term> of DITA represents the standard, <q>fallback</q> behaviors intended for DITA elements.</p>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and `@keyref`.
4.2.2.37 <text>

The <text> element associates no semantics with its content. It exists to serve as a container for text where a container is needed (for example, for using the @conref attribute, or for restricted content models in specializations).

Comment by Kristen Eberlein, 2 March 2013
A paragraph was deleted from this topic by proposal 13029: Allow <text> in more places.

Unlike <ph>, <text> cannot contain images. Unlike <keyword>, <text> does not imply keyword-like semantics. The <text> element contains only text data, or nested <text> elements. All universal attributes are available on <text>.

Inheritance
- topic/text

Example
<p>This an example of <keyword><text id="reuse">Text that is reusable</text></keyword>, with no extra semantics attached to the text when it is reused.</p>

Attributes
The following attributes are available on this element: Universal attribute group (see page 650).

4.2.2.38 <tm>

The <tm> element in DITA is used to markup and identify a term or phrase that is trademarked. Trademarks include registered trademarks, service marks, slogans and logos.

The business rules for indicating and displaying trademarks may differ from company to company and may be enforced by authoring policy and by specific processing.

Inheritance
- topic/tm

Example
<p>The advantages of using <tm trademark="DB2 Universal Database" tmtype="tm">DB2 Universal Database</tm> are well known.</p>

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and the attributes defined below.

@tmtype (REQUIRED)
- Specifies the trademark type. Allowable values are:
  - tm
    - trademark (tm)
  - reg
    - registered trademark (regtm)
  - service
service mark (service)

**-dita-use-conref-target**

See *Using the -dita-use-conref-target value* (see page 666) for more information.

@trademark
The trademarked term. This attribute is defined with the XML Data Type CDATA.

@tmowner
The trademark owner, for example "OASIS". This attribute is defined with the XML Data Type CDATA.

@tmclass
Classification of the trademark. This may be used to differentiate different groupings of trademarks. This attribute is defined with the XML Data Type CDATA.

### 4.2.2.39 <ul>

The `<ul>` element is a list of items in which the order of list items is not significant. List items are typically styled on output with a "bullet" character, depending on nesting level.

**Inheritance**
- topic/ul

**Example**

```xml
<ul>
  <li>This is an item in an unordered list.</li>
  <li>To separate it from other items in the list, the formatter puts a bullet beside it.</li>
  <li>The following paragraph, contained in the list item element, is part of the list item which contains it.</li>
  <p>This is the contained paragraph.</p>
  <li>This is the last list item in our unordered list.</li>
</ul>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0 ), *compact* (see page 0 ), and *spectitle* (see page 0 ).

### 4.2.2.40 <xref>

Use the `<xref>` element to provide an inline cross reference. It is commonly used to link to a different location within the current topic, a different topic, a specific location in another topic, or an external resource. The target of the cross-reference is specified using the @href or @keyref attributes.

When creating a cross-reference, link to the element structure, not the `<title>` element of the object. For example, to create a cross-reference to a figure, link to the `<fig>` element, not the `<title>` element within the `<fig>` element. Output processing should determine whether the text of the `<title>` element is used to render the cross-reference.

Cross references that link to elements in other topics should employ key-based addressing by using the @keyref attribute. This enables the cross-reference to point to different topics in the context of different top-level maps. Cross references that employ direct URI-based addressing using the @href attribute create dependencies; if the topic with the cross-reference is included in a map, the target topic also must be included or the cross-reference will not be resolvable in the context of that map. While you can use conditional processing to have different cross-references for different contexts, it is usually easier and more effective to use keys. By using keys, the cross-reference can be independent of the contexts it might used in, because it is up to each different map to bind the key used by the cross-reference to the appropriate target.
In many cases it is best to avoid inline cross references in favor of links of links that are generated by the related-links (see page 256) element or relationship tables. Such links typically are rendered at the end of the topic. This prevents readers from leaving a topic and missing important information. Using relationship tables also enables the links to be managed at a DITA map level, making it possible for topics to be quickly integrated into new contexts without breaking links.

**Inheritance**
- topic/xref

**Examples**

This example shows a cross-reference to another topic; that topic’s title will typically be used as the link text.

```xml
<p>Background information about DITA is provided in the topic entitled
<xref href="whatsdita.dita#tmmdita">\</xref>.</p>
```

This example shows a cross-reference to another topic; the supplied text will be used as the link text:

```xml
<p><xref href="whatsdita.dita#tmmdita">Background information about DITA</xref> is provided free of charge.</p>
```

If you are linking to an element inside of a topic, you should use the following format in the @href attribute:

```xml
filename.dita#topicid/elementid
```

If you are linking within the same file, you can leave off the file name portion filename.dita. So, for a section with the ID "mysection", you should use:

```xml
#topicid/mysection
```

Alternatively, if the link is to an element within the same topic you can use:

```xml
./mysection
```

For a list item within that section, assuming the item has an ID of "mylist", use:

```xml
#topicid/mylist
```

Alternatively, because the link is to an element within the same topic, you can use:

```xml
./mylist
```

**Comment by WEK**

Add key-based addressing examples

See Direct DITA addressing (see page 63) for details on using URI references and key references.

If your URL has an ampersand (&) in it, you need to code that using a entity reference. For example, this URL includes an & character:

```xml
http://www.example.com/docview.wss?rs=757&context=SSVNX5
```

When used in an @href attribute, the ampersand must be entered as &amp; as shown here:

```xml
<xref href="http://www.example.com/docview.wss?rs=757&amp;context=SSVNX5" scope="external">Part number SSVNX5</xref>
```
Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661), *outputclass* (see page 0), and *@keyref*.

Related Links

- **3.3.3.3 Links within topics** (see page 62)
  
  A topic may contain several types of links.

- **3.3.4 Direct DITA addressing** (see page 63)
  
  DITA provides two addressing mechanisms. DITA addresses either are direct, URI-based addresses, or they are indirect key-based addresses. Within DITA documents, individual elements are addressed by unique identifiers specified on the *@id* attribute. DITA defines two fragment identifier syntaxes for addressing DITA elements, one for topics and elements within maps and another for non-topic elements within topics.

- **4.2.4.1 link** (see page 297)
  
  The *<link>* element defines a relationship to another topic. Links are typically sorted when displayed based on their attributes, which define the type or role of the link's target in relation to the current topic.

### 4.2.3 Table elements

DITA topics support two types of tables. The *<table>* element uses the OASIS Exchange Table Model (formerly known as the CALS table model). The OASIS table supports the spanning of multiple rows or columns for special layout or organizational needs, and provides a wide variety of controls over the display properties of the data and even the table structure itself.

The other table structure in DITA is called *<simpletable>*. As the name implies, it is structurally less complex than the OASIS table, and can be used as a very simple, regular table for which close control of formatting is not as important. The main advantage of simpletable is for describing lists of data with regular headings, such as telephone directory listings, display adapter configuration data, or API properties.

#### 4.2.3.1 <table>

The *<table>* element organizes arbitrarily complex relationships of tabular information. This standard table markup allows column or row spanning and table captions or descriptions. An optional title allowed inside the *<table>* element provides a caption to describe the table.

The DITA table is based on the OASIS Exchange Table Model, augmented with DITA attributes that enable it for specialization, conref, and other DITA processing. In addition, the table includes a *<desc>* element, which enables table description that is parallel with figure description. See *simpletable* (see page 295) for a simplified table model that can be specialized to represent more regular relationships of data.

In DITA tables, in place of the *@expanse* attribute used by other DITA elements, the *@pgwide* attribute is used in order to conform to the OASIS Exchange Table Model. The *@pgwide* attribute has a similar semantic (1=page width; 0=resize to galley or column).

**Note:** The *@scale* attribute represents a stylistic markup property that is currently maintained in tables for legacy purposes. External stylesheets should enable less dependency on this attribute. Use the *@scale* attribute judiciously.

Inheritance

- topic/table

Example

```markdown
<table>
  <tgroup cols="2">
    <colspec colname="COLSPEC0" colwidth="121*/">
    <colspec colname="COLSPEC1" colwidth="76*/">
    <thead>
    <row>
```
<table>
<thead>
<tr>
<th>Animal</th>
<th>Gestation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elephant (African and Asian)</td>
<td>19-22 months</td>
</tr>
<tr>
<td>Giraffe</td>
<td>15 months</td>
</tr>
<tr>
<td>Rhinoceros</td>
<td>14-16 months</td>
</tr>
<tr>
<td>Hippopotamus</td>
<td>7 1/2 months</td>
</tr>
</tbody>
</table>

In this example, the use of the `<thead>` element for the header allows processors or screen readers to identify a header relationship between any cell in the table body and the matching header cell above that column.

Example: complex table with implied accessibility markup

In the following example, the table uses `<thead>` to identify header rows and `@rowheader` to identify a header column. This header relationship can be used to automatically create renderings of the table in other formats, such as HTML, that can be navigated using a screen reader or other assistive technology.
In this sample, navigation information for assistive technology is derived from two sources:

- The `<thead>` element contains two rows, and indicates that each `<entry>` in those rows is a header cell for that column. This means that each body cell can be associated with the header cell or cells above the column. For example, in the second body row, the entry "Peter" can be associated with the header "Name"; similarly, the entry "9,000" can be associated with the headers "expected" and "points".
- The `@rowheader` attribute implies that the first column plays a similar role as a header. This means that each body cell in columns two and three can be associated with the header cell in column one. For example, in the second body row, the entry "9,000" can be associated with the header "Peter".

As a result of these two sets of headers, a rendering of the table can associate the entry "9,000" with three headers: "Peter", "expected", and "points", making it fully navigable by a screen reader or other assistive technology.

The formatted output might be displayed as follows:

### Table 11: Sample of automated table accessibility

Names are listed in the column c1. Points are listed in both data columns, with expected points in column c2 and actual points in column c3.

<table>
<thead>
<tr>
<th>Name</th>
<th>points</th>
<th>expected</th>
<th>actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mark</td>
<td>10,000</td>
<td>11,123.45</td>
<td></td>
</tr>
<tr>
<td>Peter</td>
<td>9,000</td>
<td>11,012.34</td>
<td></td>
</tr>
<tr>
<td>Cindy</td>
<td>10,000</td>
<td>10,987.64</td>
<td></td>
</tr>
</tbody>
</table>

### Complex table with some manually specified accessibility markup

In some complex tables, the `<thead>` element and `@rowheader` attribute may not be enough to support all accessibility needs. Assume that the table above is flipped so that the names are listed across the top row, instead of in the first column, as follows:

**Table 12: Sample with two header columns**

<table>
<thead>
<tr>
<th>Name</th>
<th>Mark</th>
<th>Peter</th>
<th>Cindy</th>
</tr>
</thead>
<tbody>
<tr>
<td>points</td>
<td>expected</td>
<td>10,000</td>
<td>9,000</td>
</tr>
<tr>
<td></td>
<td>actual</td>
<td>11,123.45</td>
<td>11,012.34</td>
</tr>
</tbody>
</table>

In this case the `@rowheader` attribute cannot be used, because it is only able to specify the first column as a header column. In this case, the `@scope` attribute can be used to indicate that entries in the first and second columns function as headers for the entire row (or row group, in the case of a cell that spans more than one
row). The following code sample demonstrates the use of @scope to facilitate navigation of these rows by a screen reader or other assistive technology; note that the <thead> element is still used to imply a header relationship with the names at the top of each column.

```xml
<table frame="all"
<title>Sample with two header columns</title>
<tgroup cols="5">
<colspec colname="c1"/>
<colspec colname="c2"/>
<colspec colname="c3"/>
<colspec colname="c4"/>
<colspec colname="c5"/>
<thead>
  <row>
    <entry namest="c1" nameend="c2">Name</entry>
    <entry>Mark</entry>
    <entry>Peter</entry>
    <entry>Cindy</entry>
  </row>
</thead>
<tbody>
  <row>
    <entry morerows="1" scope="rowgroup">points</entry>
    <entry scope="row">expected</entry>
    <entry>10,000</entry>
    <entry>9,000</entry>
    <entry>10,000</entry>
  </row>
  <row>
    <entry scope="row">actual</entry>
    <entry>11,123.45</entry>
    <entry>11,012.34</entry>
    <entry>10,987.64</entry>
  </row>
</tbody>
</tgroup>
</table>
```

Example: complex table with manual accessibility markup

In extremely complex tables, such as those with a single header cell in the middle of the table, extremely fine grained accessibility controls are available to explicitly associate any content cell with any header cell. This may also be useful for cases where processors do not support the implied accessibility relationships described above.

In the following sample, header cells are identified using the @id attribute, which is referenced using the @headers attribute on appropriate content cells. This makes all header relationships in the table explicit. Note that this sample ignores the @scope attribute, which could be used to exercise manual control without setting as many attribute values; it also ignores the fact that <thead> creates a header relationship even when the @id and @headers attributes are not used.

```xml
<table frame="all"
<title>Sample with fully manual accessibility control</title>
<desc>Names are listed in the column c1. Points are listed in both data columns, with expected points in column c2 and actual points in column c3.</desc>
<tgroup cols="3">
  <colspec colname="c1"/>
  <colspec colname="c2"/>
  <colspec colname="c3"/>
  <thead>
    <row>
      <entry morerows="1">points</entry>
      <entry namest="c2" nameend="c3" id="pts">points</entry>
    </row>
    <row>
      <entry id="exp" headers="pts">expected</entry>
      <entry id="act" headers="pts">actual</entry>
    </row>
    </thead>
    </tbody>
    </row>
    <entry id="name1">Mark</entry>
```
Figure 58: DITA source

The formatted output might be displayed as follows:

Table 13: Sample with fully manual accessibility control

Names are listed in the column c1. Points are listed in both data columns, with expected points in column c2 and actual points in column c3.

<table>
<thead>
<tr>
<th></th>
<th>points</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>expected</td>
<td>actual</td>
</tr>
<tr>
<td>Mark</td>
<td>10,000</td>
<td>11,123.45</td>
</tr>
<tr>
<td>Peter</td>
<td>9,000</td>
<td>11,012.34</td>
</tr>
<tr>
<td>Cindy</td>
<td>10,000</td>
<td>10,987.64</td>
</tr>
</tbody>
</table>

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), @display and @scale from Display attribute group (see page 660), and the attributes defined below. This element also uses @colsep, @rowsep, and @rowheader from Complex-table attribute group (see page 657).

@orient

Specifies the orientation of the table in page-based outputs. This attribute is primarily useful for print-oriented display. Allowable values are:

port

The same orientation as the text flow.

land

90 degrees counterclockwise from the text flow.

dita-use-conref-target

See Using the -dita-use-conref-target value (see page 666) for more information.

In situations where a stylesheet or other formatting mechanism specifies table orientation based on other criteria, or for non-paginated outputs, the @orient attribute can be ignored.

@pgwide

Determines the horizontal placement of the element. Supported values are 1 and 0, although these are not mandated by the DTD or Schema. This attribute is defined with the XML Data Type NMTOKEN.

For print-oriented display, the value "1" places the element on the left page margin; "0" aligns the element with the left margin of the current text line and takes indentation into account.
For XHTML, the table surrounds the table data. Either value sets the table width to 100%.

4.2.3.2 <tgroup>

The <tgroup> element in a table contains the header and body rows of a table.

Inheritance
- topic/tgroup

Example
See table (see page 287).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and the attribute defined below. This element also uses @colsep, @rowsep, and @align from Complex-table attribute group (see page 657).

@cols (REQUIRED)
Indicates the number of columns in a <tgroup>. This attribute is defined with the XML Data Type NM_TOKEN.

4.2.3.3 <colspec>

The <colspec> element contains a column specification for a table, including assigning a column name and number, cell content alignment, and column width.

Inheritance
- topic/colspec

Example
See table (see page 287).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) (without the Metadata attribute group), @base from the Metadata attribute group (see page 0), outputclass (see page 0), and the attributes defined below. This element also uses @align, @char, @charoff, @colsep, @rowsep, and @rowheader from Complex-table attribute group (see page 657).

@colnum
Indicates the number of a column in the table, counting from the first logical column to the last column. This attribute is defined with the XML Data Type NM_TOKEN.

@colname
Specifies a name for the column defined by this element. The <entry> element may use @colname to refer to the name of this column. This attribute is defined with the XML Data Type NM_TOKEN.

Comment by robander
cleaned up for 1.3, in 1.2 it used the same definition as for entry: "Specifies the table column name in which an entry is found"

@colwidth
Describes the column width. This attribute is defined with the XML Data Type CDATA.
4.2.3.4 <thead>

The <thead> element is a table header that precedes the table body (<tbody>) element in a complex table.

Inheritance

- topic/thead

Example

See table (see page 287).

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and @valign from Complex-table attribute group (see page 657).

4.2.3.5 <tbody>

The <tbody> element contains the rows in a table.

Inheritance

- topic/tbody

Example

See table (see page 287).

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and @valign from Complex-table attribute group (see page 657).

4.2.3.6 <row>

The <row> element contains a single row in a table.

Inheritance

- topic/row

Example

See table (see page 287).

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0 ). This element also uses @rowsep and @valign from Complex-table attribute group (see page 657).

4.2.3.7 <entry>

The <entry> element defines a single cell in a table.

Inheritance

- topic/entry
Example

See table (see page 287).

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (without the Metadata attribute group), @base and @rev from the Metadata attribute group (see page 651), outputclass (see page 0 ), and the attributes defined below. This element also uses @align, @char, @charoff, @colsep, @rowsep, and @valign from the Complex-table attribute group (see page 657).

@rotate

Indicates whether the contents of the entry should be rotated. While the attribute is declared with the XML data type CDATA, the only predefined values are:

1

The contents of the cell are rotated 90 degrees counterclockwise.

0

No rotation should occur.

@dita-use-conref-target

See Using the -dita-use-conref-target value (see page 666) for more information.

If this attribute is not specified, no rotation should occur. In situations where a stylesheet or other formatting mechanism specifies table cell orientation, the @rotate attribute can be ignored.

@colname

Specifies the column name in which an entry is found. The value is a reference to the @colname attribute on the <colspec> element. This attribute is defined with the XML Data Type NMTOKEN.

@namest

Specifies the first logical column that is included in a horizontal span. The value is a reference to the @colname attribute on the <colspec> element. This attribute is defined with the XML Data Type NMTOKEN.

@nameend

Specifies the last logical column that is included in a horizontal span. The value is a reference to the @colname attribute on the <colspec> element. This attribute is defined with the XML Data Type NMTOKEN.

@morerows

Specifies the number of additional rows to add in a vertical span. This attribute is defined with the XML Data Type NMTOKEN.

@scope

The presence of the @scope attribute indicates that the current entry is a header for the specified scope.

Allowable values are:

row

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

col

The current entry is a header for all cells in the column.

rowgroup

The current entry is a header for all cells in the rows spanned by this entry.

colgroup

The current entry is a header for all cells in the columns spanned by this entry.

@headers

Identifies one or more entry element headers that apply to its entry. 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. This attribute is defined with the XML Data Type NMTOKEN.
4.2.3.8 <simpletable>

The `<simpletable>` element is used for tables that are regular in structure and do not need a caption. Choose the `<simpletable>` element when you want to show information in regular rows and columns. For example, multi-column tabular data such as phone directory listings or parts lists are good candidates for `<simpletable>`. Another good use of `<simpletable>` is for information that seems to beg for a three-part definition list; the @keycol attribute may be used to indicate which column represents the "key" or term-like column of your structure.

This close match of `<simpletable>` to tabular, regular data makes `<simpletable>` suitable as the basis for specialized structures such as `<properties>` (for programming information) and choice tables (for tasks).

Inheritance
- topic/simpletable

Example

Source:

```xml
<simpletable>
  <sthead>
    <stentry>Type style</stentry>
    <stentry>Elements used</stentry>
  </sthead>
  <strow>
    <stentry>Bold</stentry>
    <stentry>b</stentry>
  </strow>
  <strow>
    <stentry>Italic</stentry>
    <stentry>i</stentry>
  </strow>
  <strow>
    <stentry>Underlined</stentry>
    <stentry>u</stentry>
  </strow>
</simpletable>
```

Formatted output:

<table>
<thead>
<tr>
<th>Type style</th>
<th>Elements used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bold</td>
<td>b</td>
</tr>
<tr>
<td>Italic</td>
<td>i</td>
</tr>
<tr>
<td>Underlined</td>
<td>u</td>
</tr>
</tbody>
</table>

Example using @keycol

In this sample, the first column is identified as a header column through the use of `keycol="1"` on the `<simpletable>` element. This indicates that items in the first column should be treated as headers for the row that follows. Rendering of the header column is left up to the implementation.

Source:

```xml
<simpletable keycol="1">
  <sthead>
    <stentry>Term</stentry>
    <stentry>Categorization</stentry>
    <stentry>Definition</stentry>
  </sthead>
  <strow>
    <stentry>Definition</stentry>
  </strow>
</simpletable>
```
### Formatted output:

<table>
<thead>
<tr>
<th>Term</th>
<th>Categorization</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Widget</td>
<td>noun</td>
<td>Thing that is used for something</td>
</tr>
<tr>
<td>Frustration</td>
<td>noun</td>
<td>What you feel when you drop the widget</td>
</tr>
</tbody>
</table>

### Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Display attribute group (see page 660), Simpletable attribute group (see page 662), outputclass (see page 0), and spectitle (see page 0).

#### 4.2.3.9 <sthead>

The `<sthead>` element contains an optional header row for a `<simpletable>` element.

**Inheritance**

- topic/sthead

**Example**

See simpletable (see page 295).

**Attributes**

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

#### 4.2.3.10 <strow>

The `<strow>` element contains a single row inside of a `<simpletable>` element.

**Inheritance**

- topic/strow

**Example**

See simpletable (see page 295).

**Attributes**

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).
4.2.3.11 <stentry>

The <stentry> element represents a single cell within a <simpletable> element. You may place any number of <stentry> cells in either an <sthead> element (for headings) or <strow> element (for rows of data).

Inheritance
- topic/stentry

Example
See simpletable (see page 295).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and specentry (see page 0).

4.2.4 Related links elements

The related-links section of DITA topics is a special structure that contains links. Links support navigation from a topic to other related topics or resources.

Links are different from cross-references. While cross-references occur only within the body of a topic and can target any element in this or other topics, links only represent topic-to-topic connections, or connections to non-DITA-topic resources. Links are located after the body of a topic, in the related-links (see page 256) element.

Links can also be managed indirectly using DITA maps, which provide a more efficient way to manage links and avoids embedded pointers in each topic. This helps keep topics free from specific contexts, and makes it easier to reuse those topics in new locations.

4.2.4.1 <link>

The <link> element defines a relationship to another topic. Links are typically sorted when displayed based on their attributes, which define the type or role of the link’s target in relation to the current topic.

The optional container elements for link (<linkpool> and <linklist>) allow authors to define groups with common attributes or to preserve the authored sequence of links on output. Links placed in a <linkpool> may be rearranged or removed for display purposes (combined with other local or map-based links); links in a <linklist> should be displayed in the order they are defined. Refer to those elements for additional explanation.

Inheritance
- topic/link

Example

```
<related-links>
   <linkpool type="concept">
      <link href="czes.dita#czes" role="next"/>
      <link href="czunder.dita"/>
      <link format="html" href="czover.htm#sqljsupp" role="parent">
         <linktext>Overview of the CZ</linktext>
      </link>
      <link format="html" href="czesqlj.htm#sqljemb">
         <linktext>Working with CZESQLJ</linktext>
         <desc>When you work with CZESQLJ, you need to know...</desc>
      </link>
   </linkpool>
<related-links>
```
Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661), outputclass (see page 0), @keyref, The role and other role attributes (see page 677), and query (DEPRECATED) (see page 0).

Related Links

3.3.3.3 Links within topics (see page 62)
A topic may contain several types of links.

3.3.4 Direct DITA addressing (see page 63)
DITA provides two addressing mechanisms. DITA addresses either are direct, URI-based addresses, or they are indirect key-based addresses. Within DITA documents, individual elements are addressed by unique identifiers specified on the @id attribute. DITA defines two fragment identifier syntaxes for addressing DITA elements, one for topics and elements within maps and another for non-topic elements within topics.

4.2.2.40 xref (see page 285)
Use the <xref> element to provide an inline cross reference. It is commonly used to link to a different location within the current topic, a different topic, a specific location in another topic, or an external resource. The target of the cross-reference is specified using the @href or @keyref attributes.

4.2.4.2 <linklist>

The <linklist> element defines an author-arranged group of links. When rendering the links, processors should preserve the order of links specified within a <linklist> element.

There are two ways to organize related information links within a topic. First, you can add them all in no particular order, either by using <linkpool> elements or by placing <link> elements directly within <related-links>, in which case the rendering is implementation dependent. For example, tools may choose to sort all links based on the role or type; tools may also move or remove links to fit the context (for example, moving a prerequisite link to the top of a browser window, or removing links to the next topic if it is rendered on the same page in a PDF). These behaviors are examples only and are not required.

Second, links may be grouped using one or more <linklist> elements. When you group them using <linklist>, then the order of the links within each <linklist> is preserved when rendered. You may also use a combination of the two approaches, which will allow some links to be automatically sorted while the others are left as-is.

Attributes set on the <linkpool> and <linklist> elements are inherited by their descendants. For example, if you have a <linklist> element that contains all external links, you can set scope="external" on that outer <linklist> element and leave it off the <link> elements within that <linklist>.

Inheritance

- topic/linklist

Example

```dita
<related-links>
    <linklist scope="external">
        <title>Example links</title>
        <desc>These links will always appear in this order.</desc>
        <link href="http://www.example.org">
            <linktext>Example 1</linktext>
        </link>
        <link href="http://www.example.com">
            <linktext>Example 2</linktext>
        </link>
    </linklist>
</related-links>
```
Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), collection-type (see page 0), The role and otherrole attributes (see page 677), spectitle (see page 0), mapkeyref (see page 0), and the attributes defined below. This element also uses @format, @scope, and @type from Link relationship attribute group (see page 661).

Comment by robander

Need to resolve issue with @collection-type: https://lists.oasis-open.org/archives/dita/201312/msg00101.html

Update: Discussed at TC Jan 14, 2014, agreed to keep the "tree" value in the DTD, consider it deprecated, do not add to XSD or RNG

@duplicates

Specifies whether or not duplicate links will be filtered out of a group of links. Allowable values are:

yes

Allow duplicate links

no

Filter out duplicate links

-dita-use-conref-target

See Using the -dita-use-conref-target value (see page 666) for more information.

Comment by robander

This is strongly related to processing and should probably be cleaned up. In 1.2, after giving the yes/no values, the spec had this text: "In general, duplicate links in linklists are preserved.. Note that links are regarded as duplicates only if their content plus all attributes match. ". Additionally in 1.2, the "Default value" column had this confusing text: "The attribute value is currently ignored, but should default to yes for links in linklists and no for all other links." The spec should not state that a value is "currently ignored". I've reworked those two statements into the paragraph here, retaining the original meaning, but I'm still uncertain that we should go this far.

@collection-type

See collection-type (see page 0) for a full definition and list of supported values.

In the initial DTD implementation of DITA, this attribute was defined with an additional value of "tree"; that value was only defined for @collection-type on the <linkpool> and <linklist> elements. The "tree" value is not allowed on @collection-type when used in maps, and is not defined in the XSD or RELAX NG versions of <linkpool> or <linklist>. The extra value in the DTD implementation is retained for backwards compatibility, but is deprecated.

4.2.4.3 <linkpool>

The <linkpool> element defines a group of links that have common characteristics, such as type or audience or source. When links are in <related-links> or <linkpool> elements, the organization of links on final output is determined by the output process, not by the order that the links actually occur in the DITA topic.

There are two ways to organize related information links within a topic. First, you can add them all in no particular order, either by using <linkpool> elements or by placing <link> elements directly within <related-links>, in which case the rendering is implementation dependent. For example, tools may choose to sort all links based on the role or type; tools may also move or remove links to fit the context (for example, moving a prerequisite link to the top of a browser window, or removing links to the next topic if it is rendered on the same page in a PDF). These behaviors are examples only and are not required.

Second, links may be grouped using one or more <linklist> elements. When you group them using <linklist>, then the order of the links within each <linklist> is preserved when rendered. You may also use
a combination of the two approaches, which will allow some links to be automatically sorted while the others are left as-is.

Attributes set on the `<linkpool>` and `<linklist>` elements are inherited by their descendants. For example, if you have a `<linklist>` element that contains all external links, you can set `scope="external"` on that outer `<linklist>` element and leave it off the `<link>` elements within that `<linklist>`.

**Inheritance**

- `topic/linkpool`

**Example**

```xml
<related-links>
  <linkpool type="concept">
    <link href="czez.dita#czez" role="next"></link>
    <link href="czunder.dita"></link>
    <link format="html" href="czover.htm#sqljsupp" role="parent">
      <linktext>Overview of the CZ</linktext>
    </link>
    <link format="html" href="czeqlj.htm#sqljemb">
      <linktext>Working with CZESQLJ</linktext>
    </desc>
    When you work with CZESQLJ, you need to know...
  </linkpool>
</related-links>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), `outputclass` (see page 0), `collection-type` (see page 0), *The role and otherrole attributes* (see page 677), `mapkeyref` (see page 0), and the attributes defined below. This element also uses `@format`, `@scope`, and `@type` from *Link relationship attribute group* (see page 661).

**Comment by robander**

Need to resolve issue with `@collection-type`: https://lists.oasis-open.org/archives/dita/201312/msg00101.html

Update: Discussed at TC Jan 14, 2014, agreed to keep the "tree" value in the DTD, consider it deprecated, do not add to XSD or RNG

`@duplicates`

Specifies whether or not duplicate links will be filtered out of a group of links. Allowable values are:

- `yes`
  - Allow duplicate links
- `no`
  - Filter out duplicate links

`-dita-use-conref-target`

See *Using the -dita-use-conref-target value* (see page 666) for more information.

Note that links are regarded as duplicates only if their content plus all attributes match. The processing default should be "yes" within `<linklist>` elements and "no" for other links.

**Comment by robander**

This is strongly related to processing and should probably be cleaned up. In 1.2, after giving the yes/no values, the spec had this text: "In general, duplicate links in linklists are preserved.. Note that links are regarded as duplicates only if their content plus all attributes match." Additionally in 1.2, the "Default value" column had this confusing text: "The attribute value is currently ignored, but should default to yes for links in linklists and no for all other links." The spec should not state that a value is "currently ignored". I've reworked those two statements into the paragraph here, retaining the original meaning, but I'm still uncertain that we should go this far.
@collection-type

See collection-type (see page 0 ) for a full definition and list of supported values.

In the initial DTD implementation of DITA, this attribute was defined with an additional value of "tree"; that value was only defined for @collection-type on the <linkpool> and <linklist> elements. The "tree" value is not allowed on @collection-type when used in maps, and is not defined in the XSD or RELAX NG versions of <linkpool> or <linklist>. The extra value in the DTD implementation is retained for backwards compatibility, but is deprecated.

4.2.4.4 <linktext>

The <linktext> element provides the literal label or line of text for a link. For links to local DITA topics, the text of a link can typically be resolved during processing. Use the <linktext> element only when the target cannot be reached, such as when it is a peer or external link, or when the target is local but not in DITA format. When used inside a topic, <linktext> is used as the text for the specified link; when used within a map, <linktext> is used as the text for generated links that point to the specified topic.

**Inheritance**

"- topic/linktext " when used in topics, and "- map/linktext " when used in maps.

**Example**

```xml
<link href="tzover.htm#acssqlj">
  <linktext>Accessing relational data with SQLJ</linktext>
</link>
```

**Attributes**
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0 ).

4.2.4.5 <linkinfo>

The <linkinfo> element allows you to place a descriptive paragraph after the links that are contained in a <linklist> element.

**Inheritance**

- topic/linkinfo

**Example**

```xml
<linklist>
  <title>Repairing widgets</title>
  <link href="debug.dita" type="task"></link>
  <link href="repair.dita" type="task"></link>
  <link href="test.dita" type="task"></link>
  <linkinfo>To repair a reciprocating widget, you must follow the instructions very carefully. Note the sequence to follow. Do it.</linkinfo>
</linklist>
```

**Attributes**
The following attributes are available on this element: Universal attribute group (see page 650).
4.3 Map elements

Map elements include the core components of DITA maps, such as `<topicref>` and `<reltable>`, as well as general purpose map specializations in the map group domain.

4.3.1 Basic map elements

DITA maps are built from a few core elements that are used for referencing and organizing topics. The `<topicmeta>` element is also available to specify metadata for the map, for individual topics, or for groups of topics. Many elements inside `<topicmeta>` are also available inside the topic prolog.

4.3.1.1 `<map>`

The `<map>` element describes the relationships among a set of resources, such as DITA topics. Maps consist of references to topics, maps, and other resources organized into hierarchies, groups, and tables. Maps express these relationships in a single common format that can be used for different outputs.

The containing element for a map is the `<map>` element. Within the map, use the `<topicref>` element to add and organize references to the topics, and the `<topicgroup>` and `<reltable>` elements to provide non-hierarchical relationships. You can use the `<map>` element to set default attribute values for all `<topicref>` elements in the map.

A map describes the relationships among a set of DITA topics. The following are some examples of relationships that can be described in a map:

- Hierarchical (Parent/Child). Nested topics create a hierarchical relationship. The topic that does the nesting is the parent, and the topics that are nested are the children.
- Ordered. Child topics can be labeled as having an ordered relationship, which means they are referenced in a definite sequence.
- Family. Child topics can be labeled as having a family relationship, which means they all refer to each other.

When rendering a map, processors may make use of these relationships, such as to create a Table of Contents (TOC), aggregate topics into a PDF document, or create links between topics in output.

The `<title>` element may optionally be used to provide a title for the map (the `<title>` element is preferred over the `@title` attribute). In some scenarios the title is purely informational, and is present as an aid to the author. In other scenarios it may be useful or even required. For example, if a map is converted to Eclipse Help, the Eclipse system will require a title for the resulting table of contents. In the `<bookmap>` specialization of `<map>`, the `<title>` element provides a title for the book represented by that map.

Inheritance
- map/map

Example

In this example, there are six `<topicref>` elements. They are nested and have a hierarchical relationship. The file `bats.dita` is the parent topic and the other topics are its children. The hierarchy could be used to generate a PDF, a navigation pane in an information center, a summary of the topics, or related links between the parent topic and its children.

```xml
<map id="mybats">
  <title>Bats</title>
  <topicref href="bats.dita" type="topic">
    <topicref href="batcaring.dita" type="task"></topicref>
    <topicref href="batfeeding.dita" type="task"></topicref>
    <topicref href="batsonar.dita" type="concept"></topicref>
    <topicref href="batguano.dita" type="reference"></topicref>
    <topicref href="bathistory.dita" type="reference"></topicref>
  </topicref>
</map>
```
Attributes

Comment by robander
In 1.2 there was a unique definition of @conref on <map>: "This attribute is used to reference an ID on a map that can be reused. See The conref attribute (see page 666) for examples and details about the syntax.". Using the common definition now, not sure we need a unique one?

Update: Discussed at TC Jan 14, 2014, agreed to use the common set of conref attributes, so keeping the version above.

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below), Attributes common to many map elements (see page 655), Architectural attribute group (see page 654), outputclass (see page 0 ), and the attributes defined below. This element also uses @type, @scope, and @format from Link relationship attribute group (see page 661).

@id
Allows an ID to be specified for the map. Note that maps do not require IDs (unlike topics), and the map ID is not included in references to elements within a map. This attribute is defined with the XML Data Type ID.

@anchorref
Identifies a location within another map file where this map will be anchored at runtime. Resolution of the map is deferred until the final step in the delivery of any rendered content. For example, anchorref="map1.ditamap/a1" causes this map to be pulled into the location of the anchor point "a1" inside map1.ditamap when map1.ditamap is rendered for delivery. This attribute is defined with the XML Data Type CDATA.

@title
An identifying title for the map. May be used or ignored, depending on the capabilities of the display mechanism. Note that beginning with DITA 1.1, the map can include a <title> element, which is preferred over the @title attribute. This attribute is defined with the XML Data Type CDATA.

4.3.1.2 <topicref>

The <topicref> element identifies a topic (such as a concept, task, or reference) or other resource. A <topicref> can contain other <topicref> elements, allowing you to express navigation or table-of-contents hierarchies, as well as implying relationships between a containing (parent) <topicref> and its children. You can set the collection-type of a parent <topicref> to determine how its children are related to each other. You can also express relationships among <topicref> elements by using group and table structures (such as <topicgroup> and <reltable>). Relationships are expressed as links in the output; by default, each participant in a relationship has links to the other participants in that relationship.

You can fine tune the output from your map by setting different attributes on the <topicref> element. For example, the @linking attribute controls how a topic's relationships to other topics are expressed as links, and the @toc attribute controls whether the topic shows up in TOC or navigation output.

Inheritance
- map/topicref

Example

In this example, there are six <topicref> elements. They are nested and have a hierarchical relationship.

bats.dita is the parent topic and the other topics are its children.

```xml
<map title="Bats">
  <topicref href="bats.dita" type="topic">
    <topicref href="batcaring.dita" type="task"></topicref>
    <topicref href="batfeeding.dita" type="task"></topicref>
    <topicref href="batsonar.dita" type="concept"></topicref>
    <topicref href="batguano.dita" type="reference"></topicref>
  </topicref>
</map>
```
Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), Attributes common to many map elements (see page 655), Topicref element attributes group (see page 663), outputclass (see page 0 ), @keys, and @keyref.

@href

A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

4.3.1.3 <topicmeta>

The <topicmeta> element defines the metadata that applies to a topic when it appears in a map. When appropriate, that metadata also applies to the other topics in the map that are contained by the same element that contains the <topicmeta> element. When creating links, <topicmeta> content can also be used to override the title and short description that are used for the link. In addition, it can be used to add index entries to referenced content using the <keywords> element.

The metadata given in a <topicmeta> element is specific to a given context within a map. If a reference to a single resource appears more than once in a map or set of maps, unique metadata may be specified in each instance. For example, the two references to a single resource may specify different navigation titles or search titles, each of which is specific to a single context.

Note: The topic Cascading of attributes and metadata in a DITA map (see page 48) in the DITA Architectural Specification provides more information about which metadata elements inside <topicmeta> cascade to other <topicref> elements. In addition, the topic Reconciling topic and map metadata (see page 54) provides more information about how metadata specified in the map interacts with metadata specified in each topic.

Inheritance
- map/topicmeta

Example

In this example, the metadata defined by the <topicmeta> element applies to the associated <topicref> (bats.dita) and all of its children. The <topicmeta> element contains an audience definition which indicates that bats.dita and its children are of interest to experienced programmers who are troubleshooting.

```xml
<map>
  <topicref href="bats.dita">
    <topicmeta>
      <audience type="programmer" job="troubleshooting" experiencelevel="expert"/>
    </topicmeta>
  </topicref>
  <topicref href="batcaring.dita"></topicref>
  <topicref href="batfeeding.dita"></topicref>
</map>
```
**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and the attribute defined below.

@lockmeta

Indicates whether the metadata information should be replaced by the metadata information in the referenced topic. Allowable values are:

**yes**

The information inside `<topicmeta>` should not be replaced with information from the topic.

**no**

The information inside `<topicmeta>` may be replaced with information from the topic.

-dita-use-conref-target

See *Using the -dita-use-conref-target value* (see page 666) for more information.

### 4.3.1.4 `<anchor>`

The `<anchor>` element provides an integration point that another map can reference in order to insert its navigation into the current navigation tree. For those familiar with Eclipse help systems, this serves the same purpose as the `<navref>` element in that system. It might not be supported for all output formats.

The `<anchor>` element is typically used to allow integration of run-time components. For build-time integration, you can instead use the @conref or @conkeyref attribute on an element inside the map. For example, a `<topicref>` element may use @conref to pull in content at build-time from a `<topicref>` in another map.

**Inheritance**

- map/anchor

**Example**

In this example, the `map1.ditamap` contains an `<anchor>` element with an @id attribute set to "a1".

```xml
<map>
  <topicref navtitle="Start here" href="start.dita" toc="yes">
    <navref mapref="othermap2.ditamap"/>
    <navref mapref="othermap3.ditamap"/>
    <anchor id="a1"/>
  </topicref>
</map>
```

**Figure 59: DITA map that contains an anchor**

The @id on an `<anchor>` element can be referenced by the @anchorref attribute on another map's `<map>` element. For example, the map to be integrated at that spot could be defined as follows.

```xml
<map anchorref="map1.ditamap#a1">
  <title>This map is pulled into the MyComponent task map</title>
  <!-- ... -->
</map>
```

**Figure 60: DITA map that references an anchor**

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of @id, given below) and *outputclass* (see page 0).

@id (REQUIRED)
Provides an integration point that another map may reference in order to insert its navigation into the current navigation tree. The @anchorref attribute on a map may be used to reference this attribute. See ID attribute (see page 63) for more details. This attribute is defined with the XML Data Type NMTOKEN.

Comment by robander
This assumes TC agrees that the attribute should use full conref set, rather than just @conref as specified in 1.2
Update: TC agreed on Jan 14, 2014

4.3.1.5 <navref>

The <navref> element represents a pointer to another map which should be preserved as a transcluding link rather than resolved. Output formats that support such linking will integrate the referenced resource when displaying the referencing map to an end user.

For example, if a map is converted to the Eclipse help system format, the DITA element <navref mapref="other.ditamap"/> should be converted to the Eclipse element <link toc="other.xml"/>. When Eclipse loads the referencing map, it will replace this link element with the contents of the file other.xml, provided that the file other.xml is available.

Note that not all output formats support such linking. In order to include another map directly without depending on the output format, use a <topicref> element with the @format attribute set to "ditamap". The effect is similar to using a @conref attribute. For example, the following markup represents a literal inclusion of the map other.ditamap:

```xml
<topicref href="other.ditamap" format="ditamap"/>
```

Inheritance
- map/navref

Example
In this example, the map titled "MyComponent tasks" references the maps othermap2.ditamap and othermap3.ditamap.

```xml
<map title="MyComponent tasks">
  <navref mapref="/..\com.ibm.xml.doc\othermap1.ditamap"/>
  <navref mapref="/..\com.ibm.xml.doc\othermap2.ditamap"/>
</map>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and the attribute defined below.

@mapref
Specifies the URL (local filename, at least) of the map file to reference. It may point to a DITA map, or to a file that is appropriate for your output format (such as XML TOC file for Eclipse output). This attribute is defined with the XML Data Type CDATA.

4.3.1.6 <reltable>

The <reltable> element is a relationship table that specifies relationships between topics, based on the familiar table model of rows (<relrow>), columns (<relheader>), and cells (<relcell>). A frequently-used type of relationship table establishes relationships between task, concept, and reference topics. Each column in a relationship table typically represents a specific role in a set of relationships; for
example, the first column often contains references to tasks, while the second and third columns often reference concept and reference topics. The relationship table rows define relationships between the resources referenced in different cells of the same row; in this example, each row establishes relationships between tasks and the concept and reference topics that support the tasks. When used in this manner, relationship tables make it easy to determine where related information is missing or undefined.

By default, the contents of a <reltable> element are not output for navigation or TOC purposes; they are used only to define relationships that can be expressed as topic-to-topic links. The <relcell> elements can contain <topicref> elements, which are then related to other <topicref> elements in the same row (although not necessarily in the same cell).

Relationship tables can be used in conjunction with hierarchies and groups to manage all the related links in an information set.

**Inheritance**

- map/reltable

**Example**

In this example, a relationship table is defined with three columns; one for "concept", one for "task", and one for "reference". Three cells are defined within one row. The first cell contains one concept topic: batsonar.dita. The second cell contains two task topics: batcaring.dita and batfeeding.dita. The third cell contains two reference topics: batguano.dita and bathistory.dita.

```
<map>
  <reltable>
    <relheader>
      <relcolspec type="concept"/>
      <relcolspec type="task"/>
      <relcolspec type="reference"/>
    </relheader>
    <relrow>
      <relcell>
        <topicref href="batsonar.dita"/>
      </relcell>
      <relcell>
        <topicref href="batcaring.dita"/>  
        <topicref href="batfeeding.dita"/>
      </relcell>
      <relcell>
        <topicref href="batguano.dita"/>
        <topicref href="bathistory.dita"/>
      </relcell>
    </relrow>
  </reltable>
</map>
```

A DITA-aware tool may represent the <reltable> element graphically:

<table>
<thead>
<tr>
<th>type=&quot;concept&quot;</th>
<th>type=&quot;task&quot;</th>
<th>type=&quot;reference&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>batsonar.dita</td>
<td>batcaring.dita</td>
<td>batguano.dita</td>
</tr>
<tr>
<td></td>
<td>batfeeding.dita</td>
<td>bathistory.dita</td>
</tr>
</tbody>
</table>

On output, links should be added to topics that are in the same row, but not in the same cell. This allows simple maintenance of parallel relationships: for example, in this case, batcaring.dita and batfeeding.dita are two tasks that require the same supporting information (concept and reference topics) but might otherwise be unrelated. When topics in the same cell are in fact related, the cell's @collection-type attribute can be set to family. If some cells or columns are intended solely as supporting information and should not link back to topics in other cells, you can set the @linking attribute on the <relcell> or <relcolspec> to "targetonly".

In this example, the related links would be as follows:
Although such tables may initially take some time to learn and manipulate, they are inherently an efficient way to manage these links. In particular, they increase the prospect for reuse among topics, because those topics do not contain context-specific links. A relationship table also makes it easy to see and manage patterns; for example, the fact that `batfeeding.dita` and `batcaring.dita` have the same relationships to supporting information is clear from the table, but would require some comparison and counting to determine from the list summary just before this paragraph.

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Attributes common to many map elements* (see page 655) (with a narrowed definition of `@toc`, given below), *outputclass* (see page 0), and the attributes defined below. This element also uses `@type`, `@scope`, and `@format` from *Link relationship attribute group* (see page 661).

- **@toc**
  
  Specifies whether a topic appears in the table of contents (TOC). If the value is not specified locally, but is specified on an ancestor, the value will cascade from the closest ancestor. On this element the default value for `@toc` is "no". See *Attributes common to many map elements* (see page 655) for a complete definition of `@toc`.

- **@title**
  
  An identifying title for this element. This attribute is defined with the XML Data Type CDATA.

### 4.3.1.7 `<relrow>`

The `<relrow>` element defines a row in the relationship table (`<reltable>`). It creates a relationship between the cells in the row, which is expressed in output as links between the topics or resources referenced in those cells.

**Inheritance**

- map/relrow

**Example**

See `reltable` (see page 306).

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).
4.3.1.8 <relcell>

The <relcell> element defines a cell in the relationship table (<reltable>). The <topicref> elements that it contains are related to the <topicref> elements in other cells of the same row. By default, topics or resources that are referenced in the same cell are not related to each other, unless you change the @collection-type attribute of the <relcell> to indicate that they are related.

Inheritance
- map/relcell

Example
See reltable (see page 306).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Attributes common to many map elements (see page 655), and outputclass (see page 0). This element also uses @type, @scope, and @format from Link relationship attribute group (see page 661).

4.3.1.9 <relheader>

The <relheader> element is a row in a relationship table that contains column definitions (<relcolspec> elements). Each table can have only one set of column definitions.

Inheritance
- map/relheader

Example
See reltable (see page 306).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650).

4.3.1.10 <relcolspec>

The <relcolspec> element is used to define a column in a relationship table. The <relcolspec> element may be used to set default attribute values for the <topicref> elements in the column.

You can use the <relcolspec> element to set default values for the attributes of the topics that are referenced in the column. For example, when you set the @type attribute to "concept," all <topicref> elements in the column that do not have a @type attribute specified are treated as concepts. When values are specified for attributes of <relcell> or <relrow> elements, those values are inherited before those defined for <relcolspec> attributes. Values specified for attributes of <relcolspec> elements are inherited before those defined for the <reltable> element.

Beginning with DITA 1.2, you also can add <topicref> elements to the <relcolspec> element; this defines a relationship between the topics that are referenced in the <relcolspec> element and the topics that are referenced in the column of the relationship table. Note that this does not define a relationship between two cells in the same column; the only new relationship is between <topicref> targets in a <relcell> and <topicref> targets in that column's <relcolspec>.
Also beginning with DITA 1.2, if you add a `<title>` element to the `<relcolspec>` element, the content of the `<title>` element is used as the label for the related links that are defined and generated by the column. If the `<title>` element is not present, the labels for the related links are generated in the following ways:

- If the `<relcolspec>` element contains a `<topicref>` element that references a non-DITA resource, the value of the `<topicref>` element's navigation title is used for the label.
- If the `<relcolspec>` element contains a `<topicref>` element that references a DITA resource and the `@locktitle` attribute is set to "yes," the value of the `<topicref>` element's navigation title is used for the label.
- If the `<relcolspec>` element contains a `<topicref>` element that references a DITA resource and the `@locktitle` attribute is missing or set to "no," the label is derived from the `<navtitle>` or `<title>` element specified within the topic.
- If no title is specified and no `<topicref>` is present in the `<relcolspec>`, a rendering tool may choose to generate a title for the links generated from that column.

Inheritance
- `<map/relcolspec>`

Example

In this example, a relationship table is defined with three columns; one for "concept", one for "task", and one for "reference". Three cells are defined within one row. The first cell contains one concept topic: `puffins.dita`. The second cell contains two task topics: `puffinFeeding.dita` and `puffinCleaning.dita`. The third cell contains a reference topic: `puffinHistory.dita`. Setting the `@type` on each column allows (but does not require) processors to validate that the topics in each column are of the expected type.

```xml
<map>
  <reltable>
    <relheader>
      <relcolspec type="concept"/>
      <relcolspec type="task"/>
      <relcolspec type="reference"/>
    </relheader>
    <relrow>
      <relcell><topicref href="puffins.dita"/></relcell>
      <relcell><topicref href="puffinFeeding.dita"/></relcell>
      <relcell><topicref href="puffinHistory.dita"/></relcell>
    </relrow>
  </reltable>
</map>
```

Example with column titles

Consider the following relationship table:

```xml
<reltable>
  <relheader>
    <relcolspec type="task">
      <topicref navtitle="Troubleshooting" href="tbs.dita" locktitle="yes"/>
    </relcolspec>
    <relcolspec type="reference">
      <topicref navtitle="Messages" href="msg.dita" locktitle="yes"/>
    </relcolspec>
  </relheader>
  <relrow>
    <relcell><topicref navtitle="Debugging login errors" href="debug_login.dita"/>
    </relcell>
  </relrow>
</reltable>
```
In addition to the relationships defined by the rows in the relationship table, the following relationships are now defined by the columns in the relationship table:

- tbs.dita &rightarrow; debug_login.dita
- tbs.dita &rightarrow; checking_access.dita
- msg.dita &rightarrow; login_error_1.dita
- msg.dita &rightarrow; login_error_2.dita

Ignoring the headers for a moment, the <reltable> here would ordinarily define a two-way relationship between debug_login.dita and login_error1.dita. This will typically be expressed as a link from each to the other. An application MAY render the link with a language-appropriate heading such as "Related reference", indicating that the target of the link is a reference topic.

The headers change this by specifying a new title. In the second column, the <topicref> specifies a title of "Messages", which should now be used together with the link to anything in that column. So, a generated link from debug_login.dita to login_error1.dita should be rendered together with the title of "Messages". How this is rendered together with the link is up to the application.

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Attributes common to many map elements* (see page 655), and *outputclass* (see page 0). This element also uses @type, @scope, and @format from *Link relationship attribute group* (see page 661).

### 4.3.1.11 <ux-window>

Use the <ux-window> element to provide specifications for a window or viewport in which a user assistance topic or Web page can be displayed. The window or viewport can be referenced by the <resourceid> element associated with a topic or <topicref> element.

The <ux-window> element can be used anywhere within a map. If more than one <ux-window> element in a map has the same @name attribute, the window specification in the first instance of the element is used.

**Comment by Kristen Eberlein, 27 January 2014**
Should this read the "the first instance of the element in document order"? Emphasis added.

**Inheritance**

- map/ux-window

**Example**

In this example, a window with a name of "csh" is defined in the map. The window name is later referenced by the @ux-window attribute on the <resourceid> element.

```xml
<map title="Widget Help">
  <topicmeta>
    <ux-window id="fg23" name="csh" top="10" left="20" height="400" width="500" features="status=yes,toolbar=no,menubar=no,location=no" relative="yes" full-screen="no" />
  </topicmeta>
</map>
```
In this example, different window specifications are defined for tablet and desktop presentation:

```xml
<map title="Puggles Help">
  <topicmeta>
    <ux-window id="p76" name="ux-tablet" top="5%" left="5%" height="90%" width="90%"
      features="status=no,toolbar=no,menubar=no,location=no" relative="false",
      full-screen="no" />
    <ux-window id="p80" name="ux-deskop" top="5%" left="55%" height="60%" width="40%"
      features="status=yes,toolbar=no,menubar=no,location=yes" relative="false",
      full-screen="no" />
  </topicmeta>
  <topicref href="c_puggles_intro.dita" type="concept">
    <!-- ... -->
  </topicref>
</map>
```

**Attributes**

The following attributes are available on this element: *ID attribute group* (see page 651), *Metadata attribute group* (see page 651), *Debug attribute group* (see page 654), *class (Not for use by authors)* (see page 0), and the attributes defined below.

@name (REQUIRED)

The value used to refer to this window definition. This attribute is defined with the XML Data Type CDATA.

@top

The top position of the window. The value of this attribute is a real number optionally followed by a unit of measure from the set of pc, pt, px, in, cm, mm, em (picas, points, pixels, inches, centimeters, millimeters, and ems respectively). The default unit is px (pixels). This attribute is defined with the XML Data Type CDATA.

@left

The left position of the window. The value of this attribute is a real number optionally followed by a unit of measure from the set of pc, pt, px, in, cm, mm, em (picas, points, pixels, inches, centimeters, millimeters, and ems respectively). The default unit is px (pixels). This attribute is defined with the XML Data Type CDATA.

@height

The height of the window. The value of this attribute is a real number optionally followed by a unit of measure from the set of pc, pt, px, in, cm, mm, em (picas, points, pixels, inches, centimeters, millimeters, and ems respectively). The default unit is px (pixels). This attribute is defined with the XML Data Type CDATA.

@width

The width of the window. The value of this attribute is a real number optionally followed by a unit of measure from the set of pc, pt, px, in, cm, mm, em (picas, points, pixels, inches, centimeters, millimeters, and ems respectively). The default unit is px (pixels). This attribute is defined with the XML Data Type CDATA.

@on-top

Indicates whether the window stays on top of all windows on the desktop, rather than staying on top of only the calling window. Allowable values are: "yes", "no", and `-dita-use-conref-target`. The default value is "no".

@features

A list of other features (size, position, scrollbars, etc.) of the window. Each feature name and value must not contain any blank space, and each feature name and value must be separated by a comma or other delimiter character. This attribute is defined with the XML Data Type CDATA.
@relative
Indicates whether the window dimensions are relative to the calling window, or are absolute positions. The default value is "no". Allowable values are:

no
The window dimensions specified on this element are absolute positions (not relative to the calling window).

yes
The window dimensions specified on this element are relative to the calling window.

-dita-use-conref-target
See Using the -dita-use-conref-target value (see page 666) for more information.

@full-screen
Indicates whether the window should be displayed in a maximized state. Allowable values are: "yes", "no", and -dita-use-conref-target. The default value is "no".

4.3.2 Map group elements

The map group domain elements define, group, or reference content. Many of the map group elements are convenience elements, which means that they simply make it easier for an author to make use of existing functions.

For example, the <topichead> element allows a map to specify a heading without allowing a reference to a topic. While a <topicref> element may accomplish the same thing by creating a title and leaving off the @href attribute, the <topichead> element simply makes the intent clearer and prevents the accidental inclusion of an @href attribute.

4.3.2.1 <anchorref>

The <anchorref> element is used to reference an <anchor> element in a map. The contents of an <anchorref> element are rendered both in the original authored location and at the location of the referenced <anchor> element. The referenced <anchor> element may be defined in the current map or another map. When possible, this integration is done when displaying the map with <anchor> to an end user.

This function of the <anchorref> element is similar to that provided by the @anchorref attribute of the <map> element. However, instead of attaching an entire map to an anchor point, this element allows the author to attach only the contents of a single map branch. This enables architects to reuse a branch of content without reusing the entire map.

If the rendering platform does not support runtime integration of navigation based on the anchor point, a build system should treat the <anchorref> element similar to a "conref push" instruction by pushing the content to the spot that contains the <anchor>. Note that many <anchorref> elements may push content to the same point; the order in which items are pushed is left undefined, although the order within a single <anchorref> is preserved.

Metadata cascading must take place in the original authored context, because the branch of content defined with the <anchorref> remains independent from the referenced map. The <anchorref> content does not take on the cascading metadata at the <anchor> location. For example, if the map containing the <anchorref> element sets a local copyright, that copyright cascades to the <anchorref> element and its children; it is retained after the content is rendered at the target <anchor> element.

By default, the content of the <anchorref> element is rendered at both the anchor target and the original location. To prevent the content from being rendered at the location of the <anchorref> element, set toc="no" on the <anchorref> element, and then set toc="yes" on each of its children so that they will not inherit the toc="no" setting.
Inheritance

+ map/topicref mapgroup-d/anchorref

Example

```xml
<topicref href="carPrep.dita">
  <topicref href="beforePrep.dita"/>
  <anchor id="prepDetail"/>
  <topicref href="afterPrep.dita"/>
</topicref>

<!-- ... -->
<topicref href="astroTasks.dita">
  <topicref href="astroOverview.dita"/>
  <anchorref id="#prepDetail">...
  <topicref href="astroChecklist.dita"/>
  <topicref href="otherPreparation.dita"/>
</anchorref>
  <topicref href="astroConclusion.dita"/>
</topicref>
```

**Figure 61: Initial map contents**

```xml
<topicref href="carPrep.dita">
  <topicref href="beforePrep.dita"/>
  <anchor id="prepDetail"/>
  <topicref href="astroChecklist.dita"/>
  <topicref href="otherPreparation.dita"/>
  <topicref href="afterPrep.dita"/>
</topicref>

<!-- ... -->
<topicref href="astroTasks.dita">
  <topicref href="astroOverview.dita"/>
  <topicref href="astroChecklist.dita"/>
  <topicref href="otherPreparation.dita"/>
  <topicref href="astroConclusion.dita"/>
</topicref>
```

**Figure 62: Effective result of evaluating the <anchorref> element**

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661) (with narrowed definitions of @href, @type, and @format, all given below), *Attributes common to many map elements* (see page 655), *Topicref element attributes group* (see page 663), *outputclass* (see page 0), @keys, and @keyref.

Comment by robander

Need to verify that this gets the new attributes @cascade and @keyscope from common map attributes

@href

A pointer to an <anchor> element in this or another DITA map. When rendered, the contents of the current element will be copied to the location of the <anchor>. See *The href attribute* (see page 663) for supported syntax when referencing a map element. This attribute is defined with the XML Data Type CDATA.

@type

Describes the target of a reference. For the <anchorref> element, this value defaults to "anchor", because the element is expected to point to an <anchor> element in this or another map.

@format

The @format attribute identifies the format of the resource being referenced. For the <anchorref> element, this value defaults to "ditamap", because the element references a point in a map.
4.3.2.2 <keydef>

The `<keydef>` element is a convenience element that is used to define keys without any of the other effects that occur when using a `<topicref>` element: no content is included in output, no title is included in the table of contents, and no linking or other relationships are defined. The `<keydef>` element is not the only way to define keys; its purpose is to simplify the process by defaulting several attributes to achieve the described behaviors.

**Inheritance**

+ map/topicref mapgroup-d/keydef

**Example**

The following example defines keys that can be used to refer to the indicated topics. These keys may be used within the map, from any topic in this map, or in any context where this map is imported. Note that the @processing-role attribute defaults to "resource-only", which ensures that specified topics will not be rendered in a print document or in a navigation TOC based on this definition in the map. In addition, it means that links will not be generated to or from the `<keydef>` elements.

```xml
<map>
  <title>Defining bird keys</title>
  <keydef keys="darwinfinch galapagosfinch" href="galapagosfinch.dita"/>
  <keydef keys="goldfinch" href="about-goldfinches.dita"/>
  <keydef keys="puffin" href="about-puffins.dita"/>
  <keydef keys="loon diver" href="common-loon.dita"/>
</map>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661) (with a narrowed definition of @href, given below), *Attributes common to many map elements* (see page 655) (with a narrowed definition of @processing-role, given below), *Topicref element attributes group* (see page 663), *outputclass* (see page 0), @keyref, and the attributes defined below.

- **@keys (REQUIRED)**
  
  On this element the @keys attribute is required, because the purpose of the element is to define a key. Otherwise, the attribute is the same as described in *The keys attribute* (see page 665).

- **@href**
  
  A pointer to the resource represented by the `<topicref>`. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

- **@processing-role**
  
  For this element, the default value for @processing-role is "resource-only". Otherwise, the definition matches the one found in *Attributes common to many map elements* (see page 655).

4.3.2.3 <mapref>

The `<mapref>` element is a convenience element that is equivalent to a `<topicref>` element with the @format attribute set to "ditamap". The hierarchy of the referenced map is merged into the container map at the position of the reference, and the relationship tables of the child map are added to the parent map.

**Inheritance**

+ map/topicref mapgroup-d/mapref
Example

```
<map id="lib">
  <topicref href="netlib.dita"/>
  <topicref href="dblib.dita"/>
  <!-- ... -->
</map>
```

**Figure 63: Sample document lib.ditamap that is reusable in other locations**

```
<map id="standardlib">
  <topichead navtitle="Developing with standard libraries">
    <mapref href="lib.ditamap"/>
    <!-- ... -->
  </topichead>
  <!-- ... -->
</map>
```

**Figure 64: Map that reuses lib.ditamap**

```
<map id="standardlib">
  <topichead navtitle="Developing with standard libraries">
    <topicref href="netlib.dita"/>
    <topicref href="dblib.dita"/>
    <!-- ... -->
    <!-- ... -->
  </topichead>
  <!-- ... -->
</map>
```

**Figure 65: Rendered result**

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661) (with narrowed definitions of @href and @format, given below), *Attributes common to many map elements* (see page 655), *Topicref element attributes group* (see page 663), *outputclass* (see page 0), @keyref, and @keys.

@format

On this element the @format attribute sets a default value of "ditamap", because the purpose of the element is to reference a ditamap document. Otherwise, the attribute is the same as described in *Link relationship attribute group* (see page 661).

@href

A pointer to the resource represented by the <topicref>. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

### 4.3.2.4 <topicgroup>

The <topicgroup> element groups <topicref> elements for common treatment without affecting the structural hierarchy of the map, as opposed to nesting <topicref> elements, which does imply a structural hierarchy. The <topicgroup> element can provide linking relationships and shared, inherited attributes to the set of elements that it contains without affecting the resulting table of contents or navigation.

Beginning with DITA 1.2, you are able to specify a <navtitle> element within the <topicmeta> element inside of a <topicgroup>. The <topicgroup> element is meant as a non-titled grouping element, so adding a <navtitle> element to the <topicgroup> element has no defined purpose, and processors must ignore the title. Processors may (but need not) issue a message when ignoring the title.

DITA Version 1.3 Specification

Copyright © OASIS Open 2005, 2014. All Rights Reserved.
Inheritance
+ map/topicref mapgroup-d/topicgroup

Example
Each `<topicref>` element in the following example inherits the `@audience` and `@linking` attributes. In this way the common attributes are set for the entire group of `<topicref>` elements without affecting the navigation hierarchy.

```xml
<topicgroup audience="novice" linking="none">
  <topicref href="this.dita"/>
  <topicref href="that.dita"/>
  <topicref href="theother.dita"/>
</topicgroup>
```

Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650), *Attributes common to many map elements* (see page 655), and *outputclass* (see page 0). Although `@locktitle` is available as part of *Attributes common to many map elements* (see page 655), it has no defined purpose for this element.

The `@scope`, `@format`, and `@type` attributes from *Link relationship attribute group* (see page 661) are also available.

4.3.2.5 `<topichead>`

The `<topichead>` element provides a title-only entry in a navigation map, which should appear as a heading when the map is rendered as a table of contents. In print contexts it should also appear as a heading in the rendered content.

Beginning with DITA 1.2, the navtitle can be specified by using a `<navtitle>` element within the `<topicmeta>` element, so the `<topichead>` element no longer requires the `@navtitle` attribute. In order to ensure backward compatibility with earlier versions of DITA, the new `<navtitle>` element is not required. However, a `<topichead>` element must contain either a `@navtitle` attribute or a `<topicmeta>` element that contains a `<navtitle>` element. DITA processors should generate a warning if a navigation title is not specified.

Inheritance
+ map/topicref mapgroup-d/topichead

Example
Note that in the following example, the first `<topichead>` element uses a `<navtitle>` element to provide the title, while the second `<topichead>` element uses a `@navtitle` attribute. This is only to illustrate that both uses are valid; in general, the element is preferred over the attribute.

```xml
<map>
  <topichead>
    <topicmeta><navtitle>Computers</navtitle></topicmeta>
    <topicref href="eniac.dita"/>
    <topicref href="system360.dita"/>
    <topicref href="pdp8.dita"/>
  </topichead>
  <topichead navtitle="Books">
    <topicref href="hardback.dita"/>
    <topicref href="paperback.dita"/>
  </topichead>
</map>
```
Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Attributes common to many map elements (see page 655), @navtitle and @copy-to from Topicref element attributes group (see page 663), and outputclass (see page 0). Although @locktitle is available as part of Attributes common to many map elements (see page 655), it has no defined purpose for this element.

The @scope, @format, and @type attributes from Link relationship attribute group (see page 661) are also available.

4.3.2.6 <topicset>

The <topicset> element defines a complete unit of content that can be reused in other DITA maps or other <topicset> elements. Unlike the base <topicref> element, the <topicset> is explicitly intended to define a set of topics that are often or always used as a unit; it can be especially useful for task composition in which larger tasks are composed of smaller tasks. The @id attribute on a <topicset> is required, which ensures that the complete unit is available for reuse in other contexts.

A <topicset> is similar to a source file that contains nested topics, in that the combination of topics constitutes a complete self-contained unit. That unit of content can stand independently of the containing, prior, and following content within the original map context.

Inheritance

+ map/topicref mapgroup-d/topicset

Example

This <topicset> represents a set of overview information about SQL. The information is reusable as a unit.

```xml
<topicset id="sqlbasics" href="sqlOverview.dita">
  <topicref href="sqlSelection.dita"/>
  <topicref href="sqlJoin.dita"/>
  <topicref href="sqlFilter.dita"/>
  <!-- ... -->
</topicset>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), Attributes common to many map elements (see page 655), Topicref element attributes group (see page 663), outputclass (see page 0), @keys, and @keyref.

@id (REQUIRED)

This ID is the target for references by to the current set of information. The ID is required in order to ensure that a <topicset> is defined as a reusable unit of information. See ID attribute (see page 63) for more details.

@href

A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.
4.3.2.7 <topicsetref>

The <topicsetref> element references a <topicset> element. The referenced <topicset> element can be defined in the current map or in another map.

When possible, applications should treat the referenced <topicset> as an independent unit that always retains its identity. For example, an application that renders DITA for a dynamic navigation platform may generate a reusable navigation structure for each <topicset>, and each <topicsetref> is retained as a reference to that structure. This differs slightly from the processing of the @conref attribute, which results in a literal copy of the referenced content.

For situations that do not support reusing a topic set as an independent unit, such as a rendered PDF, applications may resolve the <topicsetref> element in a manner similar to other <topicset> elements that have the @format attribute set to "ditamap". This may result in a new instance of the <topicset> element.

As with other cases where the attribute format="ditamap" is specified or used as a default, the use of topic references nested inside of <topicsetref> is undefined.

Inheritance
+ map/topicref mapgroup-d/topicsetref

Example
The sql-tutorial.ditamap file contains a <topicset> element that groups together several topics that collectively comprise an overview of SQL.

```
<!-- Excerpt from sql-tutorial.ditamap -->
<topicset id="sqlbasics" href="sqlOverview.dita">
  <topicref href="sqlSelection.dita"/>
  <topicref href="sqlJoin.dita"/>
  <topicref href="sqlFilter.dita"/>
<!-- ... -->
</topicset>
```

Figure 66: DITA map that contains a <topicset> element

Another map can include that topic set, in addition to content related to programming with JDBC.

```
<topichead navtitle="Mastering JDBC">
  <topicsetref href="sql-tutorial.ditamap#sqlbasics"/>
  <topicref href="jdbcPrepare.dita"/>
<!-- ... -->
</topichead>
```

Figure 67: DITA map that contains a <topicsetref> element

A reader of the JDBC information will see the content integrated as a single unit.

```
<topichead navtitle="Mastering JDBC">
  <topicset id="sqlbasics" href="sqlOverview.dita">
    <topicref href="sqlSelection.dita"/>
    <topicref href="sqlJoin.dita"/>
    <topicref href="sqlFilter.dita"/>
    <!-- ... -->
  </topicset>
  <topicref href="jdbcPrepare.dita"/>
<!-- ... -->
</topichead>
```

Figure 68: Result of the reuse
**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661) (with narrowed definitions of `@href`, `@format`, and `@type`, given below), *Attributes common to many map elements* (see page 655), *Topicref element attributes group* (see page 663), *outputclass* (see page 4), `@keys`, and `@keyref`.

- **@href**
  A pointer to the `<topicset>` represented by `<topicsetref>`. See *The href attribute* (see page 663) for detailed information on syntax.

- **@format**
  The `@format` attribute identifies the format of the resource being referenced. For the `<topicsetref>` element, this attribute defaults to "ditamap", because the element typically references a branch of a map. See *The format attribute* (see page 676) for details on other supported values.

- **@type**
  Describes the target of a reference. For the `<topicsetref>` element, this attribute defaults to "topicset". See *The type attribute* (see page 674) for detailed information on other supported values and processing implications.

### 4.4 Metadata elements

Metadata elements include information that is located within the `<topicmeta>` element (in maps) or `<prolog>` element (in topics), as well as indexing elements that can be placed in additional locations within topic content.

#### 4.4.1 Prolog (metadata) elements

The prolog elements represent the metadata associated with a document. Most of the metadata in a topic prolog can also be authored in a DITA map, in the map's `<topicmeta>` element.

The primary types of information that you can store in the prolog include:

- author
- copyright information
- critical tracking dates
- permissions for use/management of the content
- key words and index terms related to the topic
- extensive metadata about the content of the document
- a resourced that allows a topic to be associated with external resources such as linking to programming components as contextual help

#### 4.4.1.1 <prolog>

The `<prolog>` element contains information about the topic as an whole (for example, author information or subject category) that is either entered by the author or maintained by a software application. Much of the metadata inside the `<prolog>` will not be displayed with the topic when the topic is rendered, but may be used by processes that generate search indexes or customize navigation.

**Inheritance**

- topic/prolog

**Example**

```xml
<prolog>
  <metadata>
    <audience type="user" job="using" experiencelevel="novice"/>
  </metadata>
</prolog>
```
**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650).

4.4.1.2 `<audience>`

The `<audience>` metadata element indicates, through the value of its `@type` attribute, the intended audience for a topic.

Since a topic can have multiple audiences, you can include multiple audience elements. For each audience you specify, you can identify the high-level task they are trying to accomplish with the `@job` attribute, and the level of experience expected with the `@experiencelevel` attribute. The `<audience>` element may be used to provide a more detailed definition of values used throughout the map or topic on the `@audience` attribute.

Many of the attributes on the `<audience>` element have enumerated values, which may be restricted by using constraints or extended by using associated attributes. For instance, the `@othertype` attribute can be used to extend the audience type enumeration.

**Inheritance**
- topic/audience

**Example**

For a command reference topic for experienced programmers, the following might be an appropriate indication of that audience:

```xml
<audience type="programmer" job="programming" experiencelevel="expert"/>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and the attributes defined below.

- `@type`
  Indicates the kind of person for whom the content of the topic is intended. Note that this differs from the `@type` attribute on many other DITA elements. Allowable values are:
  - `user`
    A user of the product
  - `purchaser`
    A product purchaser
  - `administrator`
    A product administrator
  - `programmer`
    A programmer
  - `executive`
    An executive
  - `services`
    Someone who provides services related to the product
  - `other`
    Use the value specified by the `@othertype` attribute.
-dita-use-conref-target

See Using the -dita-use-conref-target value (see page 666) for more information.

@othertype

Indicates an alternate audience type, when the type is not available in the @type attribute value list. This value is used as the user-provided audience when the @type attribute value is set to "other." This attribute is defined with the XML Data Type CDATA.

@job

Indicates the high-level task the audience for the topic is trying to accomplish. Different audiences may read the same topic in terms of different high-level tasks; for example, an administrator may read the topic while administering, while a programmer may read the same topic while customizing. Beginning with DITA 1.2, values in this attribute are not limited to a small number of choices; the following values were used in DITA 1.0 and DITA 1.1, and are still provided as sample values: installing, customizing, administering, programming, using, maintaining, troubleshooting, evaluating, planning, migrating, other, Using the -dita-use-conref-target value (see page 666) This attribute is defined with the XML Data Type NMToken.

@otherjob

If the @job attribute value is "other" the value of this attribute is used to identify a kind of job other than the default ones provided by the @job attribute. This attribute is defined with the XML Data Type CDATA.

@experiencelevel

Indicates the level of experience the audience is assumed to possess. Different audiences may have different experience levels with respect to the same topic; for example, a topic may require general knowledge from a programmer, but expert knowledge from a user. Beginning with DITA 1.2, values in this attribute are not limited to a small number of choices; the following values were used in DITA 1.0 and DITA 1.1, and are still provided as sample values:

novice
  A first time user.

general
  The most common user.

expert
  An experienced user.

-dita-use-conref-target

See Using the -dita-use-conref-target value (see page 666) for more information.

This attribute is defined with the XML Data Type NMToken.

@name

Used to associate the <audience> element with values used in the @audience attribute. This attribute is defined with the XML Data Type CDATA.

Comment by robander
This was previously marked as Required but has never been required in the DTD - placing it with normal ones, expecting TC to agree.

4.4.1.3 <author>

The <author> metadata element contains the name of the topic's author.

The author is usually the person, organization, or application that created the content. This element is equivalent to the <Creator> element in Dublin Core.

Inheritance

- topic/author
Example

```xml
<prolog>
  <author type="creator">Jane</author>
  <author type="contributor">John</author>
</prolog>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition for @type, given below), and @keyref.

@type

Describes the target of a reference. See The type attribute (see page 674) for detailed information on supported values and processing implications. Note that this differs from the @type attribute on many other DITA elements. Beginning with DITA 1.2, values in this attribute are not limited to a small number of choices; the following values were used in DITA 1.0 and DITA 1.1, and are also recognized for the <author> element (and its specializations):

creator

The primary or original author of the content.

contributor

An additional author who is not primary.

-dita-use-conref-target

See Using the -dita-use-conref-target value (see page 666) for more information.

4.4.1.4 <brand>

The <brand> element indicates the manufacturer or brand associated with the product described by the parent <prodinfo> element.

Inheritance

- topic/brand

Example

```xml
<prodinfo>
  <prodname>Some Product</prodname>
  <vrmlist><vrm version="1"/></vrmlist>
  <brand>eServer</brand>
  <series>iSeries</series>
</prodinfo>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650).

4.4.1.5 <category>

The <category> element represents any category by which a topic might be classified for retrieval or navigation. For example, the categories could be used to group topics in a generated navigation bar. Topics can belong to multiple categories.

Such classifications are likely to come from an enumerated or hierarchical set.

This element is equivalent to both the <Coverage> element and the <Subject> element in Dublin Core.
**Inheritance**
- topic/category

**Example**

```xml
<prolog>
<metadata>
<category>Things that are blue</category>
</metadata>
</prolog>
```

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650).

4.4.1.6 `<component>`

The `<component>` element describes the component of the product that this topic is concerned with. For example, a product might be made up of many components, each of which is installable separately. Components might also be shared by several products so that the same component is available for installation with many products. An implementation may (but need not) use this identification to check cross-component dependencies when some components are installed, but not others. An implementation may also (but need not) use the identification to make sure that topics are hidden, removed, or flagged in some way when the component they describe isn't installed.

**Inheritance**
- topic/component

**Example**

```xml
<prodinfo>
<prodname>BatCom</prodname>
<vrmlist>
<vrm version="v5r2"/>
</vrmlist>
<component>TCP/IP</component>
</prodinfo>
```

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650).

4.4.1.7 `<copyright>`

The `<copyright>` element specifies legal ownership of the content.

The `<copyright>` element is used for a single copyright entry. It includes the copyright years and the copyright holder. Multiple `<copyright>` statements are allowed.

This element is equivalent to the `<Rights>` element in Dublin Core.

**Inheritance**
- topic/copyright
Example

```xml
<prolog>
  <copyright>
    <copyryear year="2001-04-12"></copyryear>
    <copyrholder>IBM</copyrholder>
  </copyright>
  <copyright type="secondary">
    <copyryear year="2002-03-03"></copyryear>
    <copyrholder>Sweetones Publishing, Inc.</copyrholder>
  </copyright>
</prolog>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and the attribute defined below.

@type

Indicates the legal status of the copyright holder. Note that this differs from the @type attribute on many other DITA elements. Beginning with DITA 1.2, values in this attribute are not limited to a small number of choices; the following values were used in DITA 1.0 and DITA 1.1, and are still provided as sample values:

- primary
  The copyright holder with first claim on the copyright

- secondary
  An additional copyright holder who is not primary

-dita-use-conref-target
  See [Using the -dita-use-conref-target value](see page 666) for more information.

4.4.1.8 <copyrholder>

The `<copyrholder>` element names the copyright holder that holds legal rights to the material contained in the topic.

Inheritance

- topic/copyrholder

Example

```xml
<copyright>
  <copyryear year="2001"></copyryear>
  <copyrholder>IBM</copyrholder>
</copyright>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650).

4.4.1.9 <copyryear>

The `<copyryear>` element contains the copyright year as specified by the @year attribute.

Inheritance

- topic/copyryear
Example

```xml
<copyright>
  <copyryear year="2001"></copyryear>
  <copyrholder>IBM</copyrholder>
</copyright>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and the attribute defined below.

@year

The year in YYYY format. This attribute is defined with the XML Data Type CDATA.

4.4.1.10 `<created>`

The `<created>` element specifies the document creation date using the `@date` attribute.

Inheritance

- topic/created

Example

```xml
<prolog>
  <critdates>
    <created date="2001-06-12"></created>
    <revised modified="2001-08-20"></revised>
  </critdates>
</prolog>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *Date attributes group* (see page 659), and the attribute defined below.

@date (REQUIRED)

The document creation date. Enter the date as YYYY-MM-DD where YYYY is the year, MM is the month from 01 to 12, and DD is the day from 01-31. See *A Summary of the International Standard Date and Time Notation* for background. This attribute is defined with the XML Data Type CDATA.

4.4.1.11 `<critdates>`

The `<critdates>` element contains the critical dates in a document life cycle, such as the creation date and multiple revision dates.

This element is equivalent to the `<Date>` element in Dublin Core.

Inheritance

- topic/critdates

Example

```xml
<prolog>
  <critdates>
    <created date="2001-06-12"></created>
    <revised modified="2001-08-20"></revised>
  </critdates>
</prolog>
```
**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650).

### 4.4.1.12 `<featnum>`

The `<featnum>` element contains the feature number of a product in the metadata.

**Inheritance**

- topic/featnum

**Example**

```xml
<prodinfo>
    <prodnname>BatCom</prodnname>
    <vrmlist>
        <vrm version="v5r2"/>
    </vrmlist>
    <featnum>135</featnum>
    <component>TCP/IP</component>
</prodinfo>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650).

### 4.4.1.13 `<keywords>`

The `<keywords>` element contains a list of terms from a controlled or uncontrolled subject vocabulary that applies to the topic or map. The keywords may be used by a search engine. The keywords are marked up using the `<indexterm>` and/or `<keyword>` elements.

All `<keyword>` and/or `<indexterm>` elements in the `<keywords>` element are considered part of the topic's metadata and should be reflected in the output as appropriate for the output medium.

> **Note:** While the `<keyword>` element may be used inline, the `<keywords>` element is not an inline element. The `<keywords>` element only appears in the `<topicmeta>` or `<prolog>`, and is used to specify keywords that apply to the topic.

**Inheritance**

- topic/keywords

**Example**

The following example is metadata from an installation task:

```xml
<prolog>
    <metadata>
        <keywords>
            <keyword>installing</keyword>
            <keyword>uninstalling</keyword>
            <keyword>prerequisites</keyword>
            <keyword>helps</keyword>
            <keyword>wizards</keyword>
        </keywords>
    </metadata>
</prolog>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650).
4.4.1.14 <metadata>

The <metadata> section of the <prolog> contains information about a topic such as audience and product information. Metadata can be used by computational processes to select particular topics or to prepare search indexes or to customize navigation. Elements inside of <metadata> provide information about the content and subject of a topic; <prolog> elements outside of <metadata> provide lifecycle information for the content unit (such as the author or copyright), which are unrelated to the subject.

Beginning with DITA 1.2, the <metadata> element is available inside <topicmeta> in maps, although the contents of <metadata> are still available directly inside <topicmeta>. As with the <prolog>, the <metadata> element within <topicmeta> allows you to group elements that describe the content or subject of the target. The primary purpose for enabling the <metadata> element within maps is to allow easier reuse between topics and maps.

Inheritance
- topic/metadata

Example

Metadata within a topic:

```xml
<prolog>
  <metadata>
    <audience type="user" job="using" experiencelevel="novice"/>
  </metadata>
</prolog>
```

Metadata within a map:

```xml
<topicref href="metadata.dita" navtitle="metadata element">
  <topicmeta>
    <metadata>
      <keywords>
        <indexterm>metadata element</indexterm>
      </keywords>
    </metadata>
    </topicmeta>
</prolog>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *mapkeyref* (see page 0).

4.4.1.15 <othermeta>

The <othermeta> element can be used to identify properties not otherwise included in <metadata> and to assign name/content values to those properties. The @name attribute identifies the property and the @content attribute specifies the property's value. All <othermeta> elements are considered part of the topic's metadata and should be reflected in the output as appropriate for the output medium.

Inheritance
- topic/othermeta

Example

```xml
<othermeta name="ThreadWidthSystem" content="metric"/>
```
Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and the attributes defined below.

@name (REQUIRED)

The name of the metadata property. This attribute is defined with the XML Data Type CDATA.

@content (REQUIRED)

The value for the property named in the @name attribute. This attribute is defined with the XML Data Type CDATA.

@translate-content

Indicates whether the @content attribute of the defined metadata property should be translated or not. Allowable values are yes, no, and -dita-use-conref-target.

4.4.1.16 <permissions>

The <permissions> element in a topic's metadata specifies the level of entitlement needed to access the content.

The <permissions> element indicates any preferred controls for access to content. Topics can be filtered based on the <permissions> element. This capability depends on your output formatting process.

Comment by robander, 24 February 2014

Noted while editing - this seems odd, not sure if the previous paragraph is needed? If so, seems it could use some rewording.

Inheritance

- topic/permissions

Example

```xml
<prolog>
  <permissions view="entitled"/>
</prolog>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and the attribute defined below.

@view

Defines the classifications of viewers allowed to view the document. This attribute is defined with the XML Data Type CDATA. Beginning with DITA 1.2, values in this attribute are not limited to a small number of choices; the following values were used in DITA 1.0 and DITA 1.1, and are still provided as sample values:

- internal
  For internal use only.

- classified
  For a certain group, only.

- all
  The world.

- entitled
  Special folks, only.

- dita-use-conref-target
See *Using the -dita-use-conref-target value* (see page 666) for more information.

**Comment by robander**

1.2 spec used the content above about removing enumeration, but did not change the "Data Type" column to CDATA. Fixing for 1.3.

#### 4.4.1.17 `<platform>`

The `<platform>` metadata element contains a description of the operating system and/or hardware related to the product being described by the `<prodinfo>` element. The `<platform>` element may be used to provide a more detailed definition of values used throughout the map or topic on the `@platform` attribute.

*Inheritance*
- topic/platform

*Example*

See *prodinfo* (see page 330).

*Attributes*

The following attributes are available on this element: *Universal attribute group* (see page 650).

#### 4.4.1.18 `<prodinfo>`

The `<prodinfo>` metadata element contains information about the product or products that are the subject matter of the current topic. The `<prodinfo>` element may be used to provide a more detailed definition of values used throughout the map or topic on the `@product` attribute.

*Inheritance*
- topic/prodinfo

*Example*

```
<prolog>
<metadata>
  <prodinfo>
    <prodname>Transcription Assistant</prodname>
    <vrmlist><vrm version="1" release="3" modification="1"/></vrmlist>
    <platform>Linux</platform>
    <prognum>SN-12345T</prognum>
  </prodinfo>
</metadata>
</prolog>
```

*Attributes*

The following attributes are available on this element: *Universal attribute group* (see page 650).

#### 4.4.1.19 `<prodname>`

The `<prodname>` metadata element contains the name of the product that is supported by the information in this topic.

*Inheritance*
- topic/prodname
Example
See prodinfo (see page 330).

Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650).

4.4.1.20 <prognum>

The <prognum> metadata element identifies the program number of the associated product. This is typically an order number or a product tracking code that could be replaced by an order number when a product completes development.

Inheritance
- topic/prognum

Example
See prodinfo (see page 330).

Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650).

4.4.1.21 <publisher>

The <publisher> metadata element contains the name of the person, company, or organization responsible for making the content or subject of the topic available.

This element is equivalent to the <Publisher> element in Dublin Core.

Inheritance
- topic/publisher

Example

```
<prolog>
  <author>Ivan</author>
  <publisher>AJ Printing Inc.</publisher>
</prolog>
```

Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661), and @keyref.

4.4.1.22 <resourceid>

The <resourceid> element provides an identifier for applications that must use their own identifier scheme, such as context-sensitive help systems and databases.

Multiple @appid values can be associated with a single @appname value, and multiple @appname values can be associated with a single @appid value. Because the values for the @appid and @appname attributes work in combination to specify a specific ID for a specific application, each combination of values for the @appid and @appname attributes should be unique.
Versions of DITA before 1.3 used the @id attribute on the <resourceid> element to specify an ID for an external application. Beginning with DITA 1.3, this use of the @id attribute is deprecated in favor of using the @appid attribute.

**Inheritance**
- topic/resourceid

**Example**
In the following example, user-assistance context hooks are applied to three topics that are referenced from a DITA map. The second topic has two hooks for the same topic.

```xml
<map title="Widget Help">
  <topicref href="file_ops.dita" type="concept">
    <topicref href="saving.dita" type="task">
      <topicmeta>
        <resourceid id="ab12" appname="ua" appid="1234" ux-context-string="idh_filesave"
          ux-source-priority="topic-only" />
      </topicmeta>
    </topicref>
    <topicref href="deleting.dita" type="task">
      <topicmeta>
        <resourceid id="ab34" appname="ua" appid="2345" ux-context-string="idh_filedelete" />
        <resourceid id="ab56" appname="ua" appid="6789" ux-context-string="idh_filekill" />
      </topicmeta>
    </topicref>
    <topicref href="editing.dita" type="task">
      <topicmeta>
        <resourceid id="ab78" appname="ua" appid="5432" ux-context-string="idh_fileedit" ux-windowref="csh" />
      </topicmeta>
    </topicref>
  </topicref>
</map>
```

In the following example, a user-assistance context hook is defined in the prolog of a task topic. The context hook is made up of a context ID (value for @appid attribute) and a context string (value for @ux-context-string attribute). A user assistance window is also referenced for this topic.

```xml
<task id="fedt">
  <title>Editing a File</title>
  <prolog>
    <resourceid id="cd12" appname="ua" appid="5432" ux-context-string="idh_fileedit" ux-windowref="csh" />
  </prolog>
  <taskbody>
    <context>After you have created a new file, you can edit it.</context>
    <steps>
      <step><cmd>Open...</cmd></step>
      <step><cmd>Edit...</cmd></step>
      <step><cmd>Save...</cmd></step>
    </steps>
  </taskbody>
</task>
```

**Attributes**
The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below) and the attributes defined below.

@id
Before DITA 1.3, this attribute specified a value that was used by a specific application to identify the topic. That usage is deprecated in favor of using the `@appid` attribute. This attribute is defined with the XML Data Type CDATA.

`@appname`
A name for the external application that references the topic. This attribute is defined with the XML Data Type CDATA.

`@appid`
An ID used by an application to identify the topic. This attribute is defined with the XML Data Type CDATA.

`@ux-context-string`
Contains the value of a user-assistance context-string that is used to identify the topic. This attribute is defined with the XML Data Type CDATA.

`@ux-source-priority`
Contains a value that indicates the precedence of context hooks in the map and context hooks in the topic. This attribute is only valid when used within a `<topicref>` element in a map. Allowable values are `topic-and-map`, `topic-only`, `map-only`, `map-takes.priority`, `topic-takes.priority`, and `-dita-use-conref-target`.

`@ux-windowref`
Contains the name of the `<ux-window>` element that is used to display the topic when called from a help API. This attribute is defined with the XML Data Type CDATA.

### 4.4.1.23 `<revised>`

The `<revised>` metadata element is used to maintain tracking dates that are important in a topic development cycle, such as the last modification date, the original availability date, and the expiration date.

**Inheritance**
- topic/revised

**Example**

```xml
<prolog>
  <critdates>
    <created date="1999-01-01" golive="1999-02-15" expiry="9999-09-09"/>
    <revised modified="2003-03-03" golive="2002-02-03" expiry="9999-09-09"/>
  </critdates>
</prolog>
```

**Attributes**
The following attributes are available on this element: [Universal attribute group](see page 650), [Date attributes group](see page 659), and the attribute defined below.

`@modified` (REQUIRED)
The last modification date, entered as YYYY-MM-DD, where YYYY is the year, MM is the month from 01 to 12, and DD is the day from 01-31. This attribute is defined with the XML Data Type CDATA.

**Comment by robander**
It's CDATA in DTD, but I think we restrict in Schema to match this definition?

### 4.4.1.24 `<series>`

The `<series>` metadata element contains information about the product series that the topic supports.

**Inheritance**
- topic/series
### Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650).

#### 4.4.1.25 <source>

The `<source>` element identifies a resource from which the present topic is derived, either completely or in part.

The `<source>` element contains a description of the resource. Alternatively, the `@href` or `@keyref` attributes may be used to reference a description of the resource. It is implementation-dependent what it means when the element has both content and an attribute-based reference to another resource.

This element is equivalent to the `<Source>` element in Dublin Core.

**Inheritance**
- topic/source

**Example**

```
<prolog>
  <source>Somewhere, someplace</source>
</prolog>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661) (with a narrowed definition for `@href`, given below), and `@keyref`.

- `@href`
  
  Provides a reference to a resource from which the present resource is derived. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. This attribute is defined with the XML Data Type CDATA.

#### 4.4.1.26 <vrmlist>

The `<vrmlist>` element contains a set of `<vrm>` elements for logging the version, release, and modification information for multiple products or versions of products to which the topic applies.

**Inheritance**
- topic/vrmlist

**Example**

The recent versions of a hypothetical product might be logged thus using the `<vrmlist>` markup:

```
<prolog>
  <metadata>
    <prodinfo>
      <prodname>Widge-o-matic</prodname>
    </prodinfo>
  </metadata>
</prolog>
```
This indicates that the topic covers Version 1, release 2, modification levels 0 and 1 (often expressed as version 1.2.0 and 1.2.1).

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650).

### 4.4.1.27 <vrm>

The `<vrm>` element contains information about a single product's version, modification, and release, to which the current topic applies.

**Inheritance**

- topic/vrm

**Example**

The recent versions of a hypothetical product might be logged thus using the `<vrmlist>` markup:

```xml
<prolog>
  <metadata>
    <prodinfo>
      <prodname>Widge-o-matic</prodname>
      <vrmlist>
        <vrm version="1" release="2" modification="0"/>
        <vrm version="1" release="2" modification="1"/>
      </vrmlist>
    </prodinfo>
  </metadata>
</prolog>
```

This indicates that the topic covers Version 1, release 2, modification levels 0 and 1 (often expressed as version 1.2.0 and 1.2.1).

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and the attributes defined below.

@version *(REQUIRED)*

Indicates the released version number of the product(s) that the document describes. This attribute is defined with the XML Data Type CDATA.

Comment by robander, 12 February 2014

Just need to double check - believe that this should no longer be required in 1.3

@release

Contains the product release identifier. This attribute is defined with the XML Data Type CDATA.

@modification

Indicates the modification level of the current version and release. This attribute is defined with the XML Data Type CDATA.
4.4.2 Indexing group elements

The indexing domain provides several new elements for use with indexing. The new elements allow authors to define "See" and "See also" references, and to override the default sort order for a term.

Indexing domain elements typically work with the indexterm and index-base elements; index-base is grouped with elements that are typically useful only in specialization contexts, and can be found using the link below.

Related Links

4.7.5 index-base (see page 411)

The <index-base> element allows indexing extensions to be added by specializing this element.

4.4.2.1 <indexterm>

The content of an <indexterm> element is used to produce an index entry in the generated index. You can nest <indexterm> elements to create multi-level indexes. The content is not output as part of the topic content, only as part of the index.

An <indexterm> element without the @start or @end attribute specified is interpreted as a point reference that contributes the number of the current page to an index entry; the contents of the index entry is the content of the <indexterm> element. All <indexterm> elements with the same content are "merged" to form a single index entry in the resulting index, and all contributed page numbers are included in that index entry.

In the case of nested <indexterm> elements, each <indexterm> with no <indexterm> children (the "leaves") contributes a page number to the generated index; the ancestral <indexterm> elements for each leaf <indexterm> provide the higher levels for the multilevel entry.

An <indexterm> that occurs in a topic <prolog> is interpreted as a point reference to the title of the topic. Likewise, in a DITA map, an <indexterm> that occurs in <topicmeta> inside of a <topicref> is interpreted as a point reference to the title of the referenced topic.

It is an error if an <indexterm> containing no <indexterm> children contains both an <index-see> and an <index-see-also>. (Note: <index-see> and <index-see-also> elements within <indexterm> elements that do contain <indexterm> children are ignored.) In the case of this error condition, an implementation MAY give an error message, and may (but need not) recover by treating all such <index-see> elements as <index-see-also> elements.

Note: The <index-see> and <index-see-also> elements are domain specializations of the <index-base> element, and are discussed in detail with the indexing domain.

The @start and @end attribute on <indexterm> can be used in cases where one wants to index an extended discussion that may continue over a number of pages. The start of a range is indicated by an <indexterm> with a @start attribute. The end of a range is indicated with an <indexterm> with an @end attribute whose value matches that of the @start attribute on the start-of-range <indexterm>. Such markup contributes to the generated index a page range covering all pages in the index range.

The end-of-range <indexterm> should have no content of its own; if it contains content, that content is ignored. There is no reason for the end-of-range <indexterm> to have any <indexterm> ancestors; however, an implementation should be able to handle an end-of-range <indexterm> that is nested within one or more <indexterm> elements.

The @start and @end attributes are defined as CDATA, though it is recommended that the values should not contain any whitespace characters (such as a space or tab) or control characters. Matching of @start and @end attributes is done as a character-by-character comparison with all characters significant and no case folding occurring. The @start and @end attributes are ignored if they occur on an <indexterm> element that has child <indexterm> elements.

Index range indications may occur in the <topicmeta> of a <topicref> at the map level, in the prolog of a topic, or in the body of a topic, and are interpreted as follows (see Figure 71: Index ranges (see page 338) for samples):
• In a map, the start range points to the start of the topic title of the topic being referenced by its containing <topicref>. The end range points to the end of the final child contained by the topic being referenced by its containing <topicref>, or to the end of the final topic referenced by the current map (whichever comes first). When a start and end range occur in the same <topicmeta>, the range applies to the containing <topicref> and its children.

• In the prolog of a topic, the start range points to the start of the containing topic's title. The range ends with a matching index range end in the same prolog, regardless of whether the end range is specified. The range applies to the containing topic and all its children including child relationships defined in a map.

• In the body of a topic, the range starts where the start <indexterm> occurs and ends at the matching index range end indication within the same body, or at the end of the body, whichever comes first. Such an index range does not span sub-topics of the topic.

When index ranges with the same identifier overlap, the widest range applies, and end ranges are matched with start ranges by last-in-first-out. In other words, the ranges are interpreted as nested rather than overlapping with the highest-level container taking precedence over narrower contained ranges.

As defined above, there is no such thing as an index range start that isn't terminated by either a matching end or some maximum scope. There can, however, be unmatched index range end indications; these should be ignored.

**Inheritance**

- topic/indexterm

**Example**

• The following <indexterm> is a point reference to a specific paragraph within a topic:

  <p><indexterm>databases</indexterm>Databases are used to ...</p>

• The following <indexterm> is a point reference to the start of the title of the concept:

  <concept id="db">
  <title>About databases</title>
  <prolog>
  <metadata>
  <keywords><indexterm>databases</indexterm></keywords>
  </metadata>
  </prolog>
  <body><!-- content... --></body>
  </concept>

• The following <indexterm> is a point reference to the start of the title of aboutdatabases.dita:

  <topicref href="aboutdatabases.dita">
  <topicmeta>
  <keywords><indexterm>databases</indexterm></keywords>
  </topicmeta>
  <!-- other topicref elements -->
  </topicref>

**Figure 69: Single point index terms**

The following sample represents three levels of index markup:

  <indexterm>cheese
  <indexterm>sheeps milk
  <indexterm>pecorino</indexterm>
  </indexterm>

  <indexterm>goats milk
  <indexterm>chevre</indexterm>
  </indexterm>
The previous sample is equivalent to the following sample:

```xml
<indexterm>cheese
  <indexterm>sheeps milk
    <indexterm>pecorino</indexterm>
  </indexterm>
</indexterm>
<indexterm>cheese
  <indexterm>goats milk
    <indexterm>chevre</indexterm>
  </indexterm>
</indexterm>

In each case, a generated index would include something like the this:

- cheese
  - goats milk
    - chevre 14
  - sheeps milk
    - pecorino 14

**Figure 70: Nested index terms**

A simple index range will look something like this:

```xml
<indexterm start="cheese">Cheese</indexterm>
<!-- ... additional content -->
<indexterm end="cheese"/>
```

The previous combination of terms will generate a top-level index term for "Cheese" that covers a series of pages, such as:

- Cheese 18-24

Specifying a range for nested terms is similar. In this sample, the range is specified for the tertiary index entry "pecorino":

```xml
<indexterm>cheese
  <indexterm>sheeps milk
    <indexterm start="level-3-pecorino">pecorino</indexterm>
  </indexterm>
</indexterm>
<!-- ... additional content ... -->
<indexterm end="level-3-pecorino"/>
```

The generated index for that range would look something like this:

- cheese
  - sheeps milk
    - pecorino 18-24

There are three locations that may declare a range - the body of a topic, the prolog of a topic, and a map.

- In the following example, the range begins at the start of the second paragraph, and continues to the last paragraph. If the matching end range was not included, the range would end at the end of the body element.

```xml
<topic id="accounting">
  <title>Accounting regulations</title>
  <body>
    <p>Be ethical in your accounting.</p>
    <p>Remember to do all of the following: ...<p>
    <!-- ...pages worth of rules... -->
    <p>Failure to comply will get you audited.</p>
  </body>
</topic>
```
In the following example, the range begins with the start of the topic's title, and covers the entire topic and any sub-topics. The range ends within the same prolog, regardless of whether `<indexterm end="acct"/>` is specified in the prolog.

```xml
<topic id="accounting">
  <title>Accounting regulations</title>
  <prolog>
    <metadata>
      <keywords>
        <indexterm start="acct">Accounting</indexterm>
      </keywords>
    </metadata>
  </prolog>
</topic>
```

Now assume that the topic in the previous sample is named `acct.dita`. Ranges defined in a prolog cover sub-topics, including those nested based on a map; in the following example, this means that the range covers all of `acct.dita`, as well as `procedures.dita` and `forms.dita`:

```xml
<topicref href="acct.dita">
  <topicref href="procedures.dita"/>
  <topicref href="forms.dita"/>
</topicref>
```

In the final example, the range is specified in a map. The index range for "Accounting" begins with the start of the first topic title in `acct.dita`, and covers that file as well as any sub-topics. The index range for "Government forms" begins with the start of the first topic title in `acct.dita`, and continues until the end of the last element in the file `taxfiling.dita`. If the end range for "govt" was not specified, the range would continue to the end of the map.

```xml
<topicref href="acct.dita">
  <topicmeta>
    <keywords>
      <indexterm start="acct">Accounting</indexterm>
      <indexterm end="acct"/>
      <indexterm start="govt">Government forms</indexterm>
    </keywords>
  </topicmeta>
</topicref>
<topicref href="taxfiling.dita">
  <topicmeta>
    <keywords>
      <indexterm end="govt"/>
    </keywords>
  </topicmeta>
</topicref>
```

**Figure 71: Index ranges**

```xml
<p>Einstein's most famous equation
E=mc<sup>2</sup><indexterm>E=mc<sup>2</sup></indexterm>
expresses the relationship between mass and energy.
```

All the elements in the highlighting domain are specializations of `<ph>`.

**Figure 72: Index term with `<ph>` or `<ph>` specializations**

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650), `<keyref>`, and the attributes defined below.

@start
Specifies that an index entry is positioned at the beginning of a range. This attribute is defined with the XML Data Type CDATA.

@end

Specifies that an index entry is positioned at the end of a range; value matches the @start attribute on another <indexterm>. This attribute is defined with the XML Data Type CDATA.

4.4.2.2 <indextermref>

This element is not completely defined; it is reserved for future use.

Inheritance
- topic/indextermref

Example

Examples will be added when this element is fully defined.

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and @keyref.

4.4.2.3 <index-see>

An <index-see> element within an <indexterm> element redirects the reader to another index entry that the reader should reference instead of the current one.

The <index-see> and <index-see-also> elements allow a form of redirection to another index entry within the generated index. The <index-see> element refers to an index entry that the reader should use instead of the current one, whereas the <index-see-also> element refers to an index entry that the reader should use in addition to the current one.

Processors should ignore <index-see> and <index-see-also> elements if their parent <indexterm> element contains any <indexterm> children.

Because an <index-see> indicates a redirection to use instead of the current entry, it is an error if, for any <index-see>, there is also an <index-see-also> or an <indexterm> for the same index entry (that is, another entry with an identical sort key). For example, if an <indexterm> element with the content "Memory stick" also includes <index-see>USB drive</index-see>, it is an error if there is also an <indexterm> with the contents "Memory stick". This is to prevent index entries that are both a redirect and a page reference, such as:

* Memory stick 42, 106
* See USB drive

An implementation MAY give an error message when it encounters this condition, and MAY recover from this error condition by treating the <index-see> as an <index-see-also>.

There can be multiple <index-see> elements for a single index entry.

Inheritance
+ topic/index-base-indexing-d/index-see

Example

The following example illustrates the use of an <index-see> redirection element within an <indexterm>:

```xml
<indexterm>Carassius auratus
  <index-see>Goldfish</index-see>
</indexterm>
```
This will typically generate an index entry without a page reference:

- Carassius auratus, see Goldfish

The following example illustrates the use of an `<index-see>` redirection element to a more complex (multilevel) `<indexterm>`:

```xml
<indexterm>Feeding goldfish
  <index-see>Goldfish <indexterm>feeding</indexterm></index-see>
</indexterm>
```

This is part of the indexing markup that might generate index entries such as:

- Feeding goldfish
  - see Goldfish feeding
- Goldfish
  - feeding, 56
  - flushing, 128, 345

The following example illustrates using a specialization of `<ph>` within `<index-see>`:

```xml
<indexterm>Einstein's mass and energy equation
  <index-see>E=mc\(^2\)</index-see></indexterm>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and @keyref.

### 4.4.2.4 `<index-see-also>`

An `<index-see-also>` element within an `<indexterm>` redirects the reader to another index entry that the reader should reference in addition to the current one.

The `<index-see>` and `<index-see-also>` elements allow a form of redirection to another index entry within the generated index. The `<index-see>` element refers to an index entry that the reader should use *instead of* the current one, whereas the `<index-see-also>` element refers to an index entry that the reader should use *in addition* to the current one.

Processors should ignore `<index-see>` and `<index-see-also>` elements if their parent `<indexterm>` element contains any `<indexterm>` children.

In addition to its "see also" redirection, an `<index-see-also>` functions as a pointwise index term, thereby typically generating a page reference as well as the "see also" indication.

It is not an error for there to be multiple `<index-see-also>` elements for a single index entry.

**Inheritance**

+ topic/index-base/indexing-d/index-see-also

The following example illustrates the use of an `<index-see-also>` redirection element within an `<indexterm>`:

```xml
<indexterm>Carp
  <index-see-also>Goldfish</index-see-also>
</indexterm>
```

This will typically generate a page reference to "Carp" and a redirection:

- Carp, 56
• see also Goldfish

The following example illustrates the use of an `<index-see-also>` redirection element to a more complex (multilevel) `<indexterm>`:

```xml
<indexterm>Feeding
  <index-see-also>Goldfish <indexterm>feeding</indexterm></index-see-also>
</indexterm>
```

This is part of the indexing markup that might generate index entries such as:

- Feeding, 348
  - see also Goldfish feeding
- Goldfish
  - feeding, 56
  - flushing, 128, 345

The following example illustrates using a specialization of `<ph>` within `<index-see-also>`:

```xml
<indexterm>μ = E<sub>0</sub>/V<sup>2</sup>
  <index-see-also>E=mc<sup>2</sup></index-see-also>
</indexterm>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and `@keyref`.  

### 4.4.2.5 `<index-sort-as>`

The `<index-sort-as>` element specifies a sort phrase under which an index entry would be sorted.

This element gives an author the flexibility to sort an index entry in an index differently from how its text normally would be sorted. The common use for this is to disregard insignificant leading text, such as punctuation or words like "the" or "a". For example, the author may want `<data>` to be sorted under the letter D rather than the left angle bracket (<). An author may want to include such an entry under both the punctuation heading and the letter D, in which case there can be two index entry directives differentiated only by the sort order.

Certain languages may have special sort order needs. For example, Japanese index entries might be written partially or wholly in kanji, but need to be sorted in phonetic order according to its hiragana/katakana rendition. There is no reliable automated way to map written to phonetic text: for kanji text, there can be multiple phonetic possibilities depending on the context. The only way to correctly sort Japanese index entries is to keep the phonetic counterparts with the written forms. The phonetic text would be presented as the sort order text for indexing purposes.

The `<index-sort-as>` element's content is logically augmented by the textual content of its parent `<indexterm>` element to produce the effective sort key (in other words, the textual content acts as a secondary sort field), so two `<indexterm>` elements with different content but the same `<index-sort-as>` value would never merge into a single index entry.

An `<index-sort-as>` element provides sort key information for the `<indexterm>` that is its parent; therefore, in a multiple level `<indexterm>`, the `<index-sort-as>` only affects the level in which it occurs.

It is an error if there is more than one `<index-sort-as>` child for a given `<indexterm>`. An implementation may (but need not) give an error message, and may (but need not) recover from this error condition by ignoring all but the last `<index-sort-as>`.

When located within the `<indexterm>` element, the `<sort-as>` element is equivalent to `<index-sort-as>`. It is an error for an `<indexterm>` element to directly contain both `<sort-as>` and `<index-sort-as>` elements.
Inheritance
+ topic/index-base indexing-d/index-sort-as

This is an example of an index entry for `<data>` that will be sorted as "data":

```xml
<indexterm>&lt;data&gt;&lt;index-sort-as&gt;data&lt;/index-sort-as&gt;&lt;/indexterm>
```

Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650) and `@keyref`.

Related Links

3.3.12 Sorting (see page 111)

Processors can be configured to sort elements. Common sort processing includes glossary entries that are sorted to produce automatically-generated or automatically-sorted glossary lists, lists of parameters or reference entries in custom navigation structures, tables sorted based on the contents of cells in a specific column or row, and more.

4.4.3 Delayed conref resolution elements

The delayed conref resolution domain provides several elements for use when using DITA in situations that enable delayed or run time resolution of conref. The elements allow users to resolve some conref values statically, while delaying others for later resolution.

Many publishing systems for which DITA is used as a source format do not have a way to dynamically resolve content references; those systems will not see any benefit from this element. When DITA is used for those systems, behaviors related to this element should be ignored.

4.4.3.1 `<exportanchors>`

The `<exportanchors>` element is used to delay `@conref` resolution within DITA documents. This allows you to process or display DITA content in a way that will resolve only some of the `@conref` values in that content, while remaining values are left for later resolution. The element contains a list of IDs or keys that should not be resolved during the initial preparation of the content for display; those IDs and keys will be preserved after that preparation, as will the conref relationship itself.

The `<exportanchors>` element may be used within a topic prolog, in which case the defined IDs apply to IDs within that topic (excluding sub-topics). Alternatively it may be defined in a `<topicmeta>` element in a map. In the second case the IDs apply to the single topic referenced by the current `<topicref>` element. If the `<topicref>` references a file without referencing a specific topic, it is treated as a reference to the first or root topic. In order to define anchor ids for a topic that is not the first or root topic, a `<topicref>` must directly reference the desired sub-topic.

**Note:** When an element's ID is defined for delayed resolution, it must contain only the element ID, not the usual "topicid/elementid" syntax that is required for most other DITA references. The `<anchorid>` topic explains the format in detail.

One possible way to use this is with a system that renders DITA dynamically. A user may process information locally in a way that resolves `@conref` for all static information, while delaying resolution for information that is subject to change. The `<exportanchors>` element is used to define `@conref` values that are delayed.

Another potential use is when DITA is used as the source format for a publishing system that is able to render information dynamically. In this case some `@conref` values may be resolved, while leaving pre-selected values to be resolved live in that publishing system.

Many publishing systems for which DITA is used as a source format do not have a way to dynamically resolve content references; those systems will not see any benefit from this element. When DITA is used for those systems, behaviors related to this element should be ignored.
Inheritance
+ topic/keywords delay-d/exportanchors

Example
1. Author 1 creates topics for information component A, which is a common component used by many products. The configuration task for component A is often reused in whole or in part, so the author assigns ids to each of the steps in the procedure and exports them.

```dita
<task id="configA">
<title>ABC</title>
<shortdesc>...</shortdesc>
<prelog><metadata>
<exportanchors>
<anchorid id="step1"/>
<anchorid id="step2"/>
<anchorid id="step3"/>
</exportanchors>
</metadata></prelog>
<taskbody>
<steps>
<step id="step1"><cmd>Do this</cmd></step>
<step id="step2"><cmd>Do the other</cmd></step>
<step id="step3"><cmd>And then finish</cmd></step>
</steps>
</taskbody>
</task>
```

2. Author 2 is working on information component B, which has information component A as a prerequisite.
3. Author 2 creates a configuration task that reuses two steps from the configuration task in information component A.

```dita
<task id="configB">
<title>..</title>
<shortdesc>..</shortdesc>
<taskbody>
<steps>
<step><cmd>Do the very first thing</cmd></step>
<step conref="componentA/configA.dita#configA/step1"><cmd/></step>
<step><cmd>Do the middle thing</cmd></step>
<step conref="componentA/configA.dita#configA/step2"><cmd/></step>
</steps>
</taskbody>
</task>
```

4. Author 2 builds the content for component B into a deliverable format that supports dynamic content resolution. As with traditional conref, the source for component A must be available during this process. Because the ids in configA are exported, the build process knows to preserve the reuse relationship rather than resolve it - so the @conref reference to the steps becomes an equivalent reuse artifact in that deliverable format. This way the relationship to component A can be resolved at runtime, and pick up the user's version of component A, which may be more up-to-date than the one used by Author 2 when component B was built.

Figure 73: Use case 1: Runtime resolution of @conref to an id, determined by original author

1. Author 1 is creating content that will be packaged into multiple deliverable components. In one of those components, component A, the ids should be exported for runtime reuse by other components. In other components, the ids should not be exported because all reuse is local (for example, the output is a single infocenter, or a helpset that has only one component).
2. When author 1 builds component A, the author uses a map that exports the ids, rather than exporting the ids from the topic <prolog>.

```dita
<map>
<topicref href="componentA/configA.dita">
<topicmeta>
<exportanchors>
<anchorid id="step1"/>
```
3. The rest of the use case is the same as previous - the @conref reference is passed on to the runtime/display format to deal with, rather than being resolved during native DITA processing.

**Figure 74: Use case 2: Runtime resolution to an id exported by the information builder**

The ID on an <anchorid> element is first compared with the topic's id, and then with elements inside that topic. This results in the following situation.

- The first ID to be exported is "this", which matches the topic id, so resolution of @conref values that target the topic should be delayed.
- The second value is "that", which matches a figure within the topic, so resolution of @conref values that target the figure should be delayed.
- Note that if the "this" is also used within the topic (which is legal from a DITA perspective), it will not be possible to export that id, because processors will match on the topic's id first.

**Figure 75: Delaying resolution for a topic**

**Example**

In this example, a set of information contains multiple components. Some references to component A use keys rather than a direct reference, so that @conref can be redirected to a different component when component A is not installed. The keys may be exported, in addition to the IDs, so that some references become bound to the actual component while other references may be redirected.
The @keys attributes declares two distinct keys that may be used to refer to this topic (componentAconfig and commonconfig). Only the second is preserved using <anchorkey>. A task topic from another component may reuse steps within this topic in a variety of ways.

```
<steps>
  <step conkeyref="componentAconfig/step1"><cmd/></step>
  <step conkeyref="componentAconfig/step1.5"><cmd/></step>
  <step conkeyref="commonconfig/step2"><cmd/></step>
  <step conkeyref="commonconfig/step2.5"><cmd/></step>
  <step><cmd>And that is the end of that</cmd></step>
</steps>
```

- The componentAconfig key is not preserved, so the first <step> becomes <step conref="componentA/configA.dita#configA/step1"><cmd/></step>. At that point the <anchorid> element instructs the step1 ID to be preserved; for runtime applications which support it, this relationship will be preserved in the processed DITA output.
- The second <step> with the same key becomes <step conref="componentA/configA.dita#configA/step1.5"><cmd/></step>. However, conref relationships to step1.5 are not preserved, so this conref should be resolved into static content.
- For <step> three, the map instructs that both the key commonconfig and the ID step2 should be preserved in any content generated for this DITA topic. For formats that support runtime resolution through keys, a process must convert the @conkeyref value into an equivalent value for that format.
- Although resolution for the key used in <step> four is delayed, the specific element that is referenced should not be delayed. Thus the fourth step becomes <step conref="componentA/configA.dita#configA/step2.5"><cmd/></step>. This value is then processed as an ordinary @conref value.

This allows the information assembler to resolve references that must be to componentA while deferring references that can be fulfilled by alternative component content.

- **Note:** This example demonstrates why the <anchorid> element cannot reference an element with the usual topicid/elementid format. If the two <anchorid> elements in the example had been set to config/step1 and config/step2, then they would only ever apply in a topic with id="config". It would not be possible to redirect the key to another topic, but still preserve conref behaviors as desired.

- **Note:** Although it is not specifically called out in this example, it is possible to delay @conref resolution for an entire topic using the key. If @conkeyref on a task topic element is set to "componentAconfig", which is not delayed, the @conref will be evaluated as usual. However, if @conkeyref on the task is set to "commonconfig", which is delayed, resolution of @conref on that element should be delayed by a processor.

### Attributes

The following attributes are available on this element: Universal attribute group (see page 650).

#### 4.4.3.2 <anchorid>

The <anchorid> element allows an author to define a @conref target that will be resolved dynamically when rendered for an end user of the content. This element is useful when doing an initial process of the DITA content prior to final rendering of the content; it causes specified IDs to be preserved after that process, and @conref relationships that reuse the element will not be resolved during the initial process.

When the <anchorid> element is defined within a topic prolog, the specified IDs will be found within that topic. When an <anchorid> element is defined within a <topicref> element, the specified IDs will be found within the referenced topic (if the <topicref> references a collection of topics, such as a reference that uses only a file name, the IDs will be found within the first or root topic).

The only difference between specifying an <anchorid> in the <prolog> and specifying it in the <topicmeta> is that from the map it is possible to export the ID of the entire referenced topic. If <anchorid id="zero"/> is
specified in the <topicmeta>, and the referenced topic has an id of "zero", this means that the <anchorid> is a reference to the entire topic. If the topic id is not "zero", then the <anchorid> is a reference to the element with id="zero" within that topic.

Along with the preservation of the element's ID, any @conref attribute that references the element's ID will not be resolved during an initial process. In that case, the @conref will be resolved during a later rendering process.

This description does not imply that IDs are not discarded when <anchorid> is not used; though this element requires that IDs be preserved in some manner, it is also common for IDs to be preserved when <anchorid> is not used. Thus the primary impact of the <anchorid> element is on @conref resolution.

Many publishing systems for which DITA is used as a source format do not have a way to dynamically resolve content references; those systems will not see any benefit from this element. When DITA is used for those systems, behaviors related to this element should be ignored.

Why not use topicid/elementid?

This element differs from normal DITA referencing syntax in that it may reference an element within a topic without using the topic's ID. There are two reasons for this. First, the <anchorid> element may only be defined in a situation that refers unambiguously to a single topic (in the <prolog>, or in the <topicmeta> for a reference to a topic). Second, it allows the <anchorid> to be combined with key values.

It is possible to combine an <anchorid> with a key in order to delay resolution of @conref in the topic represented by that key (see the second set of examples below). This would not be possible if the <anchorid> element required both the topic id and the element id. That is, @keyref allows a modifiable reference to a topic, so a map may instruct processors to delay conref for item "step1" in the topic represented by the key "commonconfig". If the <anchorid> element required a topic id, the delayed conref would always be bound to that specific topic.

Inheritance

+ topic/keyword delay-d/anchorid

Example

1. Author 1 creates topics for information component A, which is a common component used by many products. The configuration task for component A is often reused in whole or in part, so the author assigns ids to each of the steps in the procedure and exports them.

   <task id="configA">
   <title>ABC</title>
   <shortdesc>...</shortdesc>
   <prolog><metadata>
   <exportanchors>
   <anchorid id="step1"/>
   <anchorid id="step2"/>
   <anchorid id="step3"/>
   </exportanchors>
   </metadata></prolog>
   <taskbody>
   <steps>
   <step id="step1"><cmd>Do this</cmd></step>
   <step id="step2"><cmd>Do the other</cmd></step>
   <step id="step3"><cmd>And then finish</cmd></step>
   </steps>
   </taskbody>
   </task>

2. Author 2 is working on information component B, which has information component A as a prerequisite.
3. Author 2 creates a configuration task that reuses two steps from the configuration task in information component A.

   <task id="configB">
   <title>..</title>
   <shortdesc>..</shortdesc>
   <taskbody>
   <steps>
   <step id="step1"><cmd>Do this</cmd></step>
   <step id="step2"><cmd>Do the other</cmd></step>
   <step id="step3"><cmd>And then finish</cmd></step>
   </steps>
   </taskbody>
   </task>
4. Author 2 builds the content for component B into a deliverable format that supports dynamic content resolution. As with traditional conref, the source for component A must be available during this process. Because the ids in configA are exported, the build process knows to preserve the reuse relationship rather than resolve it - so the \texttt{@conref} reference to the steps becomes an equivalent reuse artifact in that deliverable format. This way the relationship to component A can be resolved at runtime, and pick up the user's version of component A, which may be more up-to-date than the one used by Author 2 when component B was built.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure76.png}
\caption{Use case 1: Runtime resolution of \texttt{@conref} to an id, determined by original author}
\end{figure}

1. Author 1 is creating content that will be packaged into multiple deliverable components. In one of those components, component A, the ids should be exported for runtime reuse by other components. In other components, the ids should not be exported because all reuse is local (for example, the output is a single infocenter, or a helpset that has only one component).

2. When author 1 builds component A, the author uses a map that exports the ids, rather than exporting the ids from the topic \texttt{<prolog>}

\begin{verbatim}
<map>
  <topicref href="componentA/configA.dita">
    <topicmeta>
      <exportanchors>
        <anchorid id="step1"/>
        <anchorid id="step2"/>
        <anchorid id="step3"/>
      </exportanchors>
    </topicmeta>
  </topicref>
</map>
\end{verbatim}

3. The rest of the use case is the same as previous - the \texttt{@conref} reference is passed on to the runtime/display format to deal with, rather than being resolved during native DITA processing.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure77.png}
\caption{Use case 2: Runtime resolution to an id exported by the information builder}
\end{figure}

The ID on an \texttt{<anchorid>} element is first compared with the topic's id, and then with elements inside that topic. This results in the following situation.

\begin{verbatim}
<map>
  <topicref href="componentA/this.dita">
    <topicmeta>
      <exportanchors>
        <anchorid id="this"/>
        <anchorid id="that"/>
      </exportanchors>
    </topicmeta>
  </topicref>
</map>

<topic id="this">
  <title>This and that</title>
  <shortdesc>Oh, you know, this and that.</shortdesc>
  <body>
    <fig id="that"><p>more of that</p></fig>
  </body>
</topic>
\end{verbatim}

\begin{itemize}
  \item The first ID to be exported is "this", which matches the topic id, so resolution of \texttt{@conref} values that target the topic should be delayed.
  \item The second value is "that", which matches a figure within the topic, so resolution of \texttt{@conref} values that target the figure should be delayed.
\end{itemize}
• Note that if the "this" is also used within the topic (which is legal from a DITA perspective), it will not be possible to export that id, because processors will match on the topic's id first.

*Figure 78: Delaying resolution for a topic*

**Example**

In this example, a set of information contains multiple components. Some references to component A use keys rather than a direct reference, so that @conref can be redirected to a different component when component A is not installed. The keys may be exported, in addition to the IDs, so that some references become bound to the actual component while other references may be redirected.

```xml
<map>
  <topicref keys="componentAconfig commonconfig"
            href="componentA/configA.dita#configA">
    <topicmeta>
      <exportanchors>
        <anchorkey keyref="commonconfig"/>
        <anchorid id="step1"/>
        <anchorid id="step2"/>
      </exportanchors>
    </topicmeta>
  </topicref>
</map>
```

The @keys attributes declares two distinct keys that may be used to refer to this topic (componentAconfig and commonconfig). Only the second is preserved using <anchorkey>. A task topic from another component may reuse steps within this topic in a variety of ways.

```xml
<steps>
  <step conkeyref="componentAconfig/step1"><cmd/></step>
  <step conkeyref="componentAconfig/step1.5"><cmd/></step>
  <step conkeyref="commonconfig/step2"><cmd/></step>
  <step conkeyref="commonconfig/step2.5"><cmd/></step>
  <step><cmd>And that is the end of that</cmd></step>
</steps>
```

• The componentAconfig key is not preserved, so the first <step> becomes <step conref="componentA/configA.dita#configA/step1"><cmd/></step>. At that point the <anchorid> element instructs the step1 ID to be preserved; for runtime applications which support it, this relationship will be preserved in the processed DITA output.

• The second <step> with the same key becomes <step conref="componentA/configA.dita#configA/step1.5"><cmd/></step>. However, conref relationships to step1.5 are not preserved, so this conref should be resolved into static content.

• For <step> three, the map instructs that both the key commonconfig and the ID step2 should be preserved in any content generated for this DITA topic. For formats that support runtime resolution through keys, a process must convert the @conkeyref value into an equivalent value for that format.

• Although resolution for the key used in <step> four is delayed, the specific element that is referenced should not be delayed. Thus the fourth step becomes <step conref="componentA/configA.dita#configA/step2.5"><cmd/></step>. This value is then processed as an ordinary @conref value.

This allows the information assembler to resolve references that must be to componentA while deferring references that can be fulfilled by alternative component content.

**Note:** This example demonstrates why the <anchorid> element cannot reference an element with the usual topicid/elementid format. If the two <anchorid> elements in the example had been set to config/step1 and config/step2, then they would only ever apply in a topic with id="config". It would not be possible to redirect the key to another topic, but still preserve conref behaviors as desired.
Note: Although it is not specifically called out in this example, it is possible to delay `@conref` resolution for an entire topic using the key. If `@conkeyref` on a task topic element is set to "componentAconfig", which is not delayed, the `@conref` will be evaluated as usual. However, if `@conkeyref` on the task is set to "commonconfig", which is delayed, resolution of `@conref` on that element should be delayed by a processor.

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of `@id`, given below), `@keyref`, and `outputclass` (see page 0). `@id (REQUIRED)`

Indicates an ID within the specific topic that will be preserved during processing. Any `@conref` values referencing the indicated ID will not be resolved; when possible, the original relationship should be preserved in any processed document. Note that this element creates an exception to the general rules that IDs may only be used once within a single topic or within a map; this is because the ID is actually a pointer to another target, rather than being a target itself. This attribute is defined with the XML Data Type CDATA.

4.4.3.3 `<anchorkey>`

The `<anchorkey>` element allows an author to define a `@conref` target that will be resolved dynamically when rendered for an end user of the content. This element is useful when doing an initial process of the DITA content prior to final rendering of the content; it allows specified keys to be preserved after that process, and `conref` relationships which use that key will not be resolved during that initial process.

When a `@keyref` attribute is specified on an `<anchorkey>` element, it indicates that any `conref` relationships using that key will not be resolved. Applications that support run-time resolution of `@conref` with keys will then be able to dynamically resolve this `@conref` at display time.

There is no difference between specifying `<anchorkey>` within a map (in `<topicmeta>`) and specifying `<anchorkey>` within a topic. In both cases, processors are instructed to delay resolution of that key for the current set of information. However, the best practice is to only use `<anchorkey>` within a map. If it is specified in a topic, that topic will define a usage for the key for every user of that topic. This makes the topic less portable, because users that do not want to delay resolution of that specific key will not be able to include the topic in their information.

Many publishing systems for which DITA is used as a source format do not have a way to dynamically resolve content references; those systems will not see any benefit from this element. When DITA is used for those systems, behaviors related to this element should be ignored.

Inheritance

+ topic/keyword delay-d/anchorkey

Example

In this example, a set of information contains multiple components. Some references to component A use keys rather than a direct reference, so that `@conref` can be redirected to a different component when component A is not installed. The keys may be exported, in addition to the IDs, so that some references become bound to the actual component while other references may be redirected.

```xml
<map>
  <topicref keys="componentAconfig commonconfig"
            href="componentA/configA.dita#configA">
    <topicmeta>
      <exportanchors>
        <anchorkey keyref="commonconfig"/>
        <anchorid id="step1"/>
        <anchorid id="step2"/>
      </exportanchors>
    </topicmeta>
  </topicref>
</map>
```
The `@keys` attributes declares two distinct keys that may be used to refer to this topic (componentAconfig and commonconfig). Only the second is preserved using `<anchorkey>`. A task topic from another component may reuse steps within this topic in a variety of ways.

```dita
<steps>
  <step conkeyref="componentAconfig/step1"><cmd/></step>
  <step conkeyref="componentAconfig/step1.5"><cmd/></step>
  <step conkeyref="commonconfig/step2"><cmd/></step>
  <step conkeyref="commonconfig/step2.5"><cmd/></step>
  <step><cmd>And that is the end of that</cmd></step>
</steps>
```

- The componentAconfig key is not preserved, so the first `<step>` becomes `<step conref="componentA/configA.dita#configA/step1"><cmd/></step>`. At that point the `<anchorid>` element instructs the step1 ID to be preserved; for runtime applications which support it, this relationship will be preserved in the processed DITA output.
- The second `<step>` with the same key becomes `<step conref="componentA/configA.dita#configA/step1.5"><cmd/></step>`. However, conref relationships to step1.5 are not preserved, so this conref should be resolved into static content.
- For `<step>` three, the map instructs that both the key commonconfig and the ID step2 should be preserved in any content generated for this DITA topic. For formats that support runtime resolution through keys, a process must convert the `@conkeyref` value into an equivalent value for that format.
- Although resolution for the key used in `<step>` four is delayed, the specific element that is referenced should not be delayed. Thus the fourth step becomes `<step conref="componentA/configA.dita#configA/step2.5"><cmd/></step>`. This value is then processed as an ordinary `@conref` value.

This allows the information assembler to resolve references that must be to componentA while deferring references that can be fulfilled by alternative component content.

**Note:** This example demonstrates why the `<anchorid>` element cannot reference an element with the usual topicid/elementid format. If the two `<anchorid>` elements in the example had been set to config/step1 and config/step2, then they would only ever apply in a topic with id="config". It would not be possible to redirect the key to another topic, but still preserve conref behaviors as desired.

**Note:** Although it is not specifically called out in this example, it is possible to delay `@conref` resolution for an entire topic using the key. If `@conkeyref` on a task topic element is set to "componentAconfig", which is not delayed, the `@conref` will be evaluated as usual. However, if `@conkeyref` on the task is set to "commonconfig", which is delayed, resolution of `@conref` on that element should be delayed by a processor.

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), `outputclass` (see page 0), and the attribute defined below.

**@keyref (REQUIRED)**

Defines a key that, when possible, should be preserved in content generated from the DITA source material. Conref relationships that use this key should not be resolved when generating that material, so that `@conref` may be resolved at run-time when an end user is reading the content. This attribute is defined with the XML Data Type NMTOKEN.
4.5 Domain elements

General purpose domains are not specific to any type of information, such as the hazard statement domain that provides elements for describing hazardous situations.

4.5.1 Hazard statement elements

The hazard statement domain elements are used to provide information about product safety hazards. The domain can be included in any topic type or map. Its elements are used to inform readers about potential hazards, consequences, and avoidance strategies.

4.5.1.1 <hazardstatement>

The <hazardstatement> element contains hazard warning information. It is based on the regulations of ANSI Z535 and ISO 3864. It enables the author to select the type of hazard, add information about the specific hazard and how to avoid it, and add one or more safety symbols.

Inheritance

+ topic/note hazard-d/hazardstatement

Example

Danger: Indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.

```xml
<hazardstatement type="danger">
  <messagepanel>
    <typeofhazard>Rotating blade.</typeofhazard>
    <consequence>Moving parts can crush and cut.</consequence>
    <howtoavoid>Follow lockout procedure before servicing.</howtoavoid>
  </messagepanel>
  <hazardsymbol href="rotatingblade.png"/>
</hazardstatement>
```

Warning: Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.

```xml
<hazardstatement type="warning">
  <messagepanel>
    <typeofhazard>Hot surfaces inside.</typeofhazard>
    <consequence>Contact may cause burn.</consequence>
    <howtoavoid>Wear protective gear before servicing internal parts.</howtoavoid>
  </messagepanel>
  <hazardsymbol href="hotsurface.png"/>
</hazardstatement>
```

Caution: Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury.

```xml
<hazardstatement type="caution">
  <messagepanel>
    <typeofhazard>Lifting Hazard.</typeofhazard>
    <consequence>May result in injury.</consequence>
    <howtoavoid>See safety manual for lifting instructions.</howtoavoid>
  </messagepanel>
  <hazardsymbol href="heavy.png"/>
</hazardstatement>
```

Notice: Indicates a potential situation which, if not avoided, may result in property damage or in an undesirable result or state.

```xml
<hazardstatement type="notice">
  <messagepanel>
    <typeofhazard>Battery low</typeofhazard>
    <howtoavoid>Push and hold for charge state test.</howtoavoid>
  </messagepanel>
</hazardstatement>
```
Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), *spectitle* (see page 0), and the attributes defined below.

@type

Describes the level of hazard. Safety hazard level definitions correspond to the same values in the ANSI Z535 and the ISO 3864 standards. Note that this differs from the @type attribute on many other DITA elements. See *The type attribute* (see page 674) for detailed information on supported values and processing implications. Available values are note, tip, fastpath, restriction, important, remember, attention, caution, notice, danger, warning, other, and -dita-use-conref-target.

Comment by robander
Waiting for TC input to see whether @type=trouble should be added here as it was on <note>. Local definition with the reference to ANSI still differs from the base <note> element.

Comment by robander, 12 February 2014
Update: TC agreed not ot add trouble, most of these values should not hvae been added but are left now for backwards compatibility

@othertype

Indicates an alternate note type, when the type is not available in the @type attribute value list. This value is used as the user-provided note title when the @type attribute value is set to "other.". This attribute is defined with the XML Data Type CDATA.

4.5.1.2 <consequence>

The <consequence> element specifies the consequence of failing to avoid a hazard, for example, "Contact may cause burn."

Inheritance

+ topic/li hazard-d/consequence

Example

```xml
<hazardstatement type="warning">
  <messagepanel>
    <typeofhazard>Hot surfaces inside.</typeofhazard>
    <consequence>Contact may cause burn.</consequence>
    <howtoavoid>Wear protective gear before servicing internal parts.</howtoavoid>
  </messagepanel>
</hazardstatement>
```
4.5.1.3 <hazardsymbol>

The <hazardsymbol> element specifies a graphic. The graphic might represent a hazard, a hazardous situation, a result of not avoiding a hazard, or any combination of these messages.

Inheritance
+ topic/image hazard-d/hazardsymbol

Example

```xml
<hazardstatement type="danger">
  <messagepanel>
    <typeofhazard>Rotating blade.</typeofhazard>
    <consequence>Moving parts can crush and cut.</consequence>
    <howtoavoid>Follow lockout procedure before servicing.</howtoavoid>
  </messagepanel>
  <hazardsymbol href="rotatingblade.png"/>
</hazardstatement>
```

Attributes

Comment by robander
Note that this is an exact copy of the attribute list for <image>, EXCEPT that this element dropped the deprecated @alt and makes @href required, which makes it harder to reuse that full list. It is an exact copy of glossSymbol.

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), @keyref, and the attributes defined below.

@href (REQUIRED)
Provides a reference to the image. See The href attribute (see page 663) for detailed information on supported values and processing implications. This attribute is defined with the XML Data Type CDATA.

@scope
The @scope attribute identifies the closeness of the relationship between the current document and the target resource. Allowable values are local, peer, external, and -dita-use-conref-target. See The scope attribute (see page 677) for more information on values.

@height
Indicates the vertical dimension for the resulting image display. If necessary, the image is scaled to the specified size. The value of this attribute is a real number (expressed in decimal notation) optionally followed by a unit of measure from the set of pc, pt, px, in, cm, mm, em (picas, points, pixels, inches, centimeters, millimeters, and ems respectively). The default unit is px (pixels). Possible values include: "5", "5in", and "10.5cm". If a height value is specified and no width value is specified, the width will be scaled by the same factor as the height. If both a height value and width value are specified, some implementations may not be able to scale the two directions by a different factor and may therefore ignore one of the two values. This attribute is defined with the XML Data Type NMTOKEN.

@width
Indicates the horizontal dimension for the resulting image display. If necessary, the image is scaled to the specified size. The value of this attribute is a real number (expressed in decimal notation) optionally followed by a unit of measure from the set of pc, pt, px, in, cm, mm, em (picas, points, pixels, inches, centimeters, millimeters, and ems respectively). The default unit is px (pixels). Possible values include: "5", "5in", and "10.5cm". If a width value is specified and no height value is specified, the height will be scaled by the same factor as the width. If both a height value and width value are specified, some implementations may not be able to scale the two directions by a different factor and may therefore ignore one of the two values. This attribute is defined with the XML Data Type NMTOKEN.

@align
Controls the horizontal alignment of an image when placement is specified as "break". Common values include left, right, and center. This attribute is defined with the XML Data Type CDATA.

@scale
Specifies a percentage 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 this image's @height or @width attribute (or both), the @scale attribute is ignored. This attribute is defined with the XML Data Type NMTOKEN.

It is an error if the value of this attribute is not an unsigned integer. In this case, the implementation may (but need not) give an error message and may (but need not) recover by ignoring this attribute.

@scalefit
Allow an image to be scaled up or down to fit within available space. Allowable values are yes, no, and -dita-use-conref-target. If, for a given image, any one of @height, @width, or @scale is specified, those attributes determine the graphic size, and any setting of @scalefit is ignored. If none of those attributes are specified and scalefit="yes", then the image is scaled (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 the prevailing column (or table cell) width--that is, the width a paragraph of text would have if the graphic were a paragraph instead. 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.

@placement
Indicates whether an image should be displayed inline or separated from the surrounding text. The default is inline. Allowable values are: inline, break, or and -dita-use-conref-target.

Comment by robander
Nov 27 2013: DITA 1.2 said "The processing default is inline". But the "Default" column listed "inline" as a true default, and this has been the DTD and Schema implementation. I've removed the word "processing" from the previous definition.

@longdesc (DEPRECATED)
A reference to a textual description of the graphic or object. This attribute supports creating accessible content. See The href attribute (see page 663) for detailed information on supported values and processing implications. For examples of how this attribute is used in output, see this topic on long descriptions. NOTE: This attribute is deprecated in favor of the longdesc (see page 272) subelement to this element. This attribute is defined with the XML Data Type CDATA.

4.5.1.4 <howtoavoid>

The <howtoavoid> element contains information about how a user can avoid a hazard, for example, "Do not use solvents to clean the drum surface."

Inheritance
+ topic/li hazard-d/howtoavoid

Example

```xml
<hazardstatement type="notice">
  <messagepanel>
    <typeofhazard>Machinery Damage</typeofhazard>
    <howtoavoid>
      <sl>
        <sli>Do NOT use solvents to clean the drum surface</sli>
        <sli>Read manual for proper drum cleaning procedure</sli>
      </sl>
    </howtoavoid>
  </messagepanel>
  <hazardsymbol href="readmanual.png"/>
</hazardstatement>
```
Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.5.1.5 `<messagepanel>`

The `<messagepanel>` element contains the textual information that is displayed on the hazard statement. This information identifies the hazard, specifies how to avoid the hazard, and states the probable consequences of failing to avoid the hazard.

Inheritance
+ topic/ul hazard-d/messagepanel

Example

```xml
<hazardstatement type="caution">
  <messagepanel>
    <typeofhazard>Lifting Hazard.</typeofhazard>
    <consequence>May result in injury.</consequence>
    <howtoavoid>See safety manual for lifting instructions.</howtoavoid>
  </messagepanel>
  <hazardsymbol href="heavy.png"></hazardsymbol>
</hazardstatement>
```

Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650), *compact* (see page 0), *outputclass* (see page 0), and *spectitle* (see page 0).

4.5.1.6 `<typeofhazard>`

The `<typeofhazard>` element contains a description of the type of hazard, for example, "Hot surfaces inside."

Inheritance
+ topic/li hazard-d/typeofhazard

Example

```xml
<hazardstatement type="caution">
  <messagepanel>
    <typeofhazard>Lifting Hazard.</typeofhazard>
    <consequence>May result in injury.</consequence>
    <howtoavoid>See safety manual for lifting instructions.</howtoavoid>
  </messagepanel>
  <hazardsymbol href="heavy.png"></hazardsymbol>
</hazardstatement>
```

Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).
4.5.2 Highlighting elements

The highlighting elements are used to highlight text with styles (such as bold, italic, and monospace). Never use these elements when a semantically specific element is available. These elements are not intended for use by specializers, and are intended solely for use by authors when no semantically appropriate element is available and a formatting effect is required.

4.5.2.1 <b>

The <b> element is used to apply bold highlighting to the content of the element. This element is part of the DITA highlighting domain. Use this element only when a more semantically appropriate element is not available. For example, for specific items such as GUI controls, use the <uicontrol> element.

Inheritance
+ topic/ph hi-d/b

Example

```
<p><b>STOP!</b> This is <b>very</b> important!</p>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.5.2.2 <i>

The <i> element is used to apply italic highlighting to the content of the element. This element is part of the DITA highlighting domain. Use this element only when a more semantically appropriate element is not available. For example, for specific items such as variable names, use the <varname> element.

Inheritance
+ topic/ph hi-d/i

Example

```
<p>Unplug the unit <i>before</i> placing the metal screwdriver against the terminal screw.</p>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.5.2.3 <sup>

The <sup> element indicates that text should appear with superscript highlighting, or vertically raised in relationship to the surrounding text. Superscripts are usually a smaller font than the surrounding text. This element is part of the DITA highlighting domain. Use this element only when a more semantically appropriate element is not available.

Inheritance
+ topic/ph hi-d/sup
Example

The power produced by the electrohydraulic dam was $10^{10}$ more than the older electric plant. The difference was $H_2O$.

**Attributes**

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.5.2.4 `<sub>`

The `<sub>` element indicates that text should appear with subscript highlighting, or placed lower in relationship to the surrounding text. Subscripted text is often a smaller font than the surrounding text. Formatting may vary depending on your output process. This element is part of the DITA highlighting domain. Use this element only when a more semantically appropriate element is not available.

**Inheritance**

+ topic/ph hi-d/sub

Example

The power produced by the electrohydraulic dam was $10^{10}$ more than the older electric plant. The difference was $H_{2O}$.

**Attributes**

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.5.2.5 `<tt>`

The `<tt>` (teletype) element is used to apply monospaced highlighting to the content of the element. This element is part of the DITA highlighting domain. Use this element only when a more semantically appropriate element is not available. For example, for specific items such as inline code fragments, use the `<codeph>` element.

**Inheritance**

+ topic/ph hi-d/tt

Example

Make sure that the screen displays `<tt>File successfully created</tt>` before proceeding to the next stage of the task.

While the previous example demonstrates a potential use of `<tt>`, it might be more properly encoded using `<systemoutput>`.

**Attributes**

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).
4.5.2.6 `<u>`

The `<u>` element is used to apply underline highlighting to the content of the element. This element is part of the DITA highlighting domain. Use this element only when a more semantically appropriate element is not available. For example, for specific items such as GUI controls, use the `<uicontrol>` element.

**Inheritance**

+ topic/ph hi-d/u

**Example**

```xml
<p>Beware: `<u>overuse</u>` `<i>of</i>` `<b>highlighting</b>` is sometimes known as font-itis!</p>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.5.2.7 `<line-through>`

The `<line-through>` element indicates text that should be rendered with a line struck through the content. This element is designed to enable authors to indicate a deletion or revision for rhetorical purpose; it is not intended to be used for indicating revisions or changes made as a side effect of authoring activity. This element is part of the highlighting domain. Use this element only when a more semantically appropriate element is not available.

**Comment by Kristen Eberlein, 26 January 2014**

I think the short description needs revising. What do we mean by “as a side effect of authoring activity”?

**Inheritance**

+ topic/ph hi-d/line-through

**Example**

```xml
<p>Line-through: DITA technology can be `<line-through>maddening</line-through>` a challenge to implement.</p>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.5.2.8 `<overline>`

The `<overline>` element indicates content that should be rendered with a line above it. This element is part of the highlighting domain. Use this element only when a more semantically appropriate element is not available.

**Inheritance**

+ topic/ph hi-d/overline
Example

Overline: \(<overline><i>x</i></overline>\) is the average value of\(<i>x_{i}</i>\)/i</overline>/i</p>

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.5.3 Utilities elements

The utilities domain elements represent common features of a language that may not necessarily be semantic, such as image maps.

4.5.3.1 <area>

The <area> element describes a linkable area within an <imagemap>. It allows the author to specify a shape within the image, the coordinates of that shape, and a link target for the area.

Inheritance

+ topic/figgroup ut-d/area

Example

```
<area>
  <shape>rect</shape>
  <coords>54,1,117,60</coords>
  <xref href="d1-s2.dita"></xref>
</area>
```

A more complete example is located in the description for *imagemap* (see page 361).

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.5.3.2 <coords>

The <coords> element specifies the coordinates of a linkable region in an <imagemap>.

This element contains text data representing coordinate data for image maps. Pixels are the recommended units for describing coordinates. The syntax of the coordinate data depends on the shape described by the coordinates, and is based on the image map definition in HTML. It uses the following data for the appropriate shapes:

Shape | Data format
------|----------------
rect   | left-x, top-y, right-x, bottom-y
circle | center-x, center-y, radius
poly   | x1, y1, x2, y2, ..., xN, yN. The first x and y coordinate pair and the last should be the same to close the polygon.

Inheritance

+ topic/ph ut-d/coords
Example

```xml
<area>
  <shape>rect</shape>
  <coords>54,1,117,60</coords>
  <xref href="d1-s2.dita"></xref>
</area>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @translate, given below), outputclass (see page 0 ), and @keyref.

@translate

Indicates whether the content of the element should be translated or not. For this element the default value is "no". Setting to "yes" will override the default. The DITA architectural specification contains a list of each OASIS DITA element and its common processing default for the translate value; because this element uses an actual default, it will always be treated as translate="no" unless overridden as described. Available values are:

no

The content of this element is not translateable.

yes

The content of this element is translateable.

-dita-use-conref-target

See Using the -dita-use-conref-target value (see page 666) for more information.

4.5.3.3 <imagemap>

The `<imagemap>` element supports the basic functionality of the HTML "client-side" image map markup. `<imagemap>` allows you to designate a linkable area or region over an image, allowing a link in that region to display another topic.

An HTML client-side image map binds an image to the navigation structure (the "map") by means of an ID association from the map to the image. In contrast, the DITA version of `<imagemap>` markup simply includes the target image as the first required element in the markup, followed by a sequence of `<area>` elements that represent the links associated with the contained image.

An `<imagemap>` structure can be output either to a standard HTML image map or to alternative forms of navigation (such as table-based image maps). When output as PDF, the minimal form would be to represent at least the image; advanced PDF output processors should be able to provide equivalent region-oriented hyperlinks.

The `<xref>` content within `<area>` contains the intended alternative text or hover text for that image map area.

Inheritance

+ topic/fig ut-d/imagemap

Example

A simple `<imagemap>` looks like this (note that the rendering will depend on how this markup is supported for particular output formats):

```xml
<imagemap>
  <image href="imagemapworld.jpg">
    <alt>Map of the world showing 5 areas</alt>
  </image>
  <area><shape>rect</shape><coords>2,0,53,59</coords>
    <xref href="d1-s1.dita">Section 1 alternative text</xref>
  </area>
</imagemap>
```
The areas defined correspond to this graphic image with the areas visible:

The values for use in the `<shape>` and `<coords>` elements must follow the guidelines defined for image maps in HTML 4.1, *Client-side image maps: the MAP and AREA elements*.

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Display attribute group* (see page 660), *outputclass* (see page 0), and *spectitle* (see page 0).

**4.5.3.4 `<shape>`**

The `<shape>` element defines the shape of a linkable area in an `<imagemap>`.

The `<shape>` element supports these values:

- **rect**
  
  Define a rectangular region. If you leave the `<shape>` element blank, a rectangular shape is assumed.

- **circle**
  
  Define a circular region.

- **poly**
  
  Define a polygonal region.

- **default**
  
  Indicates the entire diagram.
Inheritance
+ topic(keyword ut-d/shape

Example

```xml
<area>
  <shape>rect</shape>
  <coords>54,1,117,60</coords>
  <xref href="d1-s2.dita"></xref>
</area>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of @translate, given below), *outputclass* (see page 0), and @keyref.

@translate

Indicates whether the content of the element should be translated or not. For this element the default value is "no". Setting to "yes" will override the default. The DITA architectural specification contains a list of each OASIS DITA element and its common processing default for the translate value; because this element uses an actual default, it will always be treated as translate="no" unless overridden as described. Available values are:

- no
  The content of this element is not translateable.

- yes
  The content of this element is translateable.

@dita-use-conref-target

See *Using the -dita-use-conref-target value* (see page 666) for more information.

4.5.3.5 <sort-as>

For elements that are sorted, the <sort-as> element provides text that is combined with the base sort phrase to construct the effective sort phrase. The text can be specified in the content of the <sort-as> element or in the @value attribute on the <sort-as> element. The <sort-as> element is useful for elements where the base sort phrase is inadequate or non-existent, for example, a glossary entry for a Japanese Kanji phrase.

The <sort-as> element can contain <text> and <keyword> elements in order to enable content referencing. If a <keyword> element is used within <sort-as>, the @keyref attribute can be used to set the sort phrase. If a <keyword> uses @keyref and would otherwise also act as a navigation link, the link aspect of the @keyref attribute is ignored.

As a specialization of <data>, the <sort-as> element is allowed in any context where <data> is allowed. However, the presence of <sort-as> within an element does not, by itself, indicate that the containing element should be sorted. Processors can choose to sort any DITA elements for any reason. Likewise, processors are not required to sort any elements. See *Sorting* (see page 111) for more information on sorting.

Some elements in the base DITA vocabulary are natural candidates for sorting, including topics, definition list entries, and rows in tables and simple tables. Authors are likely to include <sort-as> elements in the following locations:

- For topics, the <sort-as> element can be included directly in <title>, <searchtitle>, or <navtitle> when the different forms of title need different effective sort phrases. If the effective sort phrase is common to all the titles for a topic, the <sort-as> element can be included as a direct child of the topic prolog anywhere <data> is allowed.

- For glossary entry topics, the <sort-as> element can be included directly in <glossterm> or as a direct child of <prolog>.
• For topic references, the `<sort-as>` element can be included directly in the `<navtitle>` or `<title>` element within `<topicmeta>` or as a child of `<topicmeta>`.

• For definition list items, include the `<sort-as>` element in the `<dt>` element.

Processors SHOULD expect to encounter `<sort-as>` elements in the above locations and process them correctly, including the following considerations and precedence rules:

Comment by robander, 9 April 2014
Adding in some discussion from phase1 review, for potential consideration in later editing. Comments from Dick Hamilton:

It seems to me that processors MUST expect to encounter `<sort-as>` elements. Maybe it's okay to say that processors SHOULD be able to process them correctly, but given the statement after the list that "processors that sort the containing element MUST construct ....," I'm not sure that's safe.

Bottom line is that I don't see why processors shouldn't be required to handle this element as specified.

Comment from Kris Eberlein: "Point for TC to consider before changing the normative language."

Comments from Robert Anderson:

I'm not sure this is a MUST. Odd reasoning: in general, compliant processors must expect to encounter all elements, and that doesn't need to be said. This is an element that will very often make both authors and implementers say "why the heck is this available here?" If we say that they MUST support in all of these locations, that would imply to me that processors MUST support sorting of all of those elements, which may not make sense for a given processor. For example, I don't think that a DITA editor or rendering tool MUST support sorting of definition lists, which means we cannot say that processors MUST support the element inside of definition lists according to the following rules.

Similarly, the text you've quoted after the list has a major caveat in the "processors that sort" phrase -- if you sort, you MUST use the sort order, but if you don't, then this doesn't apply and you are still compliant.

This isn't to say the current (complete) wording is perfect, just that I don't think this is a MUST.

• A `<sort-as>` element that is specified in a title takes precedence over a `<sort-as>` element that is specified as a child of the topic prolog.

• Except for instances in the topic prolog, processors should only apply `<sort-as>` elements that are either a direct child of the element to be sorted or a direct child of the title- or label-defining element of the element to be sorted.

• When an element contains multiple, direct-child, `<sort-as>` elements, the first direct-child `<sort-as>` element in document order takes precedence.

• When located within the `<indexterm>` element, the `<sort-as>` element is equivalent to `<index-sort-as>`.

• Sort phrases are determined after filtering and content reference resolution is applied.
When a `<sort-as>` element is specified, processors that sort the containing element **MUST** construct the effective sort phrase by prepending the content of the `<sort-as>` element to the base sort phrase. This ensures that two items with the same `<sort-as>` element but different base sort phrases will sort in the appropriate order.

For example, if a processor uses the content of the `<title>` element as the base sort phrase, and the title of a topic is "24 Hour Support Hotline" and the value of the `<sort-as>` element is "twenty-four hour", then the effective sort phrase would be "twenty-four hour24 Hour Support Hotline".

**Inheritance**

+ topic/data ut-d/sort-as

**Example**

The following examples illustrate how a glossary entry for the Chinese ideographic character for "big" might specify an effective sort phrase of "big big":

```
<br>Figure 79: The `<sort-as>` element located within `<glossterm>`
<br>
<glossentry id="gloss-dada">
  <glossterm>&#x5927;&#x5927;</glossterm>
  <prolog>
    <sort-as>dada</sort-as>
  </prolog>
  <glossdef>Literally "big big".</glossdef>
</glossentry>
```

```
<br>Figure 80: The `<sort-as>` element within `<prolog>`
<br>
<glossentry id="gloss-dada">
  <sort-as>dada</sort-as>
  <glossterm>&#x5927;&#x5927;</glossterm>
  <glossdef>Literally "big big".</glossdef>
</glossentry>
```

**Attributes**

The following attributes are available on this element: Universal attribute group (see page 650) and the attributes defined below.

@name

Names the metadata item that the element represents. The default value is "sort-as". Specializations of `<sort-as>` can set the default value of the @name attribute to reflect the tag name of the specialized element. This attribute is defined with the XML Data Type CDATA.

@value

The value of the metadata item. When the `<sort-as>` element has content and the @value attribute is specified, the @value attribute takes precedence. If the @value attribute is not specified and the `<sort-as>` element does not contain content, then the `<sort-as>` element has no effect. This attribute is defined with the XML Data Type CDATA.

**Related Links**

3.3.12 Sorting (see page 111)

Processors can be configured to sort elements. Common sort processing includes glossary entries that are sorted to produce automatically-generated or automatically-sorted glossary lists, lists of parameters or reference entries in custom navigation structures, tables sorted based on the contents of cells in a specific column or row, and more.
4.5.4 DITAVAL reference domain

The DITAVAL reference domain is used to reference a DITAVAL file that contains conditions that apply only to a subset of a DITA map. It also can be used to replicate a subset of a DITA map for multiple audiences.

4.5.4.1 <ditavalref>

The <ditavalref> element references a DITAVAL document that specifies filter conditions that can be used to process a subset of the map content. Other DITAVAL-reference domain elements can be used to imply multiple copies of the map branch that contains them and so apply a different set of conditions to each copy.

Comment by Kristen Eberlein, 5 February 2014
In the short description, we might want to consider standardizing use of branch terminology.

When a <ditavalref> element is included in a map, the conditions in the referenced DITAVAL document are used to filter the elements in the branch. The branch includes the parent element that contains the <ditavalref> element, any child elements, and all resources that are referenced by the parent element or its children. While there is no technical restriction that forces <ditavalref> to appear before peer topic references, placing them first is considered a best practice and all examples in the specification will use this convention.

In the simple case, a map can use <ditavalref> as follows:

```xml
<map>
  <topicref href="sampleBranch.dita" audience="admin">
    <topicmeta>
      <navtitle>Navigation title for branch</navtitle>
    </topicmeta>
    <ditavalref href="conditions.ditaval"/>
    <topicref href="insideBranch.dita" platform="win linux mac"/>
  </topicref>
  <!-- Other branches not affected by conditions.ditaval -->
</map>
```

The filtering conditions specified in the conditions.ditaval file apply to the following:

- The <topicref> element that references sampleBranch.dita and all child elements: <topicmeta>, <navtitle>, and <topicref> elements
- The sampleBranch.dita topic
- The insideBranch.dita topic

When more than one <ditavalref> element is specified in the same branch at the same level, the effective result is one copy of the branch for each <ditavalref> element. If the example above contains a reference to otherConditions.ditaval as a peer to the existing <ditavalref> element, the rendered version of this map would reflect two copies of "Sample branch", each reflecting the conditions that are specified in the corresponding DITAVAL document. One copy is created using the conditions in conditions.ditaval, while the other copy uses the conditions from otherConditions.ditaval. Map authors can use specific elements from the DITAVAL reference domain to indicate how resources should be renamed, or processors may recover from naming collisions by using an alternate naming scheme. See Limitations (see page 367) below for more information.

If DITAVAL conditions are specified at multiple levels within a single branch, the conditions that are specified at the higher level take precedence. In the following branch, assume alternate rules are specified for the condition audience="novice". In that case, the condition specified in highLevel.ditaval takes precedence and so applies to the entire branch.

```xml
<topicref href="ancestor.dita">
  <ditavalref href="highLevel.ditaval"/>
</topicref>
```

```xml
<topicref href="descendent.dita">
  <ditavalref href="lowLevel.ditaval"/>
  <!-- Other topicrefs -->
</topicref>
```
If a <ditavalref> element is used that does not specify the @href attribute, the element is still processed but no additional filtering is applied. This can be used to create an unfiltered copy of a map branch alongside other filtered copies; other aspects of the <ditavalref> (such as any specified key scope or modified resource name) will still be applied to the branch.

Limitations

The following limitations apply when using the <ditavalref> element; these limitations cannot be enforced through the DTD or Schema.

Comment by Kristen Eberlein, 5 February 2014
Do we need to add a mention of RelaxNG?

When the use of the <ditavalref> element results in multiple copies of a branch, resource names within that branch can be controlled with sub-elements of the effective <ditavalref>. For situations where resource names are relevant, it is an error condition for multiple <ditavalref> elements to result in conflicting resource names for different content. For example, the following map fragment would result in two distinct copies of the c.dita topic with the same file name:

```xml
<topicref href="c.dita">
  <ditavalref href="one.ditaval"/>
  <ditavalref href="two.ditaval"/>
</topicref>
```

Processors MAY recover by using an alternate naming scheme for the conflicting copies.

Inheritance

+ map/topicref ditavalref-d/ditavalref

Example

See Examples of branch filtering (see page 97) for several examples of the <ditavalref> element.

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and the attributes defined below.

@href

Provides a reference to a DITAVAL document. If the @href attribute is unspecified, this <ditavalref> will not result in any new filtering behavior, but other aspects of the element are still evaluated. See The href attribute (see page 663) for general information on the format and processing implications of the @href attribute.

@format

Format of the target document, which MUST be a DITAVAL document. The default value for this element is "ditaval". See The format attribute (see page 676) for more information.

@processing-role

The processing role defaults to "resource-only" for DITAVAL documents, which are only used for processing and do not contain content. There is no other valid value for this attribute on this element.

4.5.4.2 <ditavalmeta>

The <ditavalmeta> element defines metadata that is associated with a DITAVAL document used for one branch of a map.

Use the <ditavalmeta> to specify the prefixes and suffixes that processors use to construct effective resource names and key scope names within the map branch. The <ditavalmeta> element also can contain other information, such as author and navigation title, that might be useful for map architects but is not rendered in the output.
Inheritance
+ map/topicmeta ditavalref-d/ditavalmeta

Example
See ditavalref (see page 366).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650).

4.5.4.3 <dvrResourcePrefix>

The <dvrResourcePrefix> element specifies the prefix to use when constructing the effective file names of the resources that are referenced from within the map branch that is implied by the ancestor <ditavalref> element.

For map branches that are implied by <ditavalref> elements, the value of the <dvrResourcePrefix> element contributes to the effective file names of resources that are referenced within the branch. The effective resource file name starts with the value of the <dvrResourcePrefix> element. If @copy-to is specified on a topic reference where <dvrResourcePrefix> or <dvrResourceSuffix> based renaming is in effect, the prefixes or suffixes are applied to the resource name inside the @copy-to attribute.

Some resources are not eligible for renaming, such as those marked with scope="external". Rules for which resources are eligible for renaming, and what names are allowed as valid resource names, are the same as those for the @copy-to attribute defined in Topicref element attributes group (see page 663).

Inheritance
+ topic/data ditavalref-d/dvrResourcePrefix

Example
If the <dvrResourcePrefix> is specified in the following way:

```
<topicref href="branch-01.dita">
  <ditavalref href="condition-01.ditaval">
    <ditavalmeta>
      <dvrResourcePrefix>cond01-</dvrResourcePrefix>
    </ditavalmeta>
  </ditavalref>
<topicref href="topics/subtopic-01.dita"/>
```

Then the effective file name of the resource subtopic-01.dita is cond01-subtopic-01.dita, as though the @copy-to attribute had been specified with that value on the <topicref> element that references subtopic-01.dita.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and the attribute defined below.

@name

The name of the metadata item. For this element the default value is "dvrResourcePrefix". This attribute is defined with the XML Data Type CDATA.
4.5.4.4 <dvrResourceSuffix>

The <dvrResourceSuffix> element specifies the suffix to use when constructing the effective file names of resources that are referenced from within the map branch that is implied by the ancestor <ditavalref> element.

For map branches that are implied by <ditavalref> elements, the value of the <dvrResourceSuffix> element contributes to the effective file names of the resources that are referenced within the branch. The base part of the effective resource file name ends with the value of the <dvrResourceSuffix> element. The base part of the resource file name consists of the portion of the file name after any directory information, and before any period followed by the file extension. For example, in the original file name `task/install.dita`, the base portion of the file name is "install". If @copy-to is specified on a topic reference where <dvrResourcePrefix> or <dvrResourceSuffix> based renaming is in effect, the prefixes or suffixes are applied to the resource name inside the @copy-to attribute.

Some resources are not eligible for renaming, such as those marked with scope="external". Rules for which resources are eligible for renaming, and what names are allowed as valid resource names, are the same as those for the @copy-to attribute defined in Topicref element attributes group (see page 663), with one exception. Where @copy-to and <dvrResourcePrefix> may include path information, path information is not valid in <dvrResourceSuffix>.

**Inheritance**

+ topic/data ditavalref-d/dvrResourceSuffix

**Example**

If the <dvrResourceSuffix> is specified in the following way:

```xml
<topicref href="branch-01.dita">
  <ditavalref href="condition-01.ditaval">
    <ditavalmeta>
      <dvrResourceSuffix>-cond01</dvrResourceSuffix>
    </ditavalmeta>
    <ditavalref>
      <topicref href="topics/subtopic-01.dita"/>
    </ditavalref>
  </ditavalmeta>
</ditavalref>
```

Then the effective file name of resource `topics/subtopic-01.dita` is `topics/subtopic-01-cond01.dita`, as though the @copy-to attribute had been specified with that value on the <topicref> to `topics/subtopic-01.dita`. Similarly, the effective file name of resource `branch-01.dita` is `branch-01-cond01.dita`.

**Attributes**

The following attributes are available on this element: Universal attribute group (see page 650) and the attribute defined below.

@name

The name of the metadata item. For this element the default value is "dvrResourceSuffix". This attribute is defined with the XML Data Type CDATA.

4.5.4.5 <dvrKeyscopePrefix>

The <dvrKeyscopePrefix> element specifies the prefix to use when constructing the effective key scope names for the map branch that is implied by the ancestor <ditavalref> element.

For map branches that are implied by <ditavalref> elements, the value of the <dvrKeyscopePrefix> element contributes to the effective key scope names of the branch. The effective key scope names start with the value of the <dvrKeyscopePrefix> element. Note that if the branch as authored does not specify a @keyscope value, specifying <dvrKeyscopePrefix> (without also specifying <dvrResourceSuffix>) results in the branch...
establishing a key scope whose name is the value of the `<dvrKeyscopePrefix>` element. The full key scope names also will reflect the value of a `<dvrKeyscopeSuffix>` element if one is specified, regardless of whether the branch as authored specifies a `@keyscope` value.

**Inheritance**

+ topic/data ditavalref-d/dvrKeyscopePrefix

**Example**

If the `<dvrKeyscopePrefix>` is specified in the following way:

```xml
<topicref keys="branch-01"
    href="branch-01.dita"
    keyscope="branch-01"
>
    <ditavalref href="condition-01.ditaval">
        <ditavalmeta>
            <dvrKeyscopePrefix>cond01</dvrKeyscopePrefix>
        </ditavalmeta>
    </ditavalref>
    <topicref href="topics/subtopic-01.dita"/>
</topicref>
```

Then the effective key scope name for the branch "branch-01" is "cond01-branch-01".

**Attributes**

The following attributes are available on this element: Universal attribute group (see page 650) and the attribute defined below.

`@name`

The name of the metadata item. For this element the default value is "dvrKeyscopePrefix". This attribute is defined with the XML Data Type CDATA.

4.5.4.6 `<dvrKeyscopeSuffix>`

The `<dvrKeyscopeSuffix>` element specifies the suffix to use when constructing the effective key scope names for the map branch that is implied by the ancestor `<ditavalref>` element.

For map branches that are implied by `<ditavalref>` elements, the value of the `<dvrKeyscopeSuffix>` element contributes to the effective key scope names of the branch. The effective key scope names end with the value of the `<dvrKeyscopeSuffix>` element. Note that if the branch as authored does not specify a `@keyscope` value, specifying `<dvrKeyscopeSuffix>` (without also specifying `<dvrKeyscopePrefix>`) results in the branch establishing a key scope whose name is the value of the `<dvrKeyscopeSuffix>` element. The full key scope names also will reflect the value of a `<dvrKeyscopePrefix>` element if one is specified, regardless of whether the branch as authored specifies a `@keyscope` value.

**Inheritance**

+ topic/data ditavalref-d/dvrKeyscopeSuffix

**Example**

If the `<dvrKeyscopeSuffix>` is specified in the following way:

```xml
<topicref keys="branch-01"
    href="branch-01.dita"
    keyscope="branch-01"
>
    <ditavalref href="condition-01.ditaval">
        <ditavalmeta>
            <dvrKeyscopeSuffix>-cond01</dvrKeyscopeSuffix>
        </ditavalmeta>
    </ditavalref>
</topicref>
```
Then the effective key scope name for the branch "branch-01" is "branch-01-cond01".

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and the attribute defined below.

@name

The name of the metadata item. For this element the default value is "dvrKeyscopeSuffix". This attribute is defined with the XML Data Type CDATA.

### 4.6 Classification elements

Classification elements support managing metadata. Those in the Subject Scheme map are used to define controlled values, and to bind them to DITA attributes as enumerations. Those declared in the classification domain are used in other maps to classify content according to the scheme.

#### 4.6.1 Subject scheme maps

A subject scheme map is used to define sets of controlled values for use in classifying content. Sets of controlled values can be bound to DITA attributes. This allows DITA users to share the controlled values for an information set without having to modify a DTD or XML schema. The list of available values can be modified quickly to adapt to new situations, without the need to manage updates to a document type. In addition, DITA users can define relationships between controlled values and extend a set of controlled values maintained by another team or organization. The list of defined values are not validated by basic XML parsers. Instead, the defined values SHOULD be validated by DITA processors.

The same core elements in a subject scheme map may be used both to define controlled values and to define hierarchical taxonomies:

- schemeref
- subjectdef

The following elements are used to bind taxonomies or controlled values to an attribute:

- enumerationdef
- elementdef
- attributedef
- defaultSubject

The remaining elements in the subject scheme map are used to make more precise statements about how values in a taxonomy relate to one another.

#### 4.6.1.1 <subjectScheme>

The `<subjectScheme>` element is a specialization of `<map>`; it defines a collection of controlled values rather than a collection of topics.

Default values in the scheme (as specified by the `<defaultSubject>` element) apply only if the XML grammar does not specify a value in another way. The precedence of the different methods of setting a value is:

1. An explicit value in the element instance
2. A default value in the XML grammar
3. Cascaded values within the document
4. Cascades from a higher level document to this document
5. A default controlled value for a scheme
6. Values in processing rules
Inheritance
- map/map subjectScheme/subjectScheme

Example

```dita
<subjectScheme>
  <!-- Pull in a scheme that defines unix OS values -->
  <schemeref href="unixOS.ditamap"/>

  <!-- Define new OS values that are merged with those in the unixOS scheme -->
  <subjectdef keys="os">
    <subjectdef keys="linux"/>
    <subjectdef keys="mswin"/>
    <subjectdef keys="zos"/>
  </subjectdef>

  <!-- Define application values -->
  <subjectdef keys="app" navtitle="Applications">
    <subjectdef keys="apacheserv" href="subject/apache.dita"/>
    <subjectdef keys="mysql" href="subject/sql.dita"/>
  </subjectdef>

  <!-- Define an enumeration of the platform attribute, equal to each value in the OS subject. This makes the following values valid for the platform attribute: linux, mswin, zos -->
  <enumerationdef>
    <attributedef name="platform"/>
    <subjectdef keyref="os"/>
  </enumerationdef>

  <!-- Define an enumeration of the otherprops attribute, equal to each value in the application subjects. This makes the following values valid for the otherprops attribute: apacheserv, mysql -->
  <enumerationdef>
    <attributedef name="otherprops"/>
    <subjectdef keyref="app"/>
  </enumerationdef>
</subjectScheme>
```

Example: how hierarchies affect filtering

In the following sample, there are subcategories within the general "os" category.

```dita
<subjectScheme>
  <subjectdef keys="os" navtitle="Operating system">
    <subjectdef keys="redhat" navtitle="RedHat Linux"/>
    <subjectdef keys="suse" navtitle="SuSE Linux"/>
  </subjectdef>

  <subjectdef keys="mswin" navtitle="Windows"/>
  <subjectdef keys="zos" navtitle="z/OS"/>

  <enumerationdef>
    <attributedef name="platform"/>
    <subjectdef keyref="os"/>
  </enumerationdef>
</subjectScheme>
```

The following values are valid on the `@platform` attribute: linux, redhat, suse, mswin, zos. If any other values are encountered, processors validating against the scheme SHOULD give a warning. As a result, the values could be used in this way:

```dita
<p platform="linux">You must set up a cron job to ...</p>
<p platform="redhat">To set up the cron job, ...</p>
```

Comment by Kristen Eberlein, 16 April 2014
The normative statement below needs to be placed in the main body of the topic -- or in the architectural spec.

Processors SHOULD be aware of hierarchies of attributes defined in subject scheme maps, and process them differently than they might if the attributes were not defined in a hierarchy. Using the values "linux" and
"redhat" from the scheme above, where the subject "linux" is a container for the subject "redhat", filtering and flagging operate as follows:

Table 14: How to evaluate subjects in a hierarchy

<table>
<thead>
<tr>
<th>Behavior of &quot;linux&quot;</th>
<th>Behavior of &quot;redhat&quot;</th>
<th>How to evaluate platform=&quot;redhat&quot;</th>
<th>How to evaluate platform=&quot;linux&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>set to &quot;exclude&quot;</td>
<td>set to &quot;exclude&quot;</td>
<td>Excluded</td>
<td>Excluded</td>
</tr>
<tr>
<td>set to &quot;include&quot; or &quot;flag&quot;</td>
<td></td>
<td>Excluded. This is an error condition, because if all linux content is excluded, redhat is also excluded. Applications may recover by generating an error message.</td>
<td>Excluded</td>
</tr>
<tr>
<td>unspecified</td>
<td></td>
<td>Excluded, because &quot;redhat&quot; is a special kind of &quot;linux&quot;, and linux is excluded</td>
<td>Excluded</td>
</tr>
<tr>
<td>set to &quot;include&quot;</td>
<td>set to &quot;exclude&quot;</td>
<td>Excluded, because all redhat content is excluded</td>
<td>Included</td>
</tr>
<tr>
<td>set to &quot;include&quot;</td>
<td></td>
<td>Included</td>
<td>Included</td>
</tr>
<tr>
<td>set to &quot;flag&quot;</td>
<td></td>
<td>Included and flagged with the &quot;redhat&quot; flag</td>
<td>Included</td>
</tr>
<tr>
<td>unspecified</td>
<td></td>
<td>Included, because all Linux content is included</td>
<td>Included</td>
</tr>
<tr>
<td>set to &quot;flag&quot;</td>
<td>set to &quot;exclude&quot;</td>
<td>Excluded, because all redhat content is excluded</td>
<td>Included and flagged with the &quot;linux&quot; flag</td>
</tr>
<tr>
<td>set to &quot;include&quot;</td>
<td></td>
<td>Included and flagged with the &quot;linux&quot; flag, because linux is flagged and redhat is a type of linux</td>
<td>Included and flagged with the &quot;linux&quot; flag</td>
</tr>
<tr>
<td>set to &quot;flag&quot;</td>
<td></td>
<td>Included and flagged with the &quot;redhat&quot; flag, because a flag is available that is specifically for redhat</td>
<td>Included and flagged with the &quot;linux&quot; flag</td>
</tr>
<tr>
<td>unspecified</td>
<td></td>
<td>Included and flagged with the &quot;linux&quot; flag, because linux is flagged and redhat is a type of linux</td>
<td>Included and flagged with the &quot;linux&quot; flag</td>
</tr>
<tr>
<td>unspecified</td>
<td>set to &quot;exclude&quot;</td>
<td>Excluded, because all redhat content is excluded</td>
<td>If the default for @platform values is &quot;include&quot;, this is included. If the default for @platform values is &quot;exclude&quot;, this is excluded.</td>
</tr>
</tbody>
</table>

If the default for @platform values is "include", this is included.

If the default for @platform values is "exclude", this is excluded.
<table>
<thead>
<tr>
<th>Behavior of &quot;linux&quot;</th>
<th>Behavior of &quot;redhat&quot;</th>
<th>How to evaluate platform=&quot;redhat&quot;</th>
<th>How to evaluate platform=&quot;linux&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>set to &quot;include&quot;</td>
<td>Included</td>
<td>Included, because all &quot;redhat&quot; content is included, and general Linux content also applies to RedHat</td>
<td></td>
</tr>
<tr>
<td>set to &quot;flag&quot;</td>
<td>Included and flagged with the &quot;redhat&quot; flag</td>
<td>Included, because all &quot;redhat&quot; content is included, and general Linux content also applies to RedHat</td>
<td></td>
</tr>
<tr>
<td>unspecified</td>
<td>If the default for @platform values is &quot;include&quot;, this is included. If the default for @platform values is &quot;exclude&quot;, this is excluded.</td>
<td>If the default for @platform values is &quot;include&quot;, this is included. If the default for @platform values is &quot;exclude&quot;, this is excluded.</td>
<td></td>
</tr>
</tbody>
</table>

Example: Sample subject scheme for the @deliveryTarget attribute

The ability to have a hierarchy of types used for filtering and flagging is especially useful for the @deliveryTarget attribute. A department producing EPUBs might need to distinguish between many variants -- EPUB, EPUB2, EPUB3, Kindle format, EPUBs optimized for specific readers, and more -- while a sister department producing more traditional, print-focused output might need to exclude content with an EPUB focus.

The following subject scheme map provides a possible set of values for the @deliveryTarget attribute.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE subjectScheme PUBLIC "-//OASIS//DTD DITA Subject Scheme Map//EN" "subjectScheme.dtd">
<subjectScheme>
  <subjectHead>
    <subjectHeadMeta>
      <navtitle>Example of values for the @deliveryTarget attribute</navtitle>
      <shortdesc>Provides a set of values for use with the @deliveryTarget conditional-processing attribute. This set of values is illustrative only; it is not normative. You can use any values with the @deliveryTarget attribute.</shortdesc>
    </subjectHeadMeta>
  </subjectHead>
  <subjectdef keys="deliveryTargetValues">
    <topicmeta><navtitle>Values for @deliveryTarget attributes</navtitle></topicmeta>
    <!-- A tree of related values -->
    <subjectdef keys="print">
      <topicmeta><navtitle>Print-primary deliverables</navtitle></topicmeta>
      <subjectdef keys="pdf">
        <topicmeta><navtitle>PDF</navtitle></topicmeta>
      </subjectdef>
      <subjectdef keys="css-print">
        <topicmeta><navtitle>CSS for print</navtitle></topicmeta>
      </subjectdef>
      <subjectdef keys="xsl-fo">
        <topicmeta><navtitle>XSL-FO</navtitle></topicmeta>
      </subjectdef>
      <subjectdef keys="afp">
        <topicmeta><navtitle>Advanced Function Printing</navtitle></topicmeta>
      </subjectdef>
    </subjectdef>
    <subjectdef keys="ms-word">
      <topicmeta><navtitle>Microsoft Word</navtitle></topicmeta>
    </subjectdef>
  </subjectdef>
</subjectScheme>
```
Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below), Attributes common to many map elements (see page 655) (with narrowed definitions of @processing-role and @toc, given below), Architectural attribute group (see page 654), outputclass (see page 0 ), and the attributes defined below. This element also uses @type, @scope, and @format from Link relationship attribute group (see page 661).

@id
Allows an ID to be specified for the map. Note that maps do not require IDs (unlike topics), and the map ID is not included in references to elements within a map. This attribute is defined with the XML Data Type ID.

@anchorref

Identifies a location within another map file where this map will be anchored at runtime. Resolution of the map is deferred until the final step in the delivery of any rendered content. For example, anchorref="map1.ditamap/a1" causes this map to be pulled into the location of the anchor point "a1" inside map1.ditamap when map1.ditamap is rendered for delivery. This attribute is defined with the XML Data Type CDATA.

@processing-role

For this element, the default value for @processing-role is "resource-only". Otherwise, the definition matches the one found in Attributes common to many map elements (see page 655).

@toc

For this element, the default value for @toc is "no". Otherwise, the definition matches the one found in Attributes common to many map elements (see page 655).

4.6.1.2 <schemeref>

The <schemeref> element provides a reference to another scheme. Typically, the referenced scheme defines a base set of controlled values that are extended by the current scheme. The values in the referenced scheme are merged with the current scheme; the result is equivalent to specifying all of the values in a single subject scheme map.

Inheritance

- map/topicref subjectScheme/schemeref

Example: Extending a category with more specific values

```
<subjectScheme>
  <subjectdef keys="os" navtitle="Operating system">
    <subjectdef keys="linux" navtitle="Linux"/>
    <subjectdef keys="redhat" navtitle="Red Hat Linux"/>
    <subjectdef keys="suse" navtitle="SuSE Linux"/>
  </subjectdef>
  <subjectdef keys="mswin" navtitle="Windows"/>
  <subjectdef keys="zos" navtitle="z/OS"/>
</subjectScheme>
```

```
<enumerationdef>
  <attributedef name="platform"/>
  <subjectdef keyref="os"/>
</enumerationdef>
```

Figure 81: The map baseOS.ditamap

Because a scheme establishes relationships between subjects rather than a contextual navigation structure, new relationships can be added to existing subjects. In particular, the referencing scheme can extend an enumeration by adding new relationships to existing subjects that belong to the enumeration. For instance, a scheme could extend the baseOS.ditamap scheme shown above by adding Macintosh OS as a child of the existing os subject and adding special versions of Windows under the existing mswin subject:

```
<subjectScheme>
  <schemeref href="baseOS.ditamap"/>
  <subjectdef keyref="os">
    <subjectdef keys="macos" navtitle="Macintosh"/>
    <subjectdef keyref="mswin">
      <subjectdef keys="winxp" navtitle="Windows XP"/>
      <subjectdef keys="winvis" navtitle="Windows Vista"/>
    </subjectdef>
  </subjectdef>
</subjectScheme>
```
The references to the subjects defined by the base scheme use the @keyref attribute to avoid duplicate definitions of the keys.

**Figure 82: Second map that references baseOS.ditamap**

The result of merging the extension scheme with the base scheme is exactly the same as the following single scheme.

```xml
<subjectScheme>
  <subjectdef keys="os" navtitle="Operating system">
    <subjectdef keys="linux" navtitle="Linux"/>
    <subjectdef keys="redhat" navtitle="RedHat Linux"/>
    <subjectdef keys="suse" navtitle="SuSE Linux"/>
  </subjectdef>
  <subjectdef keys="macos" navtitle="Macintosh"/>
  <subjectdef keys="mswin" navtitle="Windows">
    <subjectdef keys="winxp" navtitle="Windows XP"/>
    <subjectdef keys="win98" navtitle="Windows Vista"/>
  </subjectdef>
  <subjectdef keys="zos" navtitle="z/OS"/>
</subjectdef>
<enumerationdef>
  <attributedef name="platform"/>
  <subjectdef keyref="os"/>
</enumerationdef>
</subjectScheme>
```

Because the extended baseOS scheme bound the os subject to the @platform attribute, the extension scheme doesn't provide that binding. The controlled values added by the extension to the hierarchy for the os subject become part of the enumeration bound to the @platform attribute.

**Figure 83: Resulting scheme**

**Example: Extending a category upwards**

A category can also be extended upward. For instance, an extension scheme could create a Software category that includes operating systems as well as applications.

```xml
<subjectScheme>
  <schemeref href="baseOS.ditamap"/>
  <subjectdef keys="sw" navtitle="Software">
    <subjectdef keyref="os"/>
    <subjectdef keyref="app" navtitle="Applications">
      <subjectdef keyref="apacheserv" navtitle="Apache Web Server"/>
      <subjectdef keyref="mysql" navtitle="MySQL Database"/>
    </subjectdef>
  </subjectdef>
</subjectScheme>
```

If the extended baseOS scheme defines the binding of the os subject with the @platform attribute, the app subjects provided by the extension scheme aren't subordinate to the os subject and thus don't become part of that enumeration. To leave open the possibility of upward extension of an enumeration, the content provider should define the controlled values in one scheme and define the binding to the attribute separately in a extension scheme. That way, the content provider can substitute a binding to a different extension without rework.

An adopter would identify the extension scheme as the scheme governing controlled values in the DITA environment. Any base schemes referenced by the extension scheme are, from a logical view, part of the extension scheme.

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661) (with narrowed definitions of @href, @format, and @type, given below),
@navtitle and @query from Topicref element attributes group (see page 663), outputclass (see page 0 ), @keys, and @keyref.

@href
A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

@format
On this element the @format attribute sets a default value of "ditamap", because the purpose of the element is to reference a ditamap document. Otherwise, the attribute is the same as described in Link relationship attribute group (see page 661).

@type
Describes the target of a reference. For the <schemeref> element, this value defaults to "scheme", because the element is expected to point to another subject scheme.

4.6.1.3 <hasInstance>

The <hasInstance> element specifies that the contained subjects have an INSTANCE-OF relationship with the container subject. In an INSTANCE-OF hierarchy, the child subject is a specific entity or object and the parent subject is a type, kind, or class of entity or object.

Inheritance
- map/topicref subjectScheme/hasInstance

Example

This example specifies that New York City, Reykjavik, and Moscow are each specific instances of a city.

```xml
<subjectScheme>
  <hasInstance>
    <subjectdef keys="city" navtitle="City">
      <subjectdef keys="nyc" navtitle="New York City"/>
      <subjectdef keys="reykjavik" navtitle="Reykjavik"/>
      <subjectdef keys="moscow" navtitle="Moscow"/>
    </subjectdef>
  </hasInstance>
</subjectScheme>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), @processing-role from Attributes common to many map elements (see page 655), @navtitle from Topicref element attributes group (see page 663), outputclass (see page 0 ), @keys, and @keyref.

@href
A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.
4.6.1.4 <hasKind>

The <hasKind> element specifies that the contained hierarchy expresses KIND-OF relationships between subjects.

In a KIND-OF hierarchy, the child subject is a particular variety of the parent subject. A KIND-OF hierarchy is sometimes known as an IS-A, generic, or subsumption hierarchy.

**Inheritance**

- map/topicref subjectScheme/hasKind

**Example**

This example specifies that cars, trucks, and motorcycles are kinds of vehicles. In addition, compact, sedan, and station wagon are each a kind of car, while pickup and van are each a type of truck.

```xml
<subjectScheme>
  <hasKind>
    <subjectdef keys="vehicle" navtitle="Vehicle"/>
    <subjectdef keys="car" navtitle="Passenger car"/>
    <subjectdef keys="compact" navtitle="Compact car"/>
    <subjectdef keys="sedan" navtitle="Sedan"/>
    <subjectdef keys="stationWagon" navtitle="Station wagon"/>
  </subjectdef>
  <subjectdef keys="truck" navtitle="Truck"/>
    <subjectdef keys="pickup" navtitle="Pickup truck"/>
    <subjectdef keys="van" navtitle="Van"/>
  </subjectdef>
  <subjectdef keys="motorcycle" navtitle="Motorcycle"/>
  </subjectdef>
</subjectScheme>
```

**Attributes**

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), @processing-role from Attributes common to many map elements (see page 655), @navtitle from Topicref element attributes group (see page 663), outputclass (see page 0 ), @keys, and @keyref.

@href

A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

4.6.1.5 <hasNarrower>

For subjects within the <hasNarrower> element, the container subject is more general than each of the contained subjects. That is, this element makes the default hierarchical relationship explicit, although the way in which a relationship is narrower is not specified.

**Inheritance**

- map/topicref subjectScheme/hasNarrower
Example

This example specifies that Planting Roses is a narrower subject category than Horticulture, although it is part of the Horticulture subject area.

```
<subjectScheme>
  <hasNarrower>
    <subjectdef keys="horticulture" navtitle="Horticulture">
      <subjectdef keys="plantrose" navtitle="Planting Roses"/>
    </subjectdef>
  </hasNarrower>
</subjectScheme>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), @processing-role from Attributes common to many map elements (see page 655), @navtitle from Topicref element attributes group (see page 663), outputclass (see page 0), @keys, and @keyref.

@href

A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

4.6.1.6 <hasPart>

The <hasPart> element specifies that the contained hierarchy expresses PART-OF relationships between subjects.

Inheritance

- map/topicref subjectScheme/hasPart

Example

This example specifies that a tire and a horn are each a part of a car.

```
<subjectScheme>
  <hasPart>
    <subjectdef keys="car" navtitle="Car">
      <subjectdef keys="tire" navtitle="Tire"/>
      <subjectdef keys="horn" navtitle="Horn"/>
    </subjectdef>
  </hasPart>
</subjectScheme>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), @processing-role from Attributes common to many map elements (see page 655), @navtitle from Topicref element attributes group (see page 663), outputclass (see page 0), @keys, and @keyref.

@href

A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.
4.6.1.7 <hasRelated>

The <hasRelated> element identifies an associative relationship between the container subject and each of the contained subjects.

**Inheritance**
- map/topicref subjectScheme/hasRelated

**Example**

This example specifies that myProgram runs on Linux and Windows.

```xml
<subjectScheme>
  <subjectdef keys="myProgram" navtitle="My Program">
    <hasRelated keys="runsOn" navtitle="runs on">
      <subjectdef keys="linux" navtitle="Linux"/>
      <subjectdef keys="mswin" navtitle="Microsoft Windows"/>
    </hasRelated>
  </subjectdef>
</subjectScheme>
```

**Attributes**

Comment by robander
Matches other has* elements, EXCEPT that it adds @collection-type, which sets a default of "choice" and doesn't allow sequence. Sent email to TC to discuss this.

The following attributes are available on this element: **Universal attribute group** (see page 650), **Link relationship attribute group** (see page 661) (with a narrowed definition of href, given below), processing-role from **Attributes common to many map elements** (see page 655), navtitle from **Topicref element attributes group** (see page 663), **outputclass** (see page 0), @keys, and @keyref. This element also has a narrowed version of @collection-type from **Attributes common to many map elements** (see page 655), given below.

@href
A pointer to the resource represented by the <topicref>. See **The href attribute** (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

@collection-type
Collection types describe how links relate to each other. This attribute matches the use of @collection-type elsewhere in maps, except that the default on this element is "choice", and the value "sequence" is not available. Allowable values are:

- choice
  Indicates that one of the children should be selected. This is the default value.

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

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

-dita-use-conref-target
  See **Using the -dita-use-conref-target value** (see page 666) for more information.
4.6.1.8 <enumerationdef>

The <enumerationdef> element identifies one attribute and one or more categories that contain the controlled values for the enumeration. The @type attribute has a default value of keys.

When the <enumerationdef> element contains an <elementdef> element, the defined enumeration applies to that attribute only on the element specified by <elementdef>. The enumeration does not apply to the same attribute on other elements. For example, when the element contains both <elementdef name="lomDifficulty"/> and <attributedef name="value"/> this means that the only the @value attribute on the <lomDifficulty> element is limited to the specified enumeration. The @value attribute on other elements is not affected.

When the <enumerationdef> element does not contain an <elementdef> element, the attribute specified is limited to the enumeration on all elements. For example, when <enumerationdef> contains <attributedef name="value"/> but does not contain <elementdef>, the @value attribute is limited to the specified enumeration for all elements.

An enumeration can specify an empty category without children. In this case, no value is valid for the attribute.

**Note:** Whether an attribute takes a single value or multiple values from the enumeration is part of the structural definition of the element controlled by the DTD or XML Schema. That is, an attribute which is defined as CDATA can take multiple values, while an attribute defined as NMTOKEN can take only one.

**Inheritance**

- map/topicref subjectScheme/enumerationdef

**Example**

In this example, enumerations are specified for the @platform and @otherprops attributes. Note that the enumeration identifies a category of values; the values within the category are valid, while the category itself is not a valid value. For example, in the code sample here, the @platform attribute is associated with the enumeration for the category "os"; all values within the "os" category are thus valid on the @platform attribute, while the value "os" itself is not.

```
<subjectScheme>
 <!-- Pull in a scheme that defines unix OS values -->
  <schemeref href="unixOS.ditamap"/>
 <!-- Define new OS values that are merged with those in the unixOS scheme -->
  <subjectdef keys="os">
    <subjectdef keys="linux"/>
    <subjectdef keys="mswin"/>
    <subjectdef keys="zos"/>
  </subjectdef>
 <!-- Define application values -->
  <subjectdef keys="app" navtitle="Applications">
    <subjectdef keys="apacheserv" href="subject/apache.dita"/>
    <subjectdef keys="mysql" href="subject/sql.dita"/>
  </subjectdef>
 <!-- Define an enumeration of the platform attribute, equal to each value in the OS subject. This makes the following values valid for the platform attribute: linux, mswin, zos -->
  <enumerationdef>
    <attributedef name="platform"/>
    <subjectdef keyref="os"/>
  </enumerationdef>
 <!-- Define an enumeration of the otherprops attribute, equal to each value in the application subjects. This makes the following values valid for the otherprops attribute: apacheserv, mysql -->
  <enumerationdef>
    <attributedef name="otherprops"/>
    <subjectdef keyref="app"/>
  </enumerationdef>
</subjectScheme>
```
Attributes

The following attributes are available on this element: ID attribute group (see page 651), @status and @base from Metadata attribute group (see page 651), outputclass (see page 0), Debug attribute group (see page 654), and class (Not for use by authors) (see page 0).

4.6.1.9 <elementdef>

The <elementdef> element identifies an element on which an attribute is enumerated. When the <elementdef> is left out of an <enumerationdef> element, the enumeration is bound to the attribute in all elements.

Inheritance

- topic/data subjectScheme/elementdef

Example

In this example, the <lomDifficulty> element has been specialized from the <data> element. The @value attribute on the <lomDifficulty> element (but not the @value attribute on other elements) is bound to a specific set of values. This means that processors should limit that attribute on that element to the values veryEasy, easy, medium, difficult, or veryDifficult.

```
<subjectScheme>
  <subjectdef keys="difficulty">
    <subjectdef keys="veryEasy"/>
    <subjectdef keys="easy"/>
    <subjectdef keys="medium"/>
    <subjectdef keys="difficult"/>
    <subjectdef keys="veryDifficult"/>
  </subjectdef>
  <!--[-- ... -->]
  <enumerationdef>
    <elementdef name="lomDifficulty"/>
    <attributedef name="value"/>
    <subjectdef keyref="difficulty"/>
  </enumerationdef>
</subjectScheme>
```

Attributes

The following attributes are available on this element: ID attribute group (see page 651), @status and @base from Metadata attribute group (see page 651), outputclass (see page 0), Debug attribute group (see page 654), class (Not for use by authors) (see page 0), and the attributes defined below.

@name (REQUIRED)

Defines the element for which an attribute enumeration is defined. This attribute is defined with the XML Data Type CDATA.

@translate

Indicates whether the content of the element should be translated or not. For this element the default value is "no". Setting to "yes" will override the default. The DITA architectural specification contains a list of each OASIS DITA element and its common processing default for the translate value; because this element uses an actual default, it will always be treated as translate="no" unless overridden as described. Available values are:

no

The content of this element is not translateable.

yes
4.6.1.10 <attributedef>

The `<attributedef>` element specifies the attribute to which a set of controlled values from a subject scheme map are to be applied.

**Inheritance**

- topic/data subjectScheme/attributedef

**Example**

In this example, enumerations are specified for the `@platform` and `@otherprops` attributes. Note that the enumeration identifies a category of values; the values within the category are valid, while the category itself is not a valid value. For example, in the code sample here, the `@platform` attribute is associated with the enumeration for the category "os"; all values within the "os" category are thus valid on the `@platform` attribute, while the value "os" itself is not.

```xml
<subjectScheme>
  <!-- Pull in a scheme that defines unix OS values -->
  <schemeref href="unixOS.ditamap"/>
  <!-- Define new OS values that are merged with those in the unixOS scheme -->
  <subjectdef keys="os">
    <subjectdef keys="linux"/>
    <subjectdef keys="mswin"/>
    <subjectdef keys="zos"/>
  </subjectdef>
  <!-- Define application values -->
  <subjectdef keys="app" navtitle="Applications">
    <subjectdef keys="apacheserv" href="subject/apache.dita"/>
    <subjectdef keys="mysql" href="subject/sql.dita"/>
  </subjectdef>
  <!-- Define an enumeration of the platform attribute, equal to each value in the OS subject. This makes the following values valid for the platform attribute: linux, mswin, zos -->
  <enumerationdef>
    <attributedef name="platform"/>
    <subjectdef keyref="os"/>
  </enumerationdef>
  <!-- Define an enumeration of the otherprops attribute, equal to each value in the application subjects. This makes the following values valid for the otherprops attribute: apacheserv, mysql -->
  <enumerationdef>
    <attributedef name="otherprops"/>
    <subjectdef keyref="app"/>
  </enumerationdef>
</subjectScheme>
```

**Attributes**

The following attributes are available on this element: **ID attribute group** (see page 651), `@status` and `@base` from **Metadata attribute group** (see page 651), **outputclass** (see page 0), **Debug attribute group** (see page 654), **class (Not for use by authors)** (see page 0), and the attributes defined below.

**@name (REQUIRED)**

Defines an attribute that will take a set of enumerated values. This attribute is defined with the XML Data Type CDATA.

**@translate**
Indicates whether the content of the element should be translated or not. For this element the default value is "no". Setting to "yes" will override the default. The DITA architectural specification contains a list of each OASIS DITA element and its common processing default for the translate value; because this element uses an actual default, it will always be treated as translate="no" unless overridden as described. Available values are:

- **no**
  - The content of this element is not translateable.
- **yes**
  - The content of this element is translateable.

### -dita-use-conref-target

See *Using the -dita-use-conref-target value* (see page 666) for more information.

#### 4.6.1.11 <defaultSubject>

The `<defaultSubject>` element is used within an attribute enumeration to set the default value for that attribute in cases where no value is specified on the attribute. The default subject must be one of the controlled values within the categories specified for the attribute.

**Inheritance**

- map/topicref subjectScheme/defaultSubject

**Example**

The following example declares that each of the four defined "os" values is valid within the @platform attribute; if no value is specified, the default is "linux".

```
<subjectScheme>
  <subjectdef keys="os">
    <subjectdef keys="linux"/>
    <subjectdef keys="mswin"/>
    <subjectdef keys="zos"/>
    <subjectdef keys="macos"/>
  </subjectdef>
  <enumerationdef>
    <attributedef name="platform"/>
    <defaultSubject keyref="linux"/>
    <subjectdef keyref="os"/>
  </enumerationdef>
</subjectScheme>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661) (with a narrowed definition of @href, given below), *Topicref element attributes group* (see page 663), *outputclass* (see page 0), @keys, and @keyref. This element also uses @processing-role, @locktitle, and @toc from *Attributes common to many map elements* (see page 655).

- **@href**
  - A pointer to the resource represented by the `<topicref>`. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

#### 4.6.1.12 <subjectHead>

The `<subjectHead>` element provides a heading for a group of subjects, for use if the scheme is displayed. For instance, a scheme may be displayed to let a user select subjects as part of faceted browsing. The
<subjectHead> element itself does not reference a file and cannot be referenced as a key, so it does not define any controlled values.

**Inheritance**

- map/topicref subjectScheme/subjectHead

**Example**

In this example the "Server setup" label doesn't classify content but, when selected, is equivalent to the union of its child subjects. That is, the heading covers content about planning for any application, installing for any application, any task for web servers, or any task for database servers.

```xml
<subjectScheme toc="yes" search="no">
<!-- ... -->
<subjectHead>
  <subjectHeadMeta>
    <navtitle>Server setup</navtitle>
  </subjectHeadMeta>
  <subjectdef href="planningTaskType.dita"/>
  <subjectdef href="installingTaskType.dita"/>
  <subjectdef href="webServerApp.dita"/>
  <subjectdef href="databaseApp.dita"/>
</subjectHead>
<!-- ... -->
</subjectScheme>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), @navtitle from *Topicref element attributes group* (see page 663), and *outputclass* (see page 0). This element also uses @processing-role, @toc, and narrowed definitions of @collection-type and @linking from *Attributes common to many map elements* (see page 655).

@collection-type

Collection types describe how links relate to each other. The processing default is "unordered", although no default is specified in the DTD or Schema. Allowable values for @collection-type on this element are:

 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.

 -dita-use-conref-target

 See *Using the -dita-use-conref-target value* (see page 666) for more information.

@linking

 Defines some specific linking characteristics of subject topics. "normal" is the only valid value, and is specified as the default in the DTD, XSD Schema, and RELAX NG implementations. When attribute values cascade, this causes a linking value of "normal" to cascade to the subjects.

### 4.6.1.13 <subjectHeadMeta>

The <subjectHeadMeta> element allows a navigation title and short description to be associated with a subject heading.

**Inheritance**

- map/topicmeta subjectScheme/subjectHeadMeta
Example

In this example the "Server setup" label doesn't classify content but, when selected, is equivalent to the union of its child subjects. That is, the heading covers content about planning for any application, installing for any application, any task for web servers, or any task for database servers.

```xml
<subjectScheme toc="yes" search="no">
  <!-- ... -->
  <subjectHead>
    <subjectHeadMeta>
      <navtitle>Server setup</navtitle>
    </subjectHeadMeta>
  </subjectHead>
  <subjectdef href="planningTaskType.dita"/>
  <subjectdef href="installingTaskType.dita"/>
  <subjectdef href="webServerApp.dita"/>
  <subjectdef href="databaseApp.dita"/>
  <!-- ... -->
</subjectScheme>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and the attribute defined below.

@lockmeta

Indicates whether the metadata information should be replaced by the metadata information in the referenced topic. Allowable values are:

yes

The information inside <topicmeta> should not be replaced with information from the topic.

no

The information inside <topicmeta> may be replaced with information from the topic.

@dita-use-conref-target

See Using the -dita-use-conref-target value (see page 666) for more information.

4.6.1.14 <subjectdef>

The <subjectdef> element defines a subject (also known as a controlled value) within a scheme. To make the subject easy to identify, a <subjectdef> may use a @keys attribute to assign a key to the subject. A subject with a key can be identified elsewhere with a @keyref. The <subjectdef> may use a <navtitle> element or @navtitle attribute to supply a label for the subject. The <subjectdef> may also refer to a topic that captures the consensus definition for the subject.

As with normal <topicref> processing, when the <subjectdef> element specifies a <navtitle> and refers to a topic, processors should use the actual topic title in place of the <navtitle>. When the <navtitle> is preferred as a subject label, the <subjectdef> element must have the @locktitle attribute set to "yes".

Inheritance

- map/topicref subjectScheme/subjectdef

Example

```xml
<subjectScheme>
  <!-- Pull in a scheme that defines unix OS values -->
  <schemeref href="unixOS.ditamap"/>
  <!-- Define new OS values that are merged with those in the unixOS scheme -->
  <subjectdef keys="zos"/>
    <subjectdef keys="linux"/>
    <subjectdef keys="mswin"/>
    <subjectdef keys="zos"/>
  </subjectdef>
</subjectScheme>
```
Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Topicref element attributes group (see page 663), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), outputclass (see page 0 ), @keys, and @keyref. This element also uses @processing-role, @toc, @collection-type, @linking, and @locktitle from Attributes common to many map elements (see page 655).

@href
A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

4.6.1.15 <relatedSubjects>

The <relatedSubjects> element establishes associative relationships between each child subject and every other child subject (unless the association is restricted by the @linking attribute of the subjects).

For filtering and flagging, processors need only inspect the subordinate hierarchies under category subjects that are bound to attributes. Filtering and flagging processors do not have to understand specific types of relationships. Explicit relationships are useful primarily for information viewers with advanced capabilities.

The content provider can identify the relationship by specifying a @keys attribute, label the relationship by specifying a <navtitle> element or @navtitle attribute, and provide a consensus definition of the relationship including by referencing a topic. If the relationship has an identifying key, the content provider can use the @keyref attribute to specify the same relationship for different subjects.

Inheritance
- map/topicref subjectScheme/relatedSubjects

Example
The following scheme establishes that the Linux, the Apache Web Server, and the MySQL Database are related:

```xml
<subjectScheme>
<!-- Define application values -->
<subjectdef keys="app" navtitle="Applications">
  <subjectdef keys="apacheweb" href="subject/apache.dita"/>
  <subjectdef keys="mysql" href="subject/sql.dita"/>
</subjectdef>

<!-- Define an enumeration of the platform attribute, equal to each value in the OS subject. This makes the following values valid for the platform attribute: linux, mswin, zos -->
<enumerationdef>
  <attributedef name="platform"/>
  <subjectdef keyref="os"/>
</enumerationdef>

<!-- Define an enumeration of the otherprops attribute, equal to each value in the application subjects. This makes the following values valid for the otherprops attribute: apacheserv, mysql -->
<enumerationdef>
  <attributedef name="otherprops"/>
  <subjectdef keyref="app"/>
</enumerationdef>
</subjectScheme>
```
Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), @navtitle and @query from Topicref element attributes group (see page 663), outputclass (see page 0), @keys, and @keyref. This element also uses @processing-role, @collection-type, and a narrowed definition of @linking (given below) from Attributes common to many map elements (see page 655).

@href

A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

@linking

On this element, the @linking attribute has a default value of "normal". Otherwise, the attribute is the same as defined in Attributes common to many map elements (see page 655).

4.6.1.16 <subjectRelTable>

The <subjectRelTable> element is a specialized relationship table which establishes relationships between the subjects in different columns of the same row. This element provides an efficient way to author non-hierarchical relationships between subjects. Tools (such as search tools) that use subject relationships to find related content may use these associative relationships in a similar way to the hierarchical relationships.

Inheritance

- map/reltable subjectScheme/subjectRelTable

Example

The subject relationship table in this example establishes relationships between operating systems and applications. Based on the <subjectRole> element, subjects in the first column are operating systems which are the environment for an application, while subjects in the second column are applications that run in that environment. For a user interested in content about the operating system, content about the applications may also be relevant.

```xml
<subjectScheme>
  <hasKind>
    <subjectdef keys="operatingSystem">
      <subjectdef keys="linuxOS"/>
      <subjectdef keys="windowsOS"/>
    </subjectdef>
    <subjectdef keys="application">
      <subjectdef keys="IDE">
        <subjectdef keys="eclipseIDE"/>
        <subjectdef keys="visualStudioIDE"/>
      </subjectdef>
      <subjectdef keys="webBrowser">
        <subjectdef keys="firefoxBrowser"/>
        <subjectdef keys="ieBrowser"/>
      </subjectdef>
    </subjectdef>
  </hasKind>
  <!-- ... -->
</subjectScheme>
```
A table view of the `<subjectRelTable>` may look like this; each `<subjectRel>` represents a single row, and each `<subjectRole>` represents a cell.

**Table 15: `<subjectRelTable>` as a table**

<table>
<thead>
<tr>
<th>&lt;subjectdef keyref=&quot;operatingSystem&quot;&gt;</th>
<th>&lt;subjectdef keyref=&quot;application&quot;/&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;hasRelated keyref=&quot;environmentFor&quot;&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;subjectdef keyref=&quot;application&quot;/&gt;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&lt;subjectdef keyref=&quot;linuxOS&quot;/&gt;</th>
<th>&lt;subjectdef keyref=&quot;eclipseIDE&quot;/&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;subjectdef keyref=&quot;windowsOS&quot;/&gt;</td>
<td>&lt;subjectdef keyref=&quot;firefoxBrowser&quot;/&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&lt;subjectdef keyref=&quot;windowsOS&quot;/&gt;</th>
<th>&lt;subjectdef keyref=&quot;ieBrowser&quot;/&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;subjectdef keyref=&quot;visualStudioIDE&quot;/&gt;</td>
</tr>
</tbody>
</table>

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *Attributes common to many map elements* (see page 655) (with a narrowed definition of `@toc`, given below). This element also uses `@type`, `@scope`, and `@format` from *Link relationship attribute group* (see page 661).

- **@toc**
  
  Specifies whether a topic appears in the table of contents (TOC). If the value is not specified locally, but is specified on an ancestor, the value will cascade from the closest ancestor. On this element the default value for `@toc` is "no". See *Attributes common to many map elements* (see page 655) for a complete definition of `@toc`.

### 4.6.1.17 `<subjectRelHeader>`

The `<subjectRelHeader>` element specifies the roles played by subjects in associations.

You use the `<subjectRelHeader>` element to supply a header row for a subject relationship table when you want to identify the roles played by the subjects in each column. Each cell in the header row identifies a subject
topic that defines a role. When specializing the `<subjectRelTable>` element, you can accomplish the same purpose by specializing the cells within the rows to enforce the roles.

**Inheritance**

- `map/relrow subjectScheme/subjectRelHeader`

**Example**

The subject relationship table in this example establishes relationships between operating systems and applications. Based on the `<subjectRole>` element, subjects in the first column are operating systems which are the environment for an application, while subjects in the second column are applications that run in that environment. For a user interested in content about the operating system, content about the applications may also be relevant.

```xml
<subjectScheme>
  <hasKind>
    <subjectdef keys="operatingSystem">
      <subjectdef keys="linuxOS"/>
      <subjectdef keys="windowsOS"/>
    </subjectdef>
    <subjectdef keys="application">
      <subjectdef keys="IDE">
        <subjectdef keys="eclipseIDE"/>
        <subjectdef keys="visualStudioIDE"/>
      </subjectdef>
      <subjectdef keys="webBrowser">
        <subjectdef keys="firefoxBrowser"/>
        <subjectdef keys="ieBrowser"/>
      </subjectdef>
    </subjectdef>
  </hasKind>
  <!-- ... -->
  <subjectRelTable>
    <subjectRelHeader>
      <subjectRole>
        <subjectdef keyref="operatingSystem">
          <hasRelated keyref="environmentFor">
            <subjectdef keyref="application"/>
          </hasRelated>
        </subjectdef>
      </subjectRole>
      <subjectRole>
        <subjectdef keyref="application"/>
      </subjectRole>
    </subjectRelHeader>
    <subjectRel>
      <subjectRole>
        <subjectdef keyref="linuxOS"/>
        <subjectdef keyref="windowsOS"/>
      </subjectRole>
      <subjectRole>
        <subjectdef keyref="eclipseIDE"/>
        <subjectdef keyref="firefoxBrowser"/>
      </subjectRole>
    </subjectRel>
    <subjectRel>
      <subjectRole>
        <subjectdef keyref="windowsOS"/>
      </subjectRole>
      <subjectRole>
        <subjectdef keyref="ieBrowser"/>
        <subjectdef keyref="visualStudioIDE"/>
      </subjectRole>
    </subjectRel>
  </subjectRelTable>
</subjectScheme>
```

A table view of the `<subjectRelTable>` may look like this; each `<subjectRel>` represents a single row, and each `<subjectRole>` represents a cell.
Table 16: `<subjectRelTable>` as a table

<table>
<thead>
<tr>
<th>&lt;subjectdef keyref=&quot;operatingSystem&quot;&gt;</th>
<th>&lt;subjectdef keyref=&quot;application&quot;&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;subjectdef keyref=&quot;linuxOS&quot;/&gt;</td>
<td>&lt;subjectdef keyref=&quot;eclipseIDE&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;subjectdef keyref=&quot;windowsOS&quot;/&gt;</td>
<td>&lt;subjectdef keyref=&quot;firefoxBrowser&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;subjectdef keyref=&quot;windowsOS&quot;/&gt;</td>
<td>&lt;subjectdef keyref=&quot;ieBrowser&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;subjectdef keyref=&quot;visualStudioIDE&quot;/&gt;</td>
</tr>
</tbody>
</table>

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650).

4.6.18 `<subjectRel>`

The `<subjectRel>` element contains a set of subjects that are related in some manner. Each group of subjects is contained in a `<subjectRole>` element; the associations between different columns in the same row are evaluated in the same way as those in a `<relrow>` (from which `<subjectRel>` is specialized) but define relationships between the subjects instead of links between topic documents.

**Inheritance**

- map/relrow subjectScheme/subjectRel

**Example**

The subject relationship table in this example establishes relationships between operating systems and applications. Based on the `<subjectRole>` element, subjects in the first column are operating systems which are the environment for an application, while subjects in the second column are applications that run in that environment. For a user interested in content about the operating system, content about the applications may also be relevant.
A table view of the `<subjectRelTable>` may look like this; each `<subjectRel>` represents a single row, and each `<subjectRole>` represents a cell.

Table 17: `<subjectRelTable>` as a table

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux OS</td>
<td>Eclipse IDE</td>
</tr>
<tr>
<td>Windows OS</td>
<td>Firefox Browser</td>
</tr>
<tr>
<td>Windows OS</td>
<td>Internet Explorer</td>
</tr>
<tr>
<td>Windows OS</td>
<td>Visual Studio IDE</td>
</tr>
</tbody>
</table>

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650).

**4.6.1.19 `<subjectRole>`**

The `<subjectRole>` element, when used within a `<subjectRel>` element, contains a set of subjects that are related to other subjects in the same row of the current `<subjectRelTable>`. By default, no relationship is defined between multiple subjects in the same `<subjectRole>` element. When used within the `<subjectRelHeader>`, the `<subjectRole>` element defines the category of subject or relationship provided by that column.

**Inheritance**

- map/relcell subjectScheme/subjectRole

**Example**

The subject relationship table in this example establishes relationships between operating systems and applications. Based on the `<subjectRole>` element, subjects in the first column are operating systems which
are the environment for an application, while subjects in the second column are applications that run in that environment. For a user interested in content about the operating system, content about the applications may also be relevant.

A table view of the `<subjectRelTable>` may look like this; each `<subjectRel>` represents a single row, and each `<subjectRole>` represents a cell.

**Table 18: `<subjectRelTable>` as a table**
4.6.2 Classification domain elements

The classification domain elements are used to identify the subject matter of content that is referenced in a map. These subjects must be subjects defined in a subject scheme map. In particular, this allows an author to classify content in new subject categories that are not bound to existing metadata attributes.

4.6.2.1 <subjectref>

The <subjectref> element identifies a subject to classify content. The <subjectref> can identify the subject with a @keyref attribute (if the scheme has a <subjectdef> with a @keys attribute that assigns a key to the subject) or an @href attribute (if the scheme is not available and a topic exists that defines the subject).

Inheritance

+ map/topicref classify-d/subjectref

Example

In the following example, the map is classified as covering the Linux subject and the "Developing web applications" topic as covering the web and development subjects. These subjects (and their keys) are defined externally in a subject scheme map; in order to reference the subject directly without the subject scheme map, the @href attribute would be used in place of @keyref.

```dita
<map>
  <title>Working with Linux</title>
  <topicsubject keyref="linux"/>
  <!-- ... -->
  <topicref href="webapp.dita" navtitle="Developing web applications">
    <topicsubject>
      <subjectref keyref="web"/>
    </topicsubject>
  </topicref>
  <!-- ... -->
</map>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), @navtitle and @query from Topicref element attributes group (see page 663), outputclass (see page 0), @keyref, and @keys. This element also uses @collection-type, @linking, and narrowed definitions of @processing-role and @toc (given below), from Attributes common to many map elements (see page 655).
@href

A pointer to the resource represented by the &lt;topicref&gt;. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

@processing-role

For this element, the default value for @processing-role is "resource-only". Otherwise, the definition matches the one found in Attributes common to many map elements (see page 655).

@toc

Specifies whether a topic appears in the table of contents (TOC). If the value is not specified locally, but is specified on an ancestor, the value will cascade from the closest ancestor. On this element the default value for @toc is "no". See Attributes common to many map elements (see page 655) for a complete definition of @toc.

4.6.2.2 &lt;topicapply&gt;

The &lt;topicapply&gt; element identifies subjects that qualify the content for filtering or flagging but not retrieval. The &lt;topicapply&gt; element can identify a single subject. Additional subjects can be specified by nested &lt;subjectref&gt; elements.

Inheritance

+ map/topicref classify-d/topicapply

Example

The map content should be retrieved for Apache Tomcat and hidden as irrelevant for operating systems other than RedHat or SuSE.

```
<map>
  <title>Installing Apache Tomcat on RedHat or SuSE Linux</title>
  <topicsubject href="../controlledValues/tomcatServer.dita"/>
  <topicapply>
    <subjectref href="../controlledValues/redhatLinux.dita"/>
    <subjectref href="../controlledValues/suseLinux.dita"/>
  </topicapply>
  <!-- ... -->
</map>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), @navtitle and @query from Topicref element attributes group (see page 663), outputclass (see page 0 ), @keyref, and @keys. This element also uses @collection-type, @linking, and narrowed definitions of @processing-role and @toc (given below), from Attributes common to many map elements (see page 655).

@href

A pointer to the resource represented by the &lt;topicref&gt;. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

@processing-role

For this element, the default value for @processing-role is "resource-only". Otherwise, the definition matches the one found in Attributes common to many map elements (see page 655).
@toc
Specifies whether a topic appears in the table of contents (TOC). If the value is not specified locally, but is specified on an ancestor, the value will cascade from the closest ancestor. On this element the default value for @toc is "no". See Attributes common to many map elements (see page 655) for a complete definition of @toc.

4.6.2.3 <topicsubject>

The <topicsubject> element identifies the subjects covered by a topic or map.

In order to identify a primary subject, refer to the subject with the <topicsubject> itself. Subjects can be identified by @keys (if defined in the scheme) or, if the subject definition topic exists, by @href (as with ordinary topic references).

Additional secondary subjects can be specified by nested <subjectref> elements.

Inheritance
+ map/topicref classify-d/topicsubject

Example
In the following example, the map is classified as covering Linux as the primary subject; the topic "Developing web applications" also covers the secondary web and development subjects. These subjects (and their keys) are defined externally in a subject scheme map; in order to reference the subject directly without the subject scheme map, the @href attribute would be used in place of @keyref.

```<map>
<title>Working with Linux</title>
<topicsubject keyref="linux"/>
<!-- ... -->
<topicref href="webapp.dita" navtitle="Developing web applications">
  <topicsubject>
    <subjectref keyref="web"/>
    <subjectref keyref="development"/>
  </topicsubject>
<!-- ... -->
</topicref>
<!-- ... -->
</map>```

Attributes
The following attributes are available on this element:Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), @navtitle and @query from Topicref element attributes group (see page 663), outputclass (see page 0 ), @keyref, and @keys. This element also uses narrowed definitions of @processing-role and @toc (given below) from Attributes common to many map elements (see page 655).

@href
A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

@processing-role
For this element, the default value for @processing-role is "resource-only". Otherwise, the definition matches the one found in Attributes common to many map elements (see page 655).

@toc
Specifies whether a topic appears in the table of contents (TOC). If the value is not specified locally, but is specified on an ancestor, the value will cascade from the closest ancestor. On this element the default value
for @toc is "no". See Attributes common to many map elements (see page 655) for a complete definition of @toc.

4.6.2.4 <topicSubjectTable>

The <topicSubjectTable> element is a specialized relationship table which allows a map to use relationship tables to associate topics with subjects. Tools (such as search tools) may use these classifications to retrieve content that is relative to a specific subject or combination of subjects.

In a <topicSubjectTable>, the first column is reserved for references to content. Subsequent columns are reserved for subjects that classify the content, each column supplying the subjects for a different category as identified in the header. The table resembles a traditional relationship table in which the first column identifies the source and the other columns identify the targets, but the relationship reflects the subjects covered by the content rather than linking between documents.

Note: In a traditional <reltable>, topics in any given column establish relationships with topics in every other cell of the same row. In a <topicSubjectTable>, topics in the first column are related to all of the subjects in the row, but no relationship is implied between subjects in different columns of the same row. Instead, relationships are defined between subjects using a subject scheme map (which provides a <subjectRelTable> element for non-hierarchical relationships between subjects).

Inheritance

+ map/reltable classify-d/topicSubjectTable

Example

The topic subject table below classifies topics with goals for retrieval and with operating systems for filtering. The map makes use of definitions in a subject scheme map, defined separately.

```
<subjectScheme>
  <hasKind>
    <subjectdef href="goalType.dita" keys="goal"/>
    <subjectdef href="performanceGoal.dita" keys="performance"/>
    <subjectdef href="reliabilityGoal.dita" keys="reliability"/>
  </subjectdef>
  <subjectdef href="operatingSystem.dita" keys="os"/>
    <subjectdef href="linuxOS.dita" keys="linux"/>
    <subjectdef href="unixOS.dita" keys="unix"/>
    <subjectdef href="windowsOS.dita" keys="windows"/>
  </subjectdef>
</hasKind>
</subjectScheme>
```

Figure 84: Subject scheme map

The following <topicSubjectTable> classifies several topics according to subjects in the previous map. As with any <topicSubjectTable>, the first column is used to specify topics. In this specific example, the second column is used to specify a goal, based on the "goal" subject in the header. The third column is used to specify an operating system. Based on those definitions, the following classifications are made by this table:

- The topics "Configuring cron for efficient startup" and "Allocating raw storage" are each classified by the goal of "performance"; in addition, they are classified by the operating systems "linux" and "unix".
- The topics "Analyzing web logs for service issues" and "Detecting denial-of-service attacks" are each classified by the goal of "reliability"; in addition, they are classified by the operating systems "linux", "unix", and "windows".
- No relationship is defined between subjects in the table, meaning that this table does not define any relationship between the goal of "performance" and the operating systems "linux" or "unix".
A table view of this `<topicSubjectTable>` might look as follows. This is only one of many possible views; to aid in understanding the example, the content topics in the first column are displayed using only their titles, and related subjects are displayed using only their @keyref attribute value.

<table>
<thead>
<tr>
<th>task</th>
<th>goal</th>
<th>os</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring cron for efficient startup</td>
<td>performance</td>
<td>linux</td>
</tr>
<tr>
<td>Allocating raw storage</td>
<td></td>
<td>unix</td>
</tr>
<tr>
<td>Analyzing web logs for service issues</td>
<td>reliability</td>
<td></td>
</tr>
<tr>
<td>Detecting denial-of-service attacks</td>
<td></td>
<td>windows</td>
</tr>
</tbody>
</table>

*Figure 85: Topic subject table*

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *Attributes common to many map elements* (see page 655) (with a narrowed definition of @toc, given below). This element also uses @type, @scope, and @format from *Link relationship attribute group* (see page 661).

@toc

Specifies whether a topic appears in the table of contents (TOC). If the value is not specified locally, but is specified on an ancestor, the value will cascade from the closest ancestor. On this element the default value for @toc is "no". See *Attributes common to many map elements* (see page 655) for a complete definition of @toc.
4.6.2.5 <topicSubjectHeader>

The <topicSubjectHeader> element specifies constraints on the subjects used in classifications.

Use the <topicSubjectHeader> element to supply a header row for a topic classification table when you want to encourage classification with subjects from different categories (also known as a facet classification). Each cell in the header row identifies the subject for a different category. The subjects in the same column within the classification rows must appear in the category in the subject scheme. For instance, if the cell within the header row specifies the Operating System category, the subjects in the column must be kinds of operating systems.

**Inheritance**

+ map/relrow classify-d/topicSubjectHeader

**Example**

The topic subject table below classifies topics with goals for retrieval and with operating systems for filtering. The map makes use of definitions in a subject scheme map, defined separately.

```xml
<subjectScheme>
  <hasKind>
    <subjectdef href="goalType.dita" keys="goal"/>
    <subjectdef href="performanceGoal.dita" keys="performance"/>
    <subjectdef href="reliabilityGoal.dita" keys="reliability"/>
  </hasKind>
  <subjectdef href="operatingSystem.dita" keys="os"/>
    <subjectdef href="linuxOS.dita" keys="linux"/>
    <subjectdef href="unixOS.dita" keys="unix"/>
    <subjectdef href="windowsOS.dita" keys="windows"/>
  </subjectdef>
</subjectScheme>
```

**Figure 86: Subject scheme map**

The following <topicSubjectTable> classifies several topics according to subjects in the previous map. As with any <topicSubjectTable>, the first column is used to specify topics. In this specific example, the second column is used to specify a goal, based on the "goal" subject in the header. The third column is used to specify an operating system. Based on those definitions, the following classifications are made by this table:

- The topics "Configuring cron for efficient startup" and "Allocating raw storage" are each classified by the goal of "performance"; in addition, they are classified by the operating systems "linux" and "unix".
- The topics "Analyzing web logs for service issues" and "Detecting denial-of-service attacks" are each classified by the goal of "reliability"; in addition, they are classified by the operating systems "linux", "unix", and "windows".
- No relationship is defined between subjects in the table, meaning that this table does not define any relationship between the goal of "performance" and the operating systems "linux" or "unix".

```xml
<map>
  <!-- ... -->
  <topicSubjectTable>
    <topicSubjectHeader>
      <topicCell type="task"/>
      <subjectCell>
        <topicsubject keyref="goal"/>
      </subjectCell>
      <topicCell>
        <topicapply keyref="os"/>
      </topicCell>
    </topicSubjectHeader>
    <topicSubjectRow>
      <topicCell>
        <topicref href="webServerStart.dita" navtitle="Configuring cron for efficient startup"/>
      </topicCell>
      <topicCell>
        <topicref href="dbDisk.dita" navtitle="Allocating raw storage"/>
      </topicCell>
      <subjectCell>
        <topicsubject keyref="performance"/>
      </subjectCell>
    </topicSubjectRow>
    <!-- ... -->
  </topicSubjectTable>
</map>
```
A table view of this `<topicSubjectTable>` might look as follows. This is only one of many possible views; to aid in understanding the example, the content topics in the first column are displayed using only their titles, and related subjects are displayed using only their @keyref attribute value.

<table>
<thead>
<tr>
<th>task</th>
<th>goal</th>
<th>os</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring cron for efficient startup</td>
<td>performance</td>
<td>linux, unix</td>
</tr>
<tr>
<td>Allocating raw storage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analyzing web logs for service issues</td>
<td>reliability</td>
<td>linux, unix</td>
</tr>
<tr>
<td>Detecting denial-of-service attacks</td>
<td></td>
<td>windows</td>
</tr>
</tbody>
</table>

*Figure 87: Topic subject table*

### Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650).

#### 4.6.2.6 `<topicSubjectRow>`

The `<topicSubjectRow>` is a grouping element that contains one row of a subject table. It contains topic references in the first column, and relates those references to the subjects in each following column.

### Inheritance

+ map/relrow classify-d/topicSubjectRow

### Example

The topic subject table below classifies topics with goals for retrieval and with operating systems for filtering. The map makes use of definitions in a subject scheme map, defined separately.

```xml
<subjectScheme>
  <hasKind>
    <subjectdef href="goalType.dita" keys="goal"/>
    <subjectdef href="performanceGoal.dita" keys="performance"/>
    <subjectdef href="reliabilityGoal.dita" keys="reliability"/>
    <subjectdef href="operatingSystem.dita" keys="os"/>
  </hasKind>
</subjectScheme>
```
Figure 88: Subject scheme map

The following <topicSubjectTable> classifies several topics according to subjects in the previous map. As with any <topicSubjectTable>, the first column is used to specify topics. In this specific example, the second column is used to specify a goal, based on the "goal" subject in the header. The third column is used to specify an operating system. Based on those definitions, the following classifications are made by this table:

- The topics "Configuring cron for efficient startup" and "Allocating raw storage" are each classified by the goal of "performance"; in addition, they are classified by the operating systems "linux" and "unix".
- The topics "Analyzing web logs for service issues" and "Detecting denial-of-service attacks" are each classified by the goal of "reliability"; in addition, they are classified by the operating systems "linux", "unix", and "windows".
- No relationship is defined between subjects in the table, meaning that this table does not define any relationship between the goal of "performance" and the operating systems "linux" or "unix".

A table view of this <topicSubjectTable> might look as follows. This is only one of many possible views; to aid in understanding the example, the content topics in the first column are displayed using only their titles, and related subjects are displayed using only their @keyref attribute value.
<table>
<thead>
<tr>
<th>task</th>
<th>goal</th>
<th>os</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring cron for efficient startup</td>
<td>performance</td>
<td>linux</td>
</tr>
<tr>
<td>Allocating raw storage</td>
<td></td>
<td>unix</td>
</tr>
<tr>
<td>Analyzing web logs for service issues</td>
<td>reliability</td>
<td>linux</td>
</tr>
<tr>
<td>Detecting denial-of-service attacks</td>
<td></td>
<td>unix windows</td>
</tr>
</tbody>
</table>

**Figure 89: Topic subject table**

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650).

### 4.6.2.7 `<topicCell>`

The `<topicCell>` element contains topics that will be associated with subjects in each following column of the current row in the `<topicSubjectTable>`.

**Inheritance**
+ map/relrow classify-d/topicSubjectRow

**Example**

The topic subject table below classifies topics with goals for retrieval and with operating systems for filtering. The map makes use of definitions in a subject scheme map, defined separately.

```
<subjectScheme>
  <hasKind>
    <subjectdef href="goalType.dita" keys="goal"/>
    <subjectdef href="performanceGoal.dita" keys="performance"/>
    <subjectdef href="reliabilityGoal.dita" keys="reliability"/>
  </hasKind>
  <subjectdef href="operatingSystem.dita" keys="os"/>
    <subjectdef href="linuxOS.dita" keys="linux"/>
    <subjectdef href="unixOS.dita" keys="unix"/>
    <subjectdef href="windowsOS.dita" keys="windows"/>
  </subjectdef>
</hasKind>
</subjectScheme>
```

**Figure 90: Subject scheme map**

The following `<topicSubjectTable>` classifies several topics according to subjects in the previous map. As with any `<topicSubjectTable>`, the first column is used to specify topics. In this specific example, the second column is used to specify a goal, based on the "goal" subject in the header. The third column is used to specify an operating system. Based on those definitions, the following classifications are made by this table:

- The topics "Configuring cron for efficient startup" and "Allocating raw storage" are each classified by the goal of "performance"; in addition, they are classified by the operating systems "linux" and "unix".
- The topics "Analyzing web logs for service issues" and "Detecting denial-of-service attacks" are each classified by the goal of "reliability"; in addition, they are classified by the operating systems "linux", "unix", and "windows".
No relationship is defined between subjects in the table, meaning that this table does not define any relationship between the goal of "performance" and the operating systems "linux" or "unix".

A table view of this `<topicSubjectTable>` might look as follows. This is only one of many possible views; to aid in understanding the example, the content topics in the first column are displayed using only their titles, and related subjects are displayed using only their `@keyref` attribute value.

<table>
<thead>
<tr>
<th>task</th>
<th>goal</th>
<th>os</th>
</tr>
</thead>
</table>
| Configuring cron for efficient startup    | performance | linux  
| Allocating raw storage                    |          | unix      |
| Analyzing web logs for service issues     | reliability | linux  
| Detecting denial-of-service attacks       |          | unix  
|                                          |          | windows  |

*Figure 91: Topic subject table*

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *Attributes common to many map elements* (see page 655). This element also uses `@scope`, `@format`, and `@type` from *Link relationship attribute group* (see page 661).
4.6.2.8 <subjectCell>

The <subjectCell> element contains subjects that are associated with topics in the first column of the current row in the <topicSubjectTable>. The subjects themselves have no defined relationship across columns, other than the fact that they apply to the same content.

Inheritance
+ map/relicell classify-d/subjectCell

Example

The topic subject table below classifies topics with goals for retrieval and with operating systems for filtering. The map makes use of definitions in a subject scheme map, defined separately.

```xml
<subjectScheme>
  <hasKind>
    <subjectdef href="goalType.dita" keys="goal"/>
    <subjectdef href="performanceGoal.dita" keys="performance"/>
    <subjectdef href="reliabilityGoal.dita" keys="reliability"/>
  </hasKind>
  <subjectdef href="operatingSystem.dita" keys="os"/>
    <subjectdef href="linuxOS.dita" keys="linux"/>
    <subjectdef href="unixOS.dita" keys="unix"/>
    <subjectdef href="windowsOS.dita" keys="windows"/>
  </subjectdef>
</subjectScheme>
```

Figure 92: Subject scheme map

The following <topicSubjectTable> classifies several topics according to subjects in the previous map. As with any <topicSubjectTable>, the first column is used to specify topics. In this specific example, the second column is used to specify a goal, based on the "goal" subject in the header. The third column is used to specify an operating system. Based on those definitions, the following classifications are made by this table:

- The topics "Configuring cron for efficient startup" and "Allocating raw storage" are each classified by the goal of "performance"; in addition, they are classified by the operating systems "linux" and "unix".
- The topics "Analyzing web logs for service issues" and "Detecting denial-of-service attacks" are each classified by the goal of "reliability"; in addition, they are classified by the operating systems "linux", "unix", and "windows".
- No relationship is defined between subjects in the table, meaning that this table does not define any relationship between the goal of "performance" and the operating systems "linux" or "unix".

```xml
<map>
<!-- ... -->
<topicSubjectTable>
  <topicSubjectHeader>
    <topicCell type="task"/>
    <subjectCell>
      <topicsubject keyref="goal"/>
    </subjectCell>
    <subjectCell>
      <topicapply keyref="os"/>
    </subjectCell>
  </topicSubjectHeader>
  <topicSubjectRow>
    <topicCell>
      <topicref href="webServerStart.dita" navtitle="Configuring cron for efficient startup"/>
      <topicref href="dbDisk.dita" navtitle="Allocating raw storage"/>
    </topicCell>
    <subjectCell>
      <topicsubject keyref="performance"/>
    </subjectCell>
    <subjectCell>
      <topicapply keyref="linux"/>
      <topicapply keyref="unix"/>
    </subjectCell>
  </topicSubjectRow>
</topicSubjectTable>
</map>
```
A table view of this `<topicSubjectTable>` might look as follows. This is only one of many possible views; to aid in understanding the example, the content topics in the first column are displayed using only their titles, and related subjects are displayed using only their `@keyref` attribute value.

<table>
<thead>
<tr>
<th>task</th>
<th>goal</th>
<th>os</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring cron for efficient startup</td>
<td>performance</td>
<td>linux, unix</td>
</tr>
<tr>
<td>Allocating raw storage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analyzing web logs for service issues</td>
<td>reliability</td>
<td>linux, unix</td>
</tr>
<tr>
<td>Detecting denial-of-service attacks</td>
<td></td>
<td>windows</td>
</tr>
</tbody>
</table>

**Figure 93: Topic subject table**

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *Attributes common to many map elements* (see page 655). This element also uses `@scope`, `@format`, and `@type` from *Link relationship attribute group* (see page 661).

### 4.7 Specialization elements

Several DITA elements exist either for architectural reasons or for support of specialized markup yet to be designed. Although there is little need to use these elements unless you are directed to, some of them, such as `<state>`, can be used if your content makes use of these semantic distinctions. For example, a discussion of signals on a gate of an integrated logic circuit might use the `<state>` element to represent either on or off conditions of that gate.

#### 4.7.1 `<boolean>`

The `<boolean>` element was deprecated in DITA version 1.1. It was originally intended to express one of two opposite values, such as yes / no or on / off.

**Note**: This element is deprecated. It is functionally equivalent to `<state value="yes|no"/>`, which is recommended as its replacement in all cases.

**Inheritance**

- topic/boolean
Example

She said "<boolean state="yes"/>" when I asked her to marry me!

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and the attribute defined below.

@state (REQUIRED)

The state of the <boolean> element. Allowable values are "yes", "no", and "-dita-use-conref-target".

4.7.2 <data>

The <data> element represents a property within a DITA topic or map. While the <data> element can be used directly to capture properties, it is particularly useful as a basis for specialization. Default processing should treat the content as an unknown kind of metadata and ignore it for rendering, but custom processing may match the @name attribute or specialized element and use the element for automated manipulation or to format data associated with the body flow. For example, a specialized <data> element may be used to format properties as sidebars or other adornments or to harvest properties for automated processing.

The subject of the property is ordinarily the container of the <data> element. In the content model for the <prolog> and <metadata> elements, the property applies to the topic as a whole. In the <topicmeta> element, the property applies to the referenced topic. The <data-about> element may be used to identify the subject of the property with an explicit reference.

The @name attribute names the property for processes. A <title> subelement may provide a label for the property. The @datatype attribute may be used to identify the type for the value. The value of the property can be any of the following:

- A simple text value expressed with the @value attribute or textual content.
- A reference to either DITA content or a non-DITA resource expressed with the @href attribute.
- An image or other non-textual object.
- A brief unit of descriptive text that is not part of the body text flow.
- A complex structure composed of nested <data> elements.

**Caution:** Processors should ignore the content of the <data> element by default, so the <data> element should only be used for properties and not to embed text for formatting as part of the flow of the topic body. It might be tempting to specialize the <data> element for text that is part of the body flow, so as to escape the restrictions of the base content models. This abuse of the DITA architecture will cause problems. For example, if a particular kind of paragraph is specialized from <data> rather than from <p>, then when the content is exchanged with others that do not recognize the specialized element, their processors will skip the content.

The <data> element may be nested to create structures for complex properties. The @name attribute may be used to distinguish different semantics associated with different instances of the <data> element such as addresses, times, amounts, and so on. In many cases, however, it is preferable to specialize the <data> element for more precise semantics and for constraints on structures and values. For instance, a specialization can specify an enumeration for the @value attribute.

A <data> element containing properties of a topic as a whole should be located in the topic's <prolog> or <metadata> element, or in a <topicmeta> element related to a <topicref> that references the topic. The <data> element generally goes at the beginning of the element to which the properties in it refer. Where this is unwieldy, the <data> element can go in the <prolog>, with the <data-about> element identifying which specific element in the topic is the reference.
Inheritance
- topic/data

Example

Uses of the `<data>` element may include the following:

- Complex metadata properties such as bibliographic records corresponding to citations.
- Hybrid documents with data values as part of the content, such as word processor formats using form fields.
- Messages in which the payload includes human-readable content. Such applications can use the `<data>` element to define the addressing on the message envelope. For instance, a topic could model an email message by representing the address with specialized `<data>` elements in the `<prolog>` element and the content with the `<body>` element.
- Transactional documents in which the values are processed but also displayed with human-readable content. In particular, a library of building blocks for transaction documents can be implemented through a DITA domain as specialized `<data>` elements including those from the UN/CEFACT Core Components Technical Specification (http://www.unece.org/cefact/).

This structure identifies the library and version demonstrated by a code sample. The `@name` attribute is used to identify both the grouping `<data>` element and the nested `<data>` elements that provide specific properties. These properties will not appear in the output unless a processor is customized to recognize these `@name` attribute values.

```
<codeblock>
  <data name="exampleOf">
    <data name="library" href="ajaxLibrary.js"/>
    <data name="version" value="2006-6-19"/>
  </data>
  <!-- ... -->
</codeblock>
```

**Figure 94: Using the @name attribute on unspecialized `<data>` elements**

The following example specifies the delimited source code for a code fragment so an automated process can refresh the code fragment. The `<fragmentSource>`, `<sourceFile>`, `<startDelimiter>`, and `<endDelimiter>` elements are specialized from `<data>` but the `<codeFragment>` is specialized from `<codeblock>`. These properties wouldn't appear in the formatted output (except perhaps for debugging problems in the refresh):

```
<example>
  <title>An important coding technique</title>
  <codeFragment>
    <fragmentSource>
      <sourceFile value="helloWorld.java"/>
      <startDelimiter value="FRAGMENT_START_1"/>
      <endDelimiter value="FRAGMENT_END_1"/>
    </fragmentSource>
    <!-- ... -->
  </codeFragment>
</example>
```

**Figure 95: Specializing `<data>` to annotate a code sample**

The following example identifies a real estate property as part of a house description. The `<realEstateProperty>` element and its child elements are specialized from `<data>`. The `<houseDescription>` element is specialized from `<section>`. A specialized process can format the values as part of a brochure if they meet criteria for inclusion.

```
<houseDescription>
  <title>A great home for sale</title>
  <realEstateProperty>
    <realEstateBlock value="B7"/>
  </realEstateProperty>
</houseDescription>
```
Attributes

The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), @keyref, and outputclass (see page 0).

Comment by robander

DITA 1.2 didn't include @keyref but it was part of the DTD, and clearly an intended attribute. Would be part of 1.3 regardless, given move to have everything with @href include @keyref.

4.7.3 <data-about>

The <data-about> element identifies the subject of a property when the subject isn’t associated with the context in which the property is specified. The property itself is expressed by the <data> element. The <data-about> element handles exception cases where a property must be expressed somewhere other than inside the actual subject of the property. The <data-about> element is particularly useful as a basis for specialization in combination with the <data> element.

Important: Do not use the <data-about> element to identify the object of a property. The @href attribute of the <data> element serves that purpose.

Inheritance

- topic/data-about

Example

The full properties of a cited book can be maintained conveniently in the <prolog>:

```
<topic id="questions">
  <!-- ... -->
  <title>Questions and answers</title>
  <prolog>
    <data-about href="urn:isbn:0156983508" scope="external">
      <data name="title">The World Doesn't End</data>
      <data name="author">
        <data name="firstname">Charles</data>
        <data name="lastname">Simic</data>
      </data>
      <data name="published" datatype="year">1989</data>
    </data-about>
    <!-- ... -->
  </prolog>
  <!-- ... -->
</body>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661), and outputclass (see page 0).
4.7.4 <foreign>

The <foreign> element allows the introduction of non-DITA content, for example, MathML, SVG, or Rich Text Format (RTF). The <foreign> element or a specialization may contain more than one type of non-DITA content or a mix of DITA and non-DITA content. Specialization of the <foreign> element generally is implemented as a domain, but architects looking for more control over the content may implement foreign vocabularies as structural specializations.

Processors should attempt to display <foreign> content unless otherwise instructed. If the processor cannot render the content, it MAY issue a warning.

The enabler of the foreign vocabulary must provide the processing and override the base processing for <foreign>.

- If <foreign> contains more than one alternative content element, they should all be processed. In the case of <desc> they should be concatenated in a similar way to <section>, but with no title (analogous to <div> in HTML).
- If alternate content is desired, specialize the <desc> element to contain it. This specialization of <desc> should be used within the element specialized from <foreign>. Such alternate content must of course be valid wherever the <foreign> specialization is valid.
- If no <desc>, <object>, or <image> element is found within an instance of the <foreign> element, the base processing may emit a warning about the absence of processable content.
- The base processing for <object> may emit the content of <foreign> as a file at the location specified by the @data attribute of the <object> element. The <object> element should have a data attribute or a <foreign> sub-element but not both. In the event that an <object> element contains both a data attribute and an <foreign> sub-element the processing system should ignore one of them.

Inheritance

- topic/foreign

SVG Example within a <p> element

```xml
<p>... as in the formula
<svg>
  <svg:svg width="100%" height="100%" version="1.1"
    xmlns="http://www.w3.org/2000/svg">
    <ellipse cx="300" cy="150" rx="200" ry="80"
      style="fill:rgb(200,100,50);
      stroke:rgb(0,0,100);stroke-width:2"/>
  </svg:svg>
</svg>.
</p>
```

MathML Example within an <object> element

```xml
<p>... as in the formula
<object>
  <desc>4 + x</desc>
  <mathML>
    <m:mrow>
      <m:mrow>
        <m:mo>+</m:mo>
        <m:mi>x</m:mi>
      </m:mrow>
    </m:mrow>
  </mathML>
</object>
</p>
```
Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.7.5 <index-base>

The <index-base> element allows indexing extensions to be added by specializing this element. The <index-base> element can only exist as a child of an <indexterm> element. This characteristic makes it the appropriate element to specialize to add indexing extensions. For example, the <index-see>, <index-see-also>, and <index-sort-as> elements only make sense as children of <indexterm> and so are specializations of <index-base>. Those elements are all part of the indexing domain.

On its own, <index-base> has no meaning. Processors should ignore this element and its content if encountered in its unspecialized form.

Inheritance
- topic/index-base

Example

The <index-see-also> element is specialized from <index-base>; see index-see-also (see page 341) for an example of how <index-base> may be used with specialization.

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and @keyref.

4.7.6 <itemgroup>

The <itemgroup> element can be used to sub-divide or organize elements that occur inside a list item, definition, or parameter definition.

The <itemgroup> element is particularly useful as a basis for specialization, where it can be used to group content within specialized list items or definitions. For example, in the OASIS task specialization, many elements within the <step> element are specialized from <itemgroup>.

Inheritance
- topic/itemgroup

Example

Second point of a list.

Example

<itemgroup>related discourse</itemgroup>

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.7.7 <no-topic-nesting>

The <no-topic-nesting> element is a placeholder in the DITA architecture. It is not actually used by the default DITA document types; it is for use only when creating a validly customized document type where the information
designer wants to eliminate the ability to nest topics. Not intended for use by authors, and has no associated output processing.

_Inheritance_
- topic/no-topic-nesting

_Example_

_This element is not intended to be used in source files._

**Attributes**
The following attributes are available on this element: _Debug attribute group_ (see page 654) and _class (Not for use by authors)_ (see page 0).

### 4.7.8 `<state>`

The `<state>` element specifies a name/value pair whenever it is necessary to represent a named state that has a variable value. The element is primarily intended for use in specializations to represent specific states (like logic circuit states, chemical reaction states, airplane instrumentation states, and so forth).

_Inheritance_
- topic/state

_Example_

```
<step><cmd>Verify the presence of an "on" or high condition at the input gate (ie, <state name="inflag" value="high"/>)</cmd></step>
```

**Attributes**
The following attributes are available on this element: _Universal attribute group_ (see page 650), _outputclass_ (see page 0), and the attributes defined below.

@name (REQUIRED)
The name of the property whose state is being described. This attribute is defined with the XML Data Type CDATA.

@value (REQUIRED)
The state of the property identified by the @name attribute. This attribute is defined with the XML Data Type CDATA.

### 4.7.9 `<unknown>`

The `<unknown>` element is an open extension that allows information architects to incorporate xml fragments that do not necessarily fit into an existing DITA use case. Processors should ignore this element unless otherwise instructed.

_Inheritance_
- topic/unknown
Example

This example features a specialized `<unknown>` element that includes other non-DITA content. If this specialization is imported to a DTD or schema, the DTD or schema will need to handle declaring the new elements or any namespaces.

```xml
<body>
  <my-unknown class="+ topic/unknown mything/my-unknown ">
    <thing value="4"/>
    <otherthing value="16"/>
  </my-unknown>
</body>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and `outputclass` (see page 0).

4.8 Legacy conversion elements

Conversion elements exist primarily to aid in the conversion of content to DITA.

4.8.1 `<required-cleanup>`

A `<required-cleanup>` element is used as a placeholder for migrated elements that cannot be appropriately tagged without manual intervention. As the element name implies, the intent for authors is to clean up the contained material and eventually remove the `<required-cleanup>` element. Authors should not insert this element into documents.

Processing notes:

- Processors must strip this element from output by default. The content of `<required-cleanup>` is not considered to be verified data.
- Processor options may be provided to allow a draft view of migrated content in context.

Inheritance

- topic/required-cleanup

Example

Presuming an original HTML document had contained some content within a `<center>` tag (for which there is no clear migrational equivalent in DITA), the following might be the result that is valid within an XML editor, but which requires an author to decide how to better tag or revise this original content:

```xml
<section>
  <title>Some section title</title>
  <required-cleanup remap="center">Some original content migrated from a &lt;center&gt; tag.</required-cleanup>
</section>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a modified definition of `@translate`, given below), `outputclass` (see page 0), and the attributes defined below.

`@remap`

Indicates the element that the contents of the `<required-cleanup>` element were mapped from (provides an idea about what the new intent should be). This attribute is defined with the XML Data Type CDATA.

`@translate`
Indicates whether the content of the element should be translated or not. The default value for this element is "no"; setting to "yes" will override the default. The -dita-use-conref-target value is also available. The DITA architectural specification contains a list of each OASIS DITA element and its common processing default for the translate value; because this element uses an actual default, it will always be treated as translate="no" unless overridden as described.

4.9 DITAVAL elements

A conditional processing profile (DITAVAL file) is used to identify which values are to be used for conditional processing during a particular output, build, or some other purpose. The profile should have an extension of .ditaval.

The DITAVAL format has several elements: <val>, the root element, can contain a <style-conflict> element followed by <prop> or <revprop> elements; the <prop> and <revprop> elements can contain <startflag> and <endflag> elements; and the <startflag> and <endflag> elements can contain <alt-text> elements.

Notes on ditaval messages

Conditional processing code should provide a report of any attribute values encountered in content that do not have an explicit action associated with them.

Note on ditaval flagging of images

If an image in DITA content becomes flagged using a background color, the color should be represented as a thick border. If a foreground color is expressed, it should be represented as a thin border.

Related Links

3.2.3.2.1 Conditional processing attributes (see page 44)
   The metadata attributes specify properties of the content that can be used to determine how the content should be processed.
   Specialized metadata attributes can be defined to enable specific business-processing needs, such as semantic processing and data mining.

3.3.7 Conditional processing (profiling) (see page 88)
   Conditional processing, also known as profiling, is the filtering or flagging of information based on processing-time criteria.

4.9.1 <val>

The <val> element is the root element of a DITAVAL file.

For information about processing DITAVAL files, including how to filter or flag elements with multiple property attributes or multiple properties within a single attribute, see Conditional processing (profiling) (see page 88).

Example

```
<val>
  <style-conflict background-conflict-color="red"/>
  <prop action="include" att="audience" val="everybody"/>
  <prop action="flag" att="product" val="YourProd" backcolor="purple"/>
  <prop action="flag" att="product" backcolor="blue" color="yellow" style="underline" val="MyProd">
    <startflag imageref="startflag.jpg">
      <alt-text>This is the start of my product info</alt-text>
    </startflag>
    <endflag imageref="endflag.jpg">
      <alt-text>This is the end of my product info</alt-text>
    </endflag>
  </prop>
  <revprop action="flag" val="1.2"/>
</val>
```

This sample DITAVAL file performs the following actions:

* Elements with audience="everybody" are included without change.
* Elements with product="YourProd" get a background color of purple.
• Elements with `product="MyProd"` get the following actions:
  • The image `startflag.jpg` is placed at the start of the element.
  • The image `endflag.jpg` is placed at the end of the element.
  • The element gets a background color of blue.
  • The text in the element appears in yellow; the text is underlined.
• Elements marked with are flagged with the default revision flags, which are implementation dependent.
• When there are conflicts, for example, if an element is marked with `product="MyProd YourProd"`, it will be flagged with a background color of red.

**Figure 97: Sample DITAVAL file**

```xml
<val>
  <prop action="exclude"/>
  <prop action="include" att="audience" val="everybody"/>
  <prop action="include" att="audience" val="novice"/>
  <prop action="include" att="product" val="productA"/>
  <prop action="include" att="product" val="productB"/>
</val>
```

This simple DITAVAL file performs the following actions:
• The first `<prop>` element does not specify an attribute, which sets a default action of "exclude" for every prop value. This means that, by default, any property value not otherwise defined in this file evaluates to "exclude". Note that this same behavior can be limited to a single attribute; the following `<prop>` element sets a default action of "exclude" for all properties specified on the `@platform` attribute: `<prop action="exclude" att="platform"/>`
• The second and third `<prop>` elements set an action of "include" for two values on the `@audience` attribute. All other values on the `@audience` attribute still evaluate to "exclude".
• The fourth and fifth `<prop>` elements set an action of "include" for two values on the `@product` attribute. All other values on the `@product` attribute still evaluate to "exclude".

**Figure 98: DITAVAL file that overrides the default "include" action**

```xml
<val>
  <prop action="exclude" att="product" val="appserver"/>
  <prop action="include" att="product" val="mySERVER"/>
  <prop action="include" att="database" val="dbFIRST"/>
  <prop action="include" att="database" val="dbSECOND"/>
  <prop action="exclude" att="database" val="newDB"/>
</val>
```

Assume that "database" and "appserver" are used as group names within the `@product` attribute. In that case, the sample DITAVAL above performs the following actions:
• The first `<prop>` element excludes the value "appserver" when used within the `@product` attribute. It also sets a default of "exclude" for values within any appserver() group inside of the `@product` attribute.
• The second `<prop>` element sets "mySERVER" to include; this applies whether "mySERVER" appears alone in the `@product` attribute (`product="mySERVER"`) or inside of any group (`product="appserver(mySERVER)"` or `product="otherGroup(mySERVER)"`).
• The third and fourth `<prop>` elements set the database values "dbFIRST" and "dbSECOND" to include. If those values appear inside of a "database" group, they are explicitly set to "include". If they appear elsewhere in a conditional attribute (such as `product="dbFIRST"` or `platform="dbSECOND"`), this rule does not apply.
• The final `<prop>` element sets the database value "newDB" to exclude. If that value appears inside of a database group, it is explicitly set to "exclude". If it appears in any other group or attribute, this rule does not apply.
Remember that with groups, if all values inside of a single group evaluate to "exclude", that is equivalent to an entire attribute evaluating to "exclude", which results in the removal of the content. Using the above sample DITAVAL:

- `<p product="appserver">` is filtered out, because this value is excluded.
- `<p product="appserver(A B)">` is filtered out, because there is no explicit rule for A or B, and values in the "appserver" group inside of @product default to exclude.
- `<p product="appserver(A B mySERVER)">` is included, because `product="mySERVER"` evaluates to "include", which means the entire group evaluates to "include".
- `<p product="newDB">` is included, because no rule in the DITAVAL applies, so the "newDB" token defaults to "include".
- `<p product="database(newDB)">` is filtered out, because the token "newDB" is excluded when found in the database group.
- `<p product="database(dbFIRST dbSECOND newDB)">` is included, because both "dbFIRST" and "dbSECOND" are included, so the group evaluates to include.
- `<p product="database(newDB) appserver(mySERVER)">` is filtered out, because the token "newDB" is excluded when found in the database group. The entire "database" group on this paragraph evaluates to "exclude", so the element is excluded, regardless of how the "appserver" group evaluates.

**Note:** If two groups with the same name exist on different attributes, each group will evaluate the same way. For example, rules for the database group in this sample would evaluate the same whether the group is used within @product or @platform. See [Conditional processing (profiling)](see Conditional processing (profiling)) (see page 88) for suggestions on how to handle similar groups on different attributes.

*Figure 99: DITAVAL with conditions for groups*

**Attributes**

This element does not define any attributes.

4.9.2 `<style-conflict>`

The `<style-conflict>` element declares behavior to be used when one or more flagging methods collide on a single content element.

In case of conflicts between flagging methods at different levels (for example, a section is flagged green and a paragraph within the section is flagged red), the most deeply nested flagging method applies.

In case of conflicts between flagging methods on the same element (for example, a single element is being flagged with both green and red color), it is recommended that the conflicts be resolved as follows:

<table>
<thead>
<tr>
<th>Flagging method</th>
<th>Conflict behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;startflag&gt;</code> / <code>&lt;endflag&gt;</code></td>
<td>Add all flags that apply.</td>
</tr>
<tr>
<td>color</td>
<td>Follow the <code>&lt;style-conflict&gt;</code> @foreground-conflict-color setting, or use an output-appropriate default color if no conflict color is set.</td>
</tr>
<tr>
<td>backcolor</td>
<td>Follow the <code>&lt;style-conflict&gt;</code> @background-conflict-color setting, or use an output-appropriate default color if no conflict color is set.</td>
</tr>
<tr>
<td>style</td>
<td>Add all font styles that apply. If two different kinds of underline are used, default to the heaviest (double underline) and use the @foreground-conflict-color.</td>
</tr>
<tr>
<td>changebar</td>
<td>Add all change bars that apply.</td>
</tr>
</tbody>
</table>
Example

See the example in the <val> description.

Attributes

The following attributes are available on this element:

@foreground-conflict-color

<table>
<thead>
<tr>
<th>mp: question prompted by jeff ogden: should we document the @ convention for identifying attributes in the front part of the spec?</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC response: yes</td>
</tr>
<tr>
<td>The color to be used when more than one flagging color applies to a single content element. This attribute is defined with the XML Data Type CDATA.</td>
</tr>
</tbody>
</table>

@background-conflict-color

The color to be used when more than one flagging background color applies to a single content element. This attribute is defined with the XML Data Type CDATA.

4.9.3 <prop>

The <prop> element identifies an attribute, and usually values in the attribute, to take an action on. The attribute either must be a conditional-processing attribute (@platform, @product, @audience, @deliveryTarget, @props, and @otherprops) or a specialization of the @props attribute.

A <prop> element may do one of the following:

- A <prop> element with no @att attribute specified sets a default action for every <prop> element. It is an error to use more than one <prop> element with no attribute in a single document. Recovery from this error is implementation dependent; in such cases processors MAY provide an error or warning message.
- A <prop> element with an @att attribute but no @value attribute sets a default action for that specific attribute. For each specific attribute, it is an error to use more than one <prop> element with that attribute and no value in a single document. Recovery from this error is implementation dependent; in such cases processors MAY provide an error or warning message.
- A <prop> attribute with an @att attribute and a @value attribute sets an action for that value within that attribute. It is an error to use more than one <prop> element with the same attribute and value. Recovery from this error is implementation dependent; in such cases processors MAY provide an error or warning message.

Example

See the example in the <val> description.

Attributes

The following attributes are available on this element:

@att

The attribute to be acted upon. If using a literal attribute name, it must be one of @props, @audience, @platform, @product, @otherprops, @deliveryTarget, or a specialization of @props. Otherwise, the value should be the name of a group used within the @audience, @platform, @product, or @otherprops attributes. If the @att attribute is absent, then the <prop> element declares a default behavior for any conditional processing attribute. This attribute is defined with the XML Data Type CDATA.

@val

The value to be acted upon. If the @val attribute is absent, then the <prop> element declares a default behavior for any value in the specified attribute. This attribute is defined with the XML Data Type CDATA.
@action (REQUIRED)
The action to be taken. Allowable values are:

include
Include the content in output. This is the default behavior unless otherwise set.

exclude
Exclude the content from output (if all values in the particular attribute are excluded).

passthrough
Include the content in output, and preserve the attribute value as part of the output stream for further processing by a runtime engine, for example runtime filtering based on individual user settings. The value should be preserved in whatever syntax is required by the target runtime. Values that are not explicitly passed through should be removed from the output stream, even though the content is still included.

flag
Include and flag the content on output (if the content has not been excluded).

@color
If flag has been set, the color to use to flag text. Colors may be entered by name or code. Processors SHOULD support the color names listed under the heading "<color>" in http://www.w3.org/TR/2006/REC-xsl11-20061205/#datatype and for the 6 digit hex code form (#rrggb, case insensitive). If flag has not been set, this attribute is ignored. This attribute is defined with the XML Data Type CDATA.

@backcolor
If flag has been set, the color to use as background for flagged text. Colors may be entered by name or code. Processor support is recommended for the color names listed under the heading "<color>" in http://www.w3.org/TR/2006/REC-xsl11-20061205/#datatype and for the 6 digit hex code form (#rrggb, case insensitive). If flag has not been set, this attribute is ignored. This attribute is defined with the XML Data Type CDATA.

Comment by robander, 12 February 2014
If we run a scanner to update all instances of <element>, we should change "<color>" above back so that it does not use the element. It is a title, not an XML Element that has varying style.

@style
If flag has been set, the text styles to use for flagged text. This attribute can contain multiple space-delimited tokens. The following tokens SHOULD be processed by all DITAVAL processors:

• underline
• double-underline
• italics
• overline
• bold

In addition, processors may support other proprietary tokens for different types of styling. Such tokens SHOULD have a processor-specific prefix to identify them as proprietary. If a processor encounters an unsupported style token, it MAY issue a warning, and MAY render content flagged with such a style token using some default formatting.

If flag has not been set, this attribute is ignored.

4.9.4 <revprop>
Identifies a value in the @rev attribute that should be flagged in some manner. Unlike the other conditional processing attributes, which may be used for both filtering and flagging, the @rev attribute may only be used for flagging.

It is an error to include more than one <revprop> element with the same @val attribute setting. Recovery from this error is implementation dependent; in such cases processors MAY provide an error or warning message.
When no alternate text is specified for a revision flag, the default alternate text for `<revprop>` start of change is a localized translation of "Start of change", and the default alternate text for `<revprop>` end of change is a localized translation of "End of change".

The @rev attribute identifies when a particular section of a document was added in the lifetime of that document. The attribute is not considered a filtering attribute because this is not sufficient to be used for full version control, such as single-sourcing multiple product variants based on version level – it only represents one aspect of the revision level.

**Example**

See the example in the `<val>` description.

**Attributes**

The following attributes are available on this element:

@val

The value to be acted upon. If the @val attribute is absent, then the `<revprop>` element declares a default behavior for any value in the @rev attribute.

@action *(REQUIRED)*

The action to be taken. Allowable values are:

- **include**
  
  Include the content in output without flags. This is the default behavior unless otherwise set.

- **passthrough**
  
  Include the content in output, and preserve the attribute value as part of the output stream for further processing by a runtime engine, for example runtime filtering based on individual user settings. The value should be preserved in whatever syntax is required by the target runtime. Values that are not explicitly passed through should be removed from the output stream, even though the content is still included.

- **flag**
  
  Include and flag the content on output (if the content has not been excluded).

@changebar

When flag has been set, specify a changebar color, style, or character, according to the changebar support of the target output format. If flag has not been set, this attribute is ignored. This attribute is defined with the XML Data Type CDATA.

@color

If flag has been set, the color to use to flag text. Colors may be entered by name or code. Processors SHOULD support the color names listed under the heading "<color>" in http://www.w3.org/TR/2006/REC-xsl11-20061205/#datatype and for the 6 digit hex code form (#rrggb, case insensitive). If flag has not been set, this attribute is ignored. This attribute is defined with the XML Data Type CDATA.

@backcolor

If flag has been set, the color to use as background for flagged text. Colors may be entered by name or code. Processor support is recommended for the color names listed under the heading "<color>" in http://www.w3.org/TR/2006/REC-xsl11-20061205/#datatype and for the 6 digit hex code form (#rrggb, case insensitive). If flag has not been set, this attribute is ignored. This attribute is defined with the XML Data Type CDATA.

Comment by robander, 12 February 2014

If we run a scanner to update all instances of `<element>`, we should change "<color>" above back so that it does not use the element. It is a title, not an XML Element that has varying style.

@style

If flag has been set, the text styles to use for flagged text. This attribute can contain multiple space-delimited tokens. The following tokens SHOULD be processed by all DITAVAL processors:
• underline
• double-underline
• italics
• overline
• bold

In addition, processors may support other proprietary tokens for different types of styling. Such tokens SHOULD have a processor-specific prefix to identify them as proprietary. If a processor encounters an unsupported style token, it MAY issue a warning, and MAY render content flagged with such a style token using some default formatting.

If flag has not been set, this attribute is ignored.

4.9.5 <startflag>

The <startflag> takes an optional image reference and an optional <alt-text> element for identifying the beginning of flagged content. If an image is specified, the specified image will be used to flag the beginning of the content, with the <alt-text> contents as alternative text. If <alt-text> is specified without an image, that text will be used to flag the content instead of an image. If no image and no <alt-text> are specified, then this element has no defined purpose. The <startflag> element is allowed inside either <prop> or <revprop>.

Example
See the example in the <val> description.

Attributes
The following attribute is available on this element:
@imageref

Provides a URI reference to the image file, using the same syntax as the @href attribute. See The href attribute (see page 663) for information on supported values and processing implications. This attribute is defined with the XML Data Type CDATA.

4.9.6 <endflag>

The <endflag> element takes an optional image reference and an optional <alt-text> element for identifying the end of flagged content. If an image is specified, the specified image will be used to flag the end of the content, with the <alt-text> contents as alternative text. If <alt-text> is specified without an image, that text will be used to flag the content instead of an image. If no image and no <alt-text> are specified, then this element has no defined purpose. The <endflag> element is allowed inside either <prop> or <revprop>.

Example
See the example in the <val> description.

Attributes
The following attribute is available on this element:
@imageref

Provides a URI reference to the image file, using the same syntax as the @href attribute. See The href attribute (see page 663) for information on supported values and processing implications. This attribute is defined with the XML Data Type CDATA.
4.9.7 <alt-text>

The <alt-text> element is allowed inside either <startflag> or <endflag> to provide alternate text for an image when the @imageref attribute sets an image to be used for flagging. When no alternate text is specified for a revision flag, the default alternate text for <revprop> start of change is a localized translation of "Start of change", and the default alternate text for <revprop> end of change is a localized translation of "End of change".

Example

See the example in the <val> description.

Attributes

This element does not define any attributes.

4.10 Technical content elements

Elements in the technical content section include the original Concept, Task, and Reference specializations, as well as the Bookmap and Glossary specializations added with DITA 1.1. It also includes domains designed primarily for technical content, such as the task requirements and software domains.

4.10.1 Concept elements

DITA concept topics answer "What is..." questions. Use the concept topic to introduce the background or overview information for tasks or reference topics. The concept topic restricts content following a section or example to other sections or examples. For more details on when to use concept and other information types, please refer to the DITA architectural specification.

4.10.1.1 <concept>

The <concept> element is the top-level element for a topic that answers the question “what is?” 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 have an example or a graphic, but generally the structure of a concept is fairly simple.

Inheritance

- topic/topic concept/concept

Example

```
<concept id="concept">
 <title>Introduction to Bird Calling</title>
 <shortdesc>If you wish to attract more birds to your Acme Bird Feeder, learn the art of bird calling. Bird calling is an efficient way to alert more birds to the presence of your bird feeder.</shortdesc>
 <conbody>
  <p>Bird calling requires learning:</p>
  <ul>
   <li>Popular and classical bird songs</li>
   <li>How to whistle like a bird</li>
  </ul>
 </conbody>
</concept>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below), Architectural attribute group (see page 654), and outputclass (see page 0 ).

@id (REQUIRED)
An anchor point. This ID is usually required as part of the @href or @conref syntax when cross referencing or reusing content within the topic; it also enables <topicref> elements in DITA maps to optionally reference a specific topic within a DITA document. This attribute is defined with the XML Data Type ID.

### 4.10.1.2 <conbody>

The <conbody> element is the main body-level element for a concept.

Like the <body> element of a general <topic>, <conbody> allows paragraphs, lists, and other elements as well as sections and examples. However, <conbody> has a constraint that a <section> or an <example> can be followed only by other sections, examples, or <conbodydiv> elements that group sections and examples.

**Inheritance**

- topic/body concept/conbody

**Example**

See the example in concept (see page 421).

**Attributes**

The following attributes are available on this element: Universal attribute group (see page 650) (without the Metadata attribute group), @base from the Metadata attribute group (see page 651), and outputclass (see page 0).

### 4.10.1.3 <conbodydiv>

The <conbodydiv> element is similar to the <bodydiv> element in that it provides an informal container for content that may be grouped within a concept. There are no additional semantics attached to the <conbodydiv> element; it is purely a grouping element provided to help organize content.

The parent <conbody> element has a restriction that <section> or <example> elements can only be followed by other sections or examples. The <conbodydiv> element, which allows groupings of sections and examples, keeps the same restriction in place; once used, only sections, examples, or other <conbodydiv> groups are allowed.

**Inheritance**

- topic/bodydiv concept/conbodydiv

**Example**

One common use case for the <conbodydiv> element is to group a sequence of sections for reuse, so that another concept may reference the entire set with a single @conref attribute.

```xml
<concept id="sample" xml:lang="en">
    <title>Conbodydiv example</title>
    <shortdesc>This concept is a sample of how to use conbodydiv.</shortdesc>
    <conbody>
        <p>Introduce the example.</p>
        <p>Next group some sections that may be reused elsewhere.</p>
        <conbodydiv id="my_conbodydiv">
            <section><title>First</title> ... </section>
            <section><title>Second</title> ... </section>
        </conbodydiv>
        <p>Next group some sections that may be reused elsewhere.</p>
    </conbody>
</concept>
```
**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

### 4.10.2 Task elements

Task topics answer "How do I?" questions, and have a well-defined structure that describes how to complete a procedure to accomplish a specific goal. Use the task topic to describe the steps of a particular task, or to provide an overview of a higher-level task. The task topic includes sections for describing the context, prerequisites, actual steps, expected results, example, and expected next steps for a task. For more details on when to use task and other information types, please refer to the DITA architectural specification.

#### 4.10.2.1 `<task>`

The `<task>` element is the top-level element for a task topic. Tasks are the main building blocks for task-oriented user assistance. They generally provide step-by-step instructions that will enable a user to perform a task. A task answers the question of "how to?" by telling the user precisely what to do and the order in which to do it. Tasks have the same high-level structure as other topics, with a title, short description and body.

**Note:** Beginning with DITA 1.2, the DTD and Schema packages distributed by OASIS contain two task models. The general task model allows two additional elements inside the task body (<section> and <steps-informal>); it also allows multiple instances and varying order for the <prereq>, <context>, and <section> elements. The strict task model maintains the order and cardinality of the DITA 1.0 and 1.1 `<taskbody>` content model. This strict task is implemented in the DTD and Schema with a constraint module.

See the *taskbody* description for additional details about the two models and for a description of impacts to DITA 1.1 documents.

#### Inheritance

- topic/topic task/task

#### Example

```xml
<task id="sqlj">
  <title>Creating an SQLJ file</title>
  <taskbody>
    <context>Once you have set up SQLJ, you need to create a new SQLJ file.</context>
    <steps>
      <step><cmd>Open... </cmd></step>
    </steps>
  </taskbody>
</task>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of @id, given below), *Architectural attribute group* (see page 654), and *outputclass* (see page 0).

**@id (REQUIRED)**

An anchor point. This ID is usually required as part of the @href or @conref syntax when cross referencing or reusing content within the topic; it also enables `<topicref>` elements in DITA maps to optionally reference a specific topic within a DITA document. This attribute is defined with the XML Data Type ID.
4.10.2.2 taskbody

The `<taskbody>` element is the main body-level element inside a task topic. A task body is designed to contain information specific to completing a task, such as prerequisites, contextual information, and steps. DITA 1.2 introduces a much looser `<taskbody>` content model in order to allow for more variations in the structure of a task. A constraint module is also provided in order to maintain compatibility with the previous strict model; this constraint is used in the default task distributed by OASIS.

**Note:** Beginning with DITA 1.2, the DTD and Schema packages distributed by OASIS contain two task models. The general task model allows two additional elements inside the task body (`<section>` and `<steps-informal>`); it also allows multiple instances and varying order for the `<prereq>` and `<context>` elements. The strict task model maintains the order and cardinality of the DITA 1.0 and 1.1 `<taskbody>` content model. This strict task is implemented in the DTD and Schema with a constraint module.

Authors that use the default task DTD or Schema provided by OASIS will continue to see the strict task model when upgrading to DITA 1.2. Authors wishing to use the general task model will need to migrate their DITA 1.1 documents to reference the general task DTD or Schema.

DITA document type shells that include the task module as-is, or that specialize the `<task>` element without specializing `<taskbody>`, will also need to include the strict taskbody constraint module in order to maintain the order and cardinality of prior DITA versions.

Task specializations that specialize the `<taskbody>` element will not be affected by the new model, although they may be updated as needed to take advantage of the new elements.

**Inheritance**
- topic/body task/taskbody

**Example**

See task (see page 423).

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) (without the Metadata attribute group), `@base` from the *Metadata attribute group* (see page 651), and `outputclass` (see page 0).

4.10.2.3 <prereq>

The `<prereq>` element of a task contains prerequisites that the user needs to know or do before starting the current task. Implementations MAY render prerequisite links from the related-links section together with the `<prereq>` content.

**Inheritance**
- topic/section task/prereq

**Example**

```
<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>
```
**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

### 4.10.2.4 <context>

The `<context>` element provides background information for a task. This information helps the user understand what the purpose of the task is and what they will gain by completing the task. This section should be brief and does not replace or recreate a concept topic on the same subject, although the context section may include some conceptual information.

**Inheritance**
- topic/section task/context

**Example**

```xml
<task id="sqlj">
  <title>Creating an SQLJ file</title>
  <taskbody>
    <context>Once you have set up SQLJ, you need to create a new SQLJ file.</context>
  </taskbody>
</task>
```

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

### 4.10.2.5 <steps>

The `<steps>` element provides the main content of a task topic. The task is described as a series of steps that the user must follow to accomplish the task. At least one `<step>` element is required inside the `<steps>` element.

Steps with only a single step may be rendered as a paragraph rather than as a list. Two or more steps should typically be rendered as an ordered list. If all of the contained steps are simple (that is, have no more than a `<cmd>` element each) then the step list should default to compact. Otherwise it should be rendered as expanded (with blank lines between each step).

**Note:** Beginning with DITA 1.2, the general task model allows multiple `<steps>` and `<steps-unordered>` elements. However, the default task model in the OASIS distribution (known as strict task) continues to allow only one `<steps>` or one `<steps-unordered>` element.

**Inheritance**
- topic/ol task/steps

**Example**

```xml
<task id="sqlj">
  <title>Creating an SQLJ file</title>
  <taskbody>
    <context>Once you have set up SQLJ, you need to 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>
```
4.10.2.6 <steps-informal>

The <steps-informal> element allows authors to describe procedural task information without placing each step in an individual container element, which is a requirement of the related <steps> and <steps-unordered> elements. For example, <steps-informal> may contain a paragraph that describes more than one step in a single sentence, or it may contain sentences that mix steps together with information about the steps.

Inheritance
- topic/section task/steps-informal

Example

```xml
<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, water appropriately and wait.</p>
</steps-informal>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.10.2.7 <steps-unordered>

Like the <steps> element, the <steps-unordered> element provides the main content of a task topic, but particularly for cases in which the order of steps may vary from one situation to another. At least one <step> element is required inside the <steps-unordered> element.

Steps with only a single step may be rendered as a paragraph rather than as a list. Two or more steps should typically be rendered as an unordered list. If all of the contained steps are simple (that is, have no more than a <cmd> element each) then the step list should default to compact. Otherwise it should be rendered as expanded (with blank lines between each step).

Note: Beginning with DITA 1.2, the general task model allows multiple <steps> and <steps-unordered> elements. However, the default task model in the OASIS distribution (known as strict task) continues to allow only one <steps> or one <steps-unordered> element.

Inheritance
- topic/ul task/steps-unordered

Example

```xml
<task id="sqlj">
  <title>Creating an SQLJ file</title>
  <taskbody>
    <context>Once you have set up SQLJ, you need to create a new SQLJ file.</context>
    <steps-unordered>
      <step><cmd>In a text editor, create a new file.</cmd></step>
    </steps-unordered>
  </taskbody>
</task>
```
Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.10.2.8 `<step>`

The `<step>` element represents an action that a user must follow to accomplish a task. Each step in a task must contain a command `<cmd>` element which describes the particular action the user must do to accomplish the overall task. Beginning with DITA 1.2, it is possible to place a `<note>` element before the command in order to notify the user of dangers or other important information about the step. The `<step>` element can also contain additional optional information about the step, such as sub-steps, a list of choices, or result information.

Inheritance

- topic/li task/step

Example

```xml
<task id="sqlj">
  <title>Creating an SQLJ file</title>
  <taskbody>
    <context>Once you have set up SQLJ, you need to create a new SQLJ file.</context>
    <steps>
      <step>
        <cmd>Select <menucascade><uicontrol>File</uicontrol><uicontrol>New</uicontrol></menucascade>.</cmd>
        <info>New files are created with default values based on a standard template.</info>
      </step>
    </steps>
  </taskbody>
</task>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of `@importance`, given below) and *outputclass* (see page 0).

- **importance**
  
  Describes whether the current `<step>` or `<substep>` is optional or required. Output processors may (but need not) highlight steps that are optional or required. Allowed values are "optional", "required", or `-dita-use-conref-target`.

4.10.2.9 `<stepsection>`

The `<stepsection>` element provides expository text before a `<step>` element. Although the element is specialized from `<li>` and has the same content model as a list item, it is not intended to represent a step in a task.

Note: DITA applications which render `<stepsection>` elements among the `<step>` elements must provide a way to number the steps without numbering the `<stepsection>` elements (although this does not need to be the only or default presentation).

Inheritance

- topic/li task/stepsection
Example

```xml
<steps>
  <step><cmd>Get out a bowl</cmd></step>
  <stepsection>The next two steps are very important!</stepsection>
  <step><cmd>Put on safety gloves</cmd></step>
  <step><cmd>Put on goggles</cmd></step>
  <step><cmd>Pour milk and cereal into the bowl</cmd></step>
</steps>
```

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

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.10.2.10 <cmd>

The <cmd> element specifies a command, which is a required element inside the <step> element. It provides the active voice instruction to the user for completing the step, and should not be more than one sentence. If the step needs additional explanation, place the explanation in an <info> element following the <cmd>.

Inheritance

- topic/ph task/cmd

Example

```xml
<step><cmd>In a text editor, create a new file.</cmd></step>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and @keyref.

4.10.2.11 <info>

The <info> element occurs inside a <step> element to provide additional information about the step.

Inheritance

- topic/itemgroup task/info

Example

```xml
<step><cmd>Type a name for the widget.</cmd>
<info>The widget name is created when you configure the widget in the Widget Configuration Dialog. It is not an actual class name or file name, just a label for the widget as used in this application.</info>
</step>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).
4.10.2.12 `<substeps>`

The `<substeps>` element allows you to break a step down into a series of separate actions, and should be used only if necessary. Try to describe the steps of a task in a single level of steps. If you need to use more than one level of `<substep>` nesting, you should probably rewrite the task to simplify it.

**Inheritance**
- topic/ol task/substeps

**Example**

```xml
<substeps>
  <substep><cmd>Hold pencil in a steady, level position.</cmd></substep>
  <substep><cmd>Turn handle until resistance diminishes.</cmd></substep>
  <info>Note: initially, it may be somewhat difficult to turn the handle if pencil has never been sharpened before.</info></substep>
  <substep><cmd>To determine if pencil is sharp, remove it from the sharpener and inspect the tip.</cmd></substep>
</substeps>
```

**Attributes**
The following attributes are available on this element: `Universal attribute group` (see page 650) and `outputclass` (see page 0).

4.10.2.13 `<substep>`

A `<substep>` element has the same structure as a `<step>`, except that it does not allow lists of choices or sub-steps within it, in order to prevent unlimited nesting of steps.

**Inheritance**
- topic/li task/substep

**Example**

See `<substeps>` (see page 429).

**Attributes**
The following attributes are available on this element: `Universal attribute group` (see page 650) (with a narrowed definition of `@importance`, given below) and `outputclass` (see page 0).

- `importance`
  Describes whether the current `<step>` or `<substep>` is optional or required. Output processors may (but need not) highlight steps that are optional or required. Allowed values are "optional", "required", or `-dita-use-conref-target`.

4.10.2.14 `<stepxmp>`

The `<stepxmp>` element is used to illustrate a step of a task. The step example can be a couple of words, or an entire paragraph.

**Inheritance**
- topic/itemgroup task/stepxmp
Example

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

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.10.2.15 <choicetable>

The <choicetable> element contains a series of optional choices available within a step of a task.

By default, processors should highlight the choice column using bold. To change the highlighting, set the @keycol attribute of the <choicetable> tag to 0 (zero).

Inheritance

- topic/simpletable task/choicetable

```xml
<step><cmd>Then this</cmd>
  <substeps>
    <substep importance="optional"><cmd>which is done by doing this</cmd></substep>
    <substep importance="required"><cmd>and then this.</cmd></substep>
  </substeps>
</step>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Display attribute group (see page 660), Simpletable attribute group (see page 662) (with a narrowed definition of @keycol, given below), outputclass (see page 0), and spectitle (see page 0).

@keycol

On this element, the default value for @keycol is "1". Otherwise, the attribute is the same as defined in Simpletable attribute group (see page 662).

4.10.2.16 <chhead>

The <chhead> element is a container inside the <choicetable> element that provides specific heading text to override any default headings for the <choicetable> (such as "Options" and "Description"). The <chhead> element contains both a <choptionhd> and <chdeschd> element as a pair.

Inheritance

- topic/sthead task/chhead
Example

```dita
<step><cmd>Then this</cmd>
  <substeps>
    <substep importance="optional"><cmd>which is done by doing this</cmd></substep>
    <substep importance="required"><cmd>and then this.</cmd></substep>
  </substeps>
  <choicetable>
    <chhead>
      <choptionhd>Do something</choptionhd>
      <chdeschd>Or Else this</chdeschd>
    </chhead>
    <chrow><choption>Do this</choption>
      <chdesc>and this will happen</chdesc></chrow>
    <chrow><choption>Do that</choption>
      <chdesc>and that will happen</chdesc></chrow>
  </choicetable>
</step>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.10.2.17 <choptionhd>

The <choptionhd> element provides a specific label for the list of options from which a user chooses in order to accomplish a step. The default label for the list of options could be a localized translation of Option.

Comment by robander, 25 February 2014

Original text: "The default label for options is a localized translation of <b>Option</b>." This seems inappropriate, and also conflicts with the previous default given in the <chhead> topic, which was the plural "Options".

Inheritance

- topic/stentry task/choptionhd

Example

```dita
<step><cmd>Then this</cmd>
  <choicetable>
    <chhead>
      <choptionhd>Do something</choptionhd>
      <chdeschd>And this happens</chdeschd>
    </chhead>
    <chrow><choption>Do this</choption>
      <chdesc>and this will happen</chdesc></chrow>
    <chrow><choption>Do that</choption>
      <chdesc>and that will happen</chdesc></chrow>
  </choicetable>
</step>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and specentry (see page 0).

4.10.2.18 <chdeschd>

The <chdeschd> element provides a specific label for the list of descriptions of options from which a user must choose in order to accomplish a step. The default label for the list of options could be a localized translation of Description.
Comment by robander, 25 February 2014
Original text: "The default label overridden by `<chdeschd>` is a localized translation of `Description`. This seems inappropriate, updated to match a similar update in the `choptionhd` topic.

**Inheritance**

- topic/stentry task/chdeschd

**Example**

```xml
<step><cmd>Then this</cmd>
  <choicetable>
    <chhead>
      <choptionhd>Do something</choptionhd>
      <chdeschd>Or Else this</chdeschd>
    </chhead>
    <chrow><choption>Do this</choption>
      <chdesc>and this will happen</chdesc></chrow>
    <chrow><choption>Do that</choption>
      <chdesc>and that will happen</chdesc></chrow>
  </choicetable>
</step>
```

**Attributes**

The following attributes are available on this element: `Universal attribute group` (see page 650), `outputclass` (see page 0), and `specentry` (see page 0).

4.10.2.19 `<chrow>`

The `<chrow>` element is a container inside the `<choicetable>` element. The `<chrow>` element contains both a `<choption>` and `<chdesc>` element as a pair.

**Inheritance**

- topic/strow task/chrow

**Example**

```xml
<step><cmd>Then this</cmd>
  <substeps>
    <substep importance="optional"><cmd>which is done by doing this</cmd></substep>
    <substep importance="required"><cmd>and then this.</cmd></substep>
  </substeps>
  <choicetable>
    <chhead>
      <choptionhd>Do something</choptionhd>
      <chdeschd>Or Else this</chdeschd>
    </chhead>
    <chrow><choption>Do this</choption>
      <chdesc>and this will happen</chdesc></chrow>
    <chrow><choption>Do that</choption>
      <chdesc>and that will happen</chdesc></chrow>
  </choicetable>
</step>
```

**Attributes**

The following attributes are available on this element: `Universal attribute group` (see page 650) and `outputclass` (see page 0).
4.10.2.20 <choption>

The <choption> element describes an option that a user could choose to accomplish a step of a task. In a user interface, for example, this might be the name of radio button.

Inheritance
- topic/stentry task/choption

Example

```
<step><cmd>Then this</cmd>
<choicetable>
  <chhead>
    <choptionhd>Do something</choptionhd>
    <chdeschd>And this happens</chdeschd>
  </chhead>
  <chrow>
    <choption>Do this</choption>
    <chdesc>and this will happen</chdesc></chrow>
  <chrow>
    <choption>Do that</choption>
    <chdesc>and that will happen</chdesc></chrow>
</choicetable>
</step>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and specentry (see page 0 ).

4.10.2.21 <chdesc>

The <chdesc> element is a description of an option that a user chooses while performing a step to accomplish a task. It explains why the user would choose that option and might explain the result of the choice when it is not immediately obvious.

Inheritance
- topic/stentry task/chdesc

Example

```
<step><cmd>Then this</cmd>
<substeps>
  <substep importance="optional"><cmd>which is done by doing this</cmd></substep>
  <substep importance="required"><cmd>and then this</cmd></substep>
</substeps>
<choicetable>
  <chrow>
    <choption>Do this</choption>
    <chdesc>and this will happen</chdesc>
  </chrow>
  <chrow>
    <choption>Do that</choption>
    <chdesc>and that will happen</chdesc>
  </chrow>
</choicetable>
</step>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and specentry (see page 0 ).
4.10.2.22 <choices>

The <choices> element contains a list of <choice> elements. It is used when the user will need to choose one of several actions while performing the steps of a task.

Inheritance
- topic/ul task/choices

Example

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

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.10.2.23 <choice>

Each <choice> element describes one way that the user could accomplish the current step.

Inheritance
- topic/li task/choice

Example

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

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.10.2.24 <steptroubleshooting>

The <steptroubleshooting> element provides information that is designed to help remedy the situation when a step does not complete as expected. In particular, this element can be used to explain how users can recover when the results of a step do not match those listed in the <stepresult> element.

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

Inheritance
- topic/itemgroup task/steptroubleshooting
Example

The following example illustrates using the `<steptroubleshooting>` element with a single action:

```xml
<step>
  <cmd>Select <uicontrol>Perform system backup</uicontrol></cmd>
  <stepresult>
    <p>The message <systemoutput>Backup successfully completed</systemoutput> displays.</p>
  </stepresult>
  <steptroubleshooting>
    <p>If an error message displays during the system backup, locate the error ID in the <cite>Troubleshooting Guide</cite> and follow the instructions there.</p>
  </steptroubleshooting>
</step>
```

The following example illustrates using the `<steptroubleshooting>` element with several actions:

```xml
<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:
      <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>
    </p>
  </steptroubleshooting>
</step>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.10.2.25 `<stepresult>`

The `<stepresult>` element provides information on the expected outcome of a step. If a user interface is being documented, the outcome could describe a dialog box opening or the appearance of a progress indicator. Step results are useful to assure a user that they are on track, but should not be used for every step as this quickly becomes tedious.

Inheritance

- topic/itemgroup task/stepresult

Example

```xml
<steps>
  <step>
    <cmd importance="urgent">Once you have the water place it in the microwave.</cmd>
    <info>Try not to spill any, as water is very wet.</info>
  </step>
  <step importance="required">
    <cmd>Start the Microwave.</cmd>
    <stepxmp>As an example, push the <b>Start</b> button</stepxmp>
    <stepresult>The Microwave starts running. You should hear it humming.</stepresult>
  </step>
  <step importance="optional">...
```
Once the water begins to boil, stop the Microwave.

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.10.2.26 **tutorialinfo**

The *tutorialinfo* element contains additional information that is useful when the task is part of a tutorial.

**Inheritance**

- topic/itemgroup task/tutorialinfo

**Example**

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

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.10.2.27 **tasktroubleshooting**

The *tasktroubleshooting* element provides information designed to help users remedy the situation when a task does not complete as expected. In particular, this 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.

**Tip:** Do not use *note type="troubleshooting"* inside of *tasktroubleshooting* because its meaning there would be ambiguous.

**Inheritance**

- topic/section task/tasktroubleshooting

**Example**

```xml
<steps>
  <step><cmd>...</cmd></step>
  <!-- ... more steps ... -->
</steps>

<result>
  <p>The <uicontrol>User Type</uicontrol> menu updates to display the new types you added.</p>
</result>
<tasktroubleshooting>
```
If the User Type menu does not display the additions, try one or more of the following:

- Refresh the page
- Verify that <wintitle>Add Types</wintitle> is not still open; if so, go to it and press <uicontrol>OK</uicontrol>.

For example, you could also do xyz.

Once completed, you need to consider abc.

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

**4.10.2.28 <result>**

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

**Note:** To describe the outcome of specific step, use the <stepresult> element instead.

**Inheritance**

- topic/section task/result

**Example**

```
<task id="sqlj">
  <title>Creating an SQLJ file</title>
  <context>Once you have set up SQLJ, you need to create a new SQLJ file. You cannot add #sqlj statements directly in the Source pane of the Workbench.</context>
  <result>The SQLJ file is successfully created when the SQLJ server displays the "File Created" dialog.</result>
</task>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

**4.10.2.29 <postreq>**

The <postreq> element describes steps or tasks that the user should do after the successful completion of the current task. It is often supported by links to the next task or tasks in the <related-links> section.

**Inheritance**

- topic/section task/postreq
Example

<postreq>Notify the proctor upon completing this self-test.</postreq>

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.10.3 Reference elements

Reference topics describe factual material about a subject, such as the commands in a programming language. This format is also suitable for bibliographies, catalogues, the list of ingredients for recipes, and similar collections of structured descriptive prose. For more details on when to use reference and other information types, please refer to the DITA architectural specification.

4.10.3.1 <reference>

The <reference> element defines a top-level container for a reference topic. Reference topics document programming constructs or facts about a product. Examples of reference topics include (but are not limited to) product specifications, environmental specifications, equipment lists, parts lists, required tools, language elements, class descriptions, commands, functions, and API information. All of these items provide quick access to facts, but no deeper explanation of related concepts or tasks. Reference topics have the same high-level structure as any other topic type, with a title, short description, and body. Within the body, reference topics are typically organized into one or more sections, property lists, and tables. The reference topic type provides general rules that apply to all kinds of reference information, using elements like <refsyn> for syntax or signatures, and <properties> for lists of properties and values.

Inheritance
- topic/topic reference/reference

Example

<reference id="refexample">
<title>A reference topic</title>
<refbody>
<refsyn>Describe command or api syntax here, possibly using synph or syntax elements markup for explicit definition of syntax or prototype construction.</refsyn>
<section><title>Some section title</title></section>
<properties>
<property>
<proptype>type</proptype>
<propvalue>value</propvalue>
<propdesc>description</propdesc>
</property>
</properties>
</refbody>
</reference>

Figure 100: Reference topic for software material

The following information could apply to an entire set of maintenance procedures, each of which would link to this topic.

<reference id="requiredTools">
<title>Tools required to maintain a big machine</title>
<refbody>
<section>
<title>Small tools</title>
<ul>
<li>Hard hat</li>
</ul>
</section>
</reference>
Attributes

The following attributes are available on this element: **Universal attribute group** (see page 650) (with a narrowed definition of @id, given below), **Architectural attribute group** (see page 654), and **outputclass** (see page 0).

@id (REQUIRED)

An anchor point. This ID is usually required as part of the @href or @conref syntax when cross referencing or reusing content within the topic; it also enables <topicref> elements in DITA maps to optionally reference a specific topic within a DITA document. This attribute is defined with the XML Data Type ID.

4.10.3.2 <refbody>

The <refbody> element is a container for the main content of the reference topic. Reference topics limit the body structure to tables (both simple and standard), property lists, syntax sections, and generic sections and examples, in any sequence or number.

Reference topics represent the kind of information that users typically consult to understand programming objects, configuration file options, recipes, terminological descriptions, product or other specifications, equipment or parts lists, or any other set of factual information.

**Inheritance**

- topic/body reference/refbody

**Example**

See reference (see page 438).

Attributes

The following attributes are available on this element: **Universal attribute group** (see page 650) (without the Metadata attribute group), @base from the **Metadata attribute group** (see page 651), and **outputclass** (see page 0).

4.10.3.3 <refbodydiv>

The <refbodydiv> element is similar to the <bodydiv> element in that it provides an informal container for content that may be grouped within a reference. Reference topics place many restrictions on their content compared to generic topics; the <refbodydiv> element maintains these restrictions by only allowing elements that are already available within the body of a reference. There are no additional semantics attached to the <refbodydiv> element; it is purely a grouping element provided to help organize content.

The <refbodydiv> element may nest itself, which means that it may be specialized to create structured information within a specialized reference topic. Another common use case for the <refbodydiv> element is to group a sequence of related elements for reuse, so that another topic may reference the entire set with a single @conref attribute.
Inheritance

Example

```xml
<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>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.10.3.4 <refsyn>

The <refsyn> element is a special section inside a reference topic. The section often contains syntax or signature content (for example, a command-line utility's calling syntax, or an API's signature). The <refsyn> contains a brief, possibly diagrammatic description of the subject's interface or high-level structure.

Inheritance

Example

```xml
<reference id="MyAPI">
<title>MyAPI</title>
<refbody>
  <refsyn>Describe the MyAPI syntax here, possibly using synph or syntax elements markup for explicit definition of syntax or prototype construction.</refsyn>
</refbody>
</reference>
```

Figure 102: Reference topic for software material

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *spectitle* (see page 0).

4.10.3.5 <properties>

The <properties> element gives a list of properties for the subject of the current topic, for example whether a class is public or protected. Each property can include the type, value, and a description. The typical rendering is usually in a table-like format. To represent multiple values for a single type, create additional property elements and use only the <propvalue> element (and <propdesc> when needed) for each successive value.
Inheritance
- topic/simpletable reference/properties

Example

```xml
<properties>
  <prophead>
    <proptypehd>Visual Element</proptypehd>
    <propvaluehd>Value</propvaluehd>
    <propdeschd>Implication</propdeschd>
  </prophead>
  <property>
    <proptype>color</proptype>
    <propvalue>red</propvalue>
    <propdesc>depicts anger</propdesc>
  </property>
  <property>
    <propvalue>green</propvalue>
    <propdesc>depicts permission</propdesc>
  </property>
</properties>
```

Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650), *Display attribute group* (see page 660), *Simpletable attribute group* (see page 662), *outputclass* (see page 0), and *spectitle* (see page 0).

4.10.3.6 <prophead>

The `<prophead>` element supports headings for the `<properties>` element.

Inheritance
- topic/sthead reference/prophead

Example
See the example in *properties* (see page 440).

Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.10.3.7 <proptypehd>

The `<proptypehd>` element supports headings for the type column of a properties table.

Inheritance
- topic/stentry reference/proptypehd

Example
See the example in *properties* (see page 440).

Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *specentry* (see page 0).
4.10.3.8 <propvaluehd/>

The <propvaluehd/> element supports headings for the value column of a properties table.

Inheritance
- topic/stentry reference/propvaluehd

Example
See the example in properties (see page 440).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and specentry (see page 0).

4.10.3.9 <propdeschd/>

The <propdeschd/> element supports headings for the description column of a properties table.

Inheritance
- topic/stentry reference/propdeschd

Example
See the example in properties (see page 440).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and specentry (see page 0).

4.10.3.10 <property>

The <property/> element represents a single property of the current topic's subject. For example, if the current reference topic describes a programming class, the property might show that the class is protected rather than public. The <property/> element generally appears together with a series of other properties; it contains three optional elements to provide a type, value, or description of the property.

Inheritance
- topic/strow reference/property

Example
See properties (see page 440).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).
4.10.3.11 <proptype>

The <proptype> element describes the type of the property.

Inheritance
- topic/stentry reference/proptype

Example
See properties (see page 440).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and specentry (see page 0 ).

4.10.3.12 <propvalue>

The <propvalue> element indicates one or more values for the current property type. Values may be placed separate <property> elements if they need separate descriptions. The <proptype> element need not be repeated.

Inheritance
- topic/stentry reference/propvalue

Example
See properties (see page 440).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and specentry (see page 0 ).

4.10.3.13 <propdesc>

The <propdesc> element is used to provide a short description of the property type and its listed values.

Inheritance
- topic/stentry reference/propdesc

Example
See properties (see page 440).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and specentry (see page 0 ).
4.10.4 Troubleshooting elements

4.10.4.1 <troubleshooting>

The <troubleshooting> element is the top-level element for a troubleshooting topic. Troubleshooting topics document corrective action such as troubleshooting or alarm clearing. Troubleshooting topics begin with a description of a condition that the reader might want to correct, followed by one or more cause-remedy pairs. Each cause-remedy pair is a potential solution to the trouble described in the condition.

Troubleshooting topics represent the kind of information that users typically consult to fix a problem.

Inheritance
- topic/topic troubleshooting/troubleshooting

Example

```xml
<troubleshooting id="resolve" xml:lang="en-us">
  <title>E247 - Memory fault has occurred</title>
  <shortdesc>The system has detected a problem in memory.</shortdesc>
  <troublebody>
    <condition>
      <p>The fault indicator flashes on the front panel, and the error log contains the following message: E247 - Memory fault has occurred.</p>
    </condition>
    <troubleSolution>
      <cause>
        Transient memory fault.
      </cause>
      <remedy>
        <responsibleParty>System administrator</responsibleParty>
        <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>
    <troubleSolution>
      <cause>
        Recurring memory fault indicates possible problem with the system memory board. Reseating the board may fix the problem.
      </cause>
      <remedy>
        <responsibleParty>Maintenance technician</responsibleParty>
        <steps conref="BoardReseat.dita#BoardReseatg/steps">...
          <cmd/>
          <cmd/>
        </steps>
      </remedy>
    </troubleSolution>
    <troubleSolution>
      <cause>
        Reseating the system memory board did not fix the problem. Replacing the board may fix the problem.
      </cause>
      <remedy>
        <responsibleParty>Maintenance technician</responsibleParty>
      </remedy>
    </troubleSolution>
  </troublebody>
</troubleshooting>
```
<steps conref="BoardReplace.dita#BoardReplace/steps">
  <step>
    <cmd/>
  </step>
</steps>
</remedy>
</troubleSolution>
</troubleshooting>

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below), Architectural attribute group (see page 654), and outputclass (see page 0).

@id (REQUIRED)

An anchor point. This ID is usually required as part of the @href or @conref syntax when cross referencing or reusing content within the topic; it also enables <topicref> elements in DITA maps to optionally reference a specific topic within a DITA document. This attribute is defined with the XML Data Type ID.

4.10.4.2 <troublebody>

The <troublebody> element is a container for the main content of the troubleshooting topic.

Inheritance

- topic/body troubleshooting/troublebody

Example

See troubleshooting (see page 444).

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.10.4.3 <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. This section should add to or clarify information that is in the title or short description of the troubleshooting topic. This information should be brief.

Comment by robander, 14 April 2014

Nancy's comment during the phase 1 review:

Why should this piece of information be brief? If the condition is complex, it might not be possible or advisable to make it brief.

Leaving it to the technical committee to change, if needed, but wonder if it might be appropriate to change the last sentence to "This information should be brief; if it is not possible to make the information brief, this may be a sign that a full troubleshooting topic is needed."

Inheritance

- topic/section troubleshooting/condition
Example
See troubleshooting (see page 444).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and spectitle (see page 0).

4.10.4.4 <troubleSolution>
The <troubleSolution> element is a container element for cause and remedy information. 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.

Inheritance
- topic/section troubleshooting/troubleSolution

Example
See troubleshooting (see page 444).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.10.4.5 <cause>
The <cause> element describes a potential source of the problem that is described in the <condition> element, the topic <title> element, or the <shortdesc> element. This information should be brief.

Comment by robander, 14 April 2014
Nancy’s comment during the phase 1 review:

Why should this piece of information be brief? If the condition is complex, it might not be possible or advisable to make it brief.

Leaving it to the technical committee to change, if needed, but wonder if it might be appropriate to change the last sentence to "This information should be brief; if it is not possible to make the information brief, this may be a sign that a full troubleshooting topic is needed."

Inheritance
- topic/section troubleshooting/cause

Example
See troubleshooting (see page 444).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and spectitle (see page 0).
4.10.4.6 <remedy>

The `<remedy>` element contains steps that are a potential solution for the problem described in the `<condition>` element, the topic `<title>` element, or the `<shortdesc>` element. Typically, a `<remedy>` is preceded by a `<cause>`.

**Inheritance**
- topic/section troubleshooting/remedy

**Example**
See `troubleshooting` (see page 444).

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *spectitle* (see page 0).

4.10.4.7 <responsibleParty>

Use the `<responsibleParty>` element to identify the party who is responsible for performing a remedy procedure.

**Inheritance**
- topic/p troubleshooting/responsibleParty

**Example**
See `troubleshooting` (see page 444).

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.10.5 Glossary elements

Glossary elements include those elements designed to specify terms and their definitions, as well as elements that are designed to group, reference, or otherwise make use of information in the glossentry topic.

4.10.5.1 Glossentry elements

Use the glossentry topic type to define glossary terms. Each glossentry topic should define a single sense of a term.

4.10.5.1.1 <glossentry>

The `<glossentry>` element defines a single sense of a glossary term. Glossary entries for different term senses can be reused independently of one another. DITA 1.2 adds several elements to the `<glossentry>` topic type, allowing it to specify additional information about a term (beyond just the definition).

The recommended (but not required) book processing is to sort and group glossary entries based on the localized term so a back-of-the-book glossary can contain a collated list of terms with the definitions of the individual senses of each term indented under the term. The glossary can have a different organization in different languages depending on the translation of the terms. One possible implementation of a glossary in online processing is to associate a hotspot for mentions of terms in `<term>` elements and display the definition on hover or click.
Inheritance
- topic/topic concept/concept glossentry/glossentry

Example

```xml
<glossentry id="usbfd">
  <glossterm>USB flash drive</glossterm>
  <glossdef>A small portable drive.</glossdef>
  <glossBody>
    <glossPartOfSpeech value="noun"/>
    <glossUsage>Do not provide in upper case (as in "USB Flash Drive") because that suggests a trademark.</glossUsage>
    <glossAlt>
      <glossAcronym>UFD</glossAcronym>
      <glossUsage>Explain the acronym on first occurrence.</glossUsage>
    </glossAlt>
    <glossAlt id="memoryStick">
      <glossSynonym>memory stick</glossSynonym>
      <glossUsage>This is a colloquial term.</glossUsage>
    </glossAlt>
    <glossAlt>
      <glossAbbreviation>stick</glossAbbreviation>
      <glossStatus value="prohibited"/>
      <glossUsage>This is too colloquial.</glossUsage>
      <glossAlternateFor href="#usbfd/memoryStick"/>
    </glossAlt>
    <glossAlt>
      <glossAbbreviation>flash</glossAbbreviation>
      <glossStatus value="prohibited"/>
      <glossUsage>This short form is ambiguous.</glossUsage>
    </glossAlt>
  </glossBody>
</glossentry>
```

Figure 103: The glossary term "USB flash drive" with additional information

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below), Architectural attribute group (see page 654), and outputclass (see page 0).

@id (REQUIRED)

An anchor point. This ID is usually required as part of the @href or @conref syntax when cross referencing or reusing content within the topic; it also enables <topicref> elements in DITA maps to optionally reference a specific topic within a DITA document. This attribute is defined with the XML Data Type ID.

4.10.5.1.2 <glossterm>

The <glossterm> element specifies the preferred term associated with a definition of a sense. If the same term has multiple senses, create a separate <glossentry> topic for each sense.

Inheritance
- topic/title concept/title glossentry/glossterm

Example

See the example in <glossentry>.

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (without the Metadata attribute group), @base from the Metadata attribute group (see page 651), and outputclass (see page 0).
4.10.5.1.3 <glossdef>

The <glossdef> element specifies the definition of one sense of a term. If a term has multiple senses, create a separate <glossentry> topic to define each sense.

**Inheritance**
- topic/abstract concept/abstract glossentry/glossdef

**Example**

See the example in <glossentry>.

**Attributes**

The following attributes are available on this element: Universal attribute group (see page 650) (without the Metadata attribute group), @base from the Metadata attribute group (see page 651), and outputclass (see page 0).

4.10.5.1.4 <glossAbbreviation>

The <glossAbbreviation> element provides an abbreviated form of the term contained in a <glossterm> element.

**Inheritance**
- topic/title concept/title glossentry/glossAbbreviation

**Example**

See the example in <glossentry>.

**Attributes**

The following attributes are available on this element: Universal attribute group (see page 650) (without the Metadata attribute group), @base from the Metadata attribute group (see page 651), and outputclass (see page 0).

4.10.5.1.5 <glossAcronym>

The <glossAcronym> element defines an acronym as an alternate form for the term defined in the <glossterm> element.

This element may be used together 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> for information on how the two elements interact.

**Note:** Several issues arise when acronyms are translated into other languages. For example, an acronym in one language may not have an equivalent in another language. When acronyms are first displayed, some languages will display the expanded form first followed by the acronym in parenthesis, while other languages do the reverse. For some acronyms, a translation may need to render both the original and the translated version of the acronym. For all of these reasons, DITA allows an author or translator to control what is presented to a reader by using the <glossSurfaceForm> element, which will often accompany the <glossAcronym>. The <abbreviated-form> topic contains information on how the <glossSurfaceForm> and <glossAcronym> elements affect references to the primary term.

**Inheritance**
- topic/title concept/title glossentry/glossAcronym
Example

See the example in `<glossentry>`.

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) (without the Metadata attribute group), `@base` from the *Metadata attribute group* (see page 651), and *outputclass* (see page 0).

4.10.5.1.6 `<glossAlt>`

The `<glossAlt>` element contains a variant term for the preferred term. 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 may be many ways to refer to a term; each variant is placed in its own `<glossAlt>` element. The `<glossUsage>` element may be used within `<glossAlt>` to indicate when use of the alternate term is appropriate.

>Note: Any list of alternative terms is, of course, specific to the language, so translation of a `<glossentry>` topic may result in empty elements within a `<glossAlt>` container.

Inheritance

- topic/section concept/section glossentry/glossAlt

Example

See the example in `<glossentry>`.

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.10.5.1.7 `<glossAlternateFor>`

The `<glossAlternateFor>` element indicates when a variant term has a relationship to another variant term as well as to the preferred term.

The `<glossAlternateFor>` element is available inside the `<glossAlt>` element, which is a container that provides a variant for the primary `<glossentry>` term. In some cases, the variant may also be an alternate for another term. In the example below, the abbreviation "stick" is a variant of the primary term (USB flash drive). The `<glossAlternateFor>` element indicates that "stick" is also a variant of the synonym "memory stick".

Inheritance

- topic/xref concept/xref glossentry/glossAlternateFor

Example

See the example in `<glossentry>`.

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661) (with a narrowed definition of `@href`, given below), `@keyref`, and *outputclass* (see page 0).

`@href`
References a term for which the current variant is an alternate (in addition to the primary term of this <glossentry> topic). The reference will often be to another <glossAlt> element within the same <glossentry> topic, indicating that the current variant is an alternate for both the primary term and the referenced alternate term. See The href attribute (see page 663) for details on syntax. This attribute is defined with the XML Data Type CDATA.

4.10.5.1.8 <glossBody>

The <glossBody> element is used to provide details about a glossary term (such as part of speech or additional forms of the term).

Inheritance
- topic/body concept/conbody glossentry/glossBody

Example
See the example in <glossentry>.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) (without the Metadata attribute group), @base from the Metadata attribute group (see page 651), and outputclass (see page 0 ).

4.10.5.1.9 <glossPartOfSpeech>

The <glossPartOfSpeech> element identifies the part of speech for the preferred and alternate terms. Alternate terms must have the same part of speech as the preferred term because all terms in the <glossentry> topic designate the same subject. If the part of speech isn't specified, the default is a noun for the standard enumeration.

Note: The standard enumeration is extensible or replaceable. If validation is required, the enumeration should be validated by means of the DITA Controlled Values mechanism or through processing rather than validated as an XML enumeration.

This should this include a link to the controlled values information in the architectural spec.

Inheritance
- topic/data concept/data glossentry/glossPartOfSpeech

Example
See the example in <glossentry>.

Attributes
The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0 ).

4.10.5.1.10 <glossProperty>

The <glossProperty> element is an extension point which allows additional details about the preferred term or its subject.

Inheritance
- topic/data concept/data glossentry/glossProperty
Example

<glossentry id="algorithm" xml:lang="es-es">
  <glossterm>El algoritmo</glossterm>
  <glossdef>Un algoritmo define un método de calcular un resultado.</glossdef>
  <glossBody>
    <glossPartOfSpeech value="noun"/>
    <glossProperty name="gender" value="masculine"/>
  </glossBody>
</glossentry>

Attributes

The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.5.1.11 <glossScopeNote>

The <glossScopeNote> element contains a clarification of the subject designated by the <glossterm>, such as examples of included or excluded companies or products. For instance, a scope note for "Linux" might explain that the term doesn't apply to UNIX products and may give some examples of Linux products that are included as well as UNIX products that are excluded.

Inheritance

- topic/note concept/note glossentry/glossScopeNote

Example

<glossentry id="linuxOS" xml:lang="en-us">
  <glossterm>Linux Operating System</glossterm>
  <glossdef>An operating system based on the kernel created by Linus Torvalds.</glossdef>
  <glossBody>
    <glossPartOfSpeech value="noun"/>
    <glossScopeNote>Doesn't apply to UNIX products that bundle other kernels. Also, doesn't apply to the Linux Open Source Project that work on Linux distributions but, instead, only to the distributions themselves. Examples include RedHat, SuSE, and Ubuntu.</glossScopeNote>
  </glossBody>
</glossentry>

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), spectitle (see page 0), and the attributes defined below.

@type

Defines the type of a note. For example, if the note is a tip, the word Tip may be used to draw the reader's attention to it. Note that this differs from the @type attribute on many other DITA elements. See The type attribute (see page 674) for detailed information on supported values and processing implications. Available values are note, tip, fastpath, restriction, important, remember, attention, caution, notice, danger, warning, trouble, other, and -dita-use-conref-target.

@othertype

Indicates an alternate note type, when the type is not available in the @type attribute value list. This value is used as the user-provided note title when the @type attribute value is set to "other". This attribute is defined with the XML Data Type CDATA.
4.10.5.1.12 <glossShortForm>

The <glossShortForm> element provides a shorter alternative to the primary term specified in the <glossterm> element.

Note: Any list of alternative terms is, of course, specific to the language, so translation of a <glossentry> topic may result in an empty <glossShortForm> element if there is no equivalent in the target language.

Inheritance
- topic/title concept/title glossentry/glossShortForm

Example

```
<glossentry id="www">
  <glossterm>World Wide Web</glossterm>
  <glossdef>A collection of documents available through the Internet.</glossdef>
  <glossBody>
    <glossPartOfSpeech value="noun"/>
    <glossAlt>
      <glossShortForm>the Web</glossShortForm>
    </glossAlt>
  </glossBody>
</glossentry>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) (without the Metadata attribute group), @base from the Metadata attribute group (see page 651), and outputclass (see page 0).

4.10.5.1.13 <glossStatus>

The <glossStatus> element identifies the usage status of a preferred or alternate term. If the status isn't specified, the <glossterm> provides a preferred term and an alternate term provides an allowed term.

Note: The values provided by the <glossStatus> element are extensible using DITA's Controlled Values mechanism. If validation is required, they should be validated using that mechanism rather than by encoding values directly within the XML declaration.

Inheritance
- topic/data concept/data glossentry/glossStatus

Example

See the example in <glossentry>.

Attributes
The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).
4.10.5.1.14 <glossSurfaceForm>

The <glossSurfaceForm> element specifies an unambiguous presentation of the <glossterm> that may combine multiple forms. The surface form is suitable to introduce the term in new contexts.

The <glossSurfaceForm> element is most often used for terms that also specify the <glossAcronym> element. In that case it contains the term in a manner that introduces both the term and the acronym, so that later references to the term may be replaced with the acronym alone. See the <abbreviated-form> element for a full description of how the surface form is used together with acronyms.

**Inheritance**
- topic/p concept/p glossentry/glossSurfaceForm

**Example**

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

This <glossentry> topic defines "Anti-lock Braking System (ABS)" provides an unambiguous way to render the primary term "Anti-lock Braking System". The <abbreviated-form> explains how the <glossSurfaceForm> element affects references to this topic.

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.10.5.1.15 <glossSymbol>

The <glossSymbol> element identifies a standard image associated with the subject of the <glossterm>.

**Inheritance**
- topic/image concept/image glossentry/glossSymbol

**Example**

```xml
<glossentry id="atlanticpuffin">
  <glossterm>Atlantic Puffin</glossterm>
  <glossdef>A sea bird that lives in the atlantic</glossterm>
  <glossSymbol href="puffinicon.jpg" scope="local">
    <alt>Atlantic puffin icon</alt>
  </glossSymbol>
</glossentry>
```

**Attributes**

**Comment by robander**
Note that this is an exact copy of the attribute list for <image>, EXCEPT that this element dropped the deprecated @alt and makes @href required, which makes it harder to reuse that full list. It is an exact copy of hazardsymbol.
The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), *@keyref*, and the attributes defined below.

**@href (REQUIRED)**

Provides a reference to the image. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. This attribute is defined with the XML Data Type CDATA.

**@scope**

The @scope attribute identifies the closeness of the relationship between the current document and the target resource. Allowable values are local, peer, external, and *-dita-use-conref-target*. See *The scope attribute* (see page 677) for more information on values.

**@height**

Indicates the vertical dimension for the resulting image display. If necessary, the image is scaled to the specified size. The value of this attribute is a real number (expressed in decimal notation) optionally followed by a unit of measure from the set of pc, pt, px, in, cm, mm, em (picas, points, pixels, inches, centimeters, millimeters, and ems respectively). The default unit is px (pixels). Possible values include: "5", "5in", and "10.5cm". If a height value is specified and no width value is specified, the width will be scaled by the same factor as the height. If both a height value and width value are specified, some implementations may not be able to scale the two directions by a different factor and may therefore ignore one of the two values. This attribute is defined with the XML Data Type NMTOKEN.

**@width**

Indicates the horizontal dimension for the resulting image display. If necessary, the image is scaled to the specified size. The value of this attribute is a real number (expressed in decimal notation) optionally followed by a unit of measure from the set of pc, pt, px, in, cm, mm, em (picas, points, pixels, inches, centimeters, millimeters, and ems respectively). The default unit is px (pixels). Possible values include: "5", "5in", and "10.5cm". If a width value is specified and no height value is specified, the height will be scaled by the same factor as the width. If both a height value and width value are specified, some implementations may not be able to scale the two directions by a different factor and may therefore ignore one of the two values. This attribute is defined with the XML Data Type NMTOKEN.

**@align**

Controls the horizontal alignment of an image when placement is specified as "break". Common values include left, right, and center. This attribute is defined with the XML Data Type CDATA.

**@scale**

Specifies a percentage 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 this image's @height or @width attribute (or both), the @scale attribute is ignored. This attribute is defined with the XML Data Type NMTOKEN.

It is an error if the value of this attribute is not an unsigned integer. In this case, the implementation may (but need not) give an error message and may (but need not) recover by ignoring this attribute.

**@scalefit**

Allow an image to be scaled up or down to fit within available space. Allowable values are yes, no, and *-dita-use-conref-target*. If, for a given image, any one of @height, @width, or @scale is specified, those attributes determine the graphic size, and any setting of @scalefit is ignored. If none of those attributes are specified and scalefit="yes", then the image is scaled (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 the prevailing column (or table cell) width—that is, the width a paragraph of text would have if the graphic were a paragraph instead. 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.

**@placement**

Indicates whether an image should be displayed inline or separated from the surrounding text. The default is inline. Allowable values are: inline, break, or and *-dita-use-conref-target*. **Comment by robander**
@longdesc (DEPRECATED)

A reference to a textual description of the graphic or object. This attribute supports creating accessible content. See The href attribute (see page 663) for detailed information on supported values and processing implications. For examples of how this attribute is used in output, see this topic on long descriptions. NOTE: This attribute is deprecated in favor of the longdescref (see page 272) subelement to this element. This attribute is defined with the XML Data Type CDATA.

4.10.5.1.16 <glossSynonym>

The <glossSynonym> element provides a term that is a synonym of the primary value in the <glossterm> element.

Note: Any list of alternative terms is, of course, specific to the language, so translation of a <glossentry> topic may result in an empty <glossSynonym> element if there is no equivalent in the target language.

Inheritance
- topic/title concept/title glossentry/glossSynonym

Example
See the example in <glossentry>.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) (without the Metadata attribute group), @base from the Metadata attribute group (see page 651), and outputclass (see page 0).

4.10.5.1.17 <glossUsage>

The <glossUsage> element provides information about the correct use of a term, such as where or how it can be used.

Inheritance
- topic/note concept/note glossentry/glossUsage

Example
See the example in <glossentry>.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), spectitle (see page 0), and the attributes defined below.

@type
Defines the type of a note. For example, if the note is a tip, the word Tip may be used to draw the reader's attention to it. Note that this differs from the @type attribute on many other DITA elements. See The type attribute (see page 674) for detailed information on supported values and processing implications. Available values are note, tip, fastpath, restriction, important, remember, attention, caution, notice, danger, warning, trouble, other, and -dita-use-conref-target.
Indicates an alternate note type, when the type is not available in the @type attribute value list. This value is used as the user-provided note title when the @type attribute value is set to "other". This attribute is defined with the XML Data Type CDATA.

4.10.5.2 Glossary related elements

Elements related to the glossary specialization are not defined as part of the glossary topic type, but are often used in conjunction with those elements.

4.10.5.2.1 <abbreviated-form>

The <abbreviated-form> element represents a reference to a term that may appear in an abbreviated form (often an acronym). The long and short forms of the term are generally defined in a <glossentry> topic. Processors should display the referenced term when rendering an <abbreviated-form> element.

Rendering the <abbreviated-form> element

The <abbreviated-form> element is intended to reference a <glossentry> topic that contains both a term and an abbreviated form of that term. The topic may also provide a surface form that differs from the original term. The full term or surface form should be rendered in introductory contexts where the term might be unfamiliar to a reader or in other contexts where a precise term is appropriate. In other contexts a process should substitute the abbreviated form of the term. Note that the definition of an introductory context will differ for every deliverable format.

For instance, a process composing a book deliverable may emit 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 may emit the surface form as a hover tooltip on every instance of the term.

Renderers should follow these rules in order to determine how to render an <abbreviated-form> element. As described, the definition of "introductory context" will vary based on the rendered format and processor.

1. If the referenced topic is not a <glossentry> topic (or a specialization of <glossentry>), the title of the topic SHOULD be displayed.
2. Otherwise, if <abbreviated-form> is located in an introductory context, and the referenced topic contains a non-empty <glossSurfaceForm> element, processors SHOULD render the contents of the <glossSurfaceForm> element from the referenced <glossentry> topic.
3. Otherwise, if <abbreviated-form> is located in an introductory context, processors SHOULD render the contents of the <glossTerm> element from the referenced <glossentry> topic.
4. Otherwise (in non-introductory contexts), if the referenced <glossentry> topic contains a non-empty <glossAcronym> element, processors SHOULD render the abbreviated form of the term by displaying the contents of the <glossAcronym> element from the referenced <glossentry> topic.
5. Otherwise, processors SHOULD render the contents of the <glossTerm> element from the referenced <glossentry> topic.

For instance, if an <abbreviated-form> element with the attribute keyref="abs" provided the first occurrence of the ABS term within a book, the sentence could be rendered as follows:

"The Anti-lock Brake System (ABS) will prevent the car from skidding in adverse weather conditions."

If the ABS term had appeared previously within the book, the same sentence could instead be rendered as follows:

"The ABS will prevent the car from skidding in adverse weather conditions."
Inheritance
+ topic/term abbrev-d/abbreviated-form

Example
The term and acronym may be defined as follows, in a <glossentry> topic. Note that the @id attribute of the
doc topic does not need to match the term or acronym.

Note that there are three important elements for the purposes of rendering the <abbreviated-form> element.

- The <glossSurfaceForm> element defines the term as it should be displayed in an introductory context.
- The <glossAcronym> element defines the acronym associated with this term.
- The <glossterm> element provides a fallback version of the term, which will be displayed in situations
  where the preferred representation is unavailable.

The <glossentry> topic will be added to a map in the following manner. Again, the key (in this case "abs")
does not need to match the term or acronym value.

An author that wishes to reference this topic may do so using the <abbreviated-form> element. The @keyref
attribute should reference the value defined on the @keys attribute above.

When rendered, the introductory usage of "abs" will display the surface form of the referenced term, while the
later uses will display the acronym, as demonstrated here.

An Anti-lock Braking System (ABS) helps a driver to stop. For this reason many find an ABS useful.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see
page 0 ), and @keyref.

4.10.5.2.2 <glossref>

The <glossref> element is a convenience element in maps for creating a reference to a glossary topic. It has a
required @keys attribute, which forces the author to create a key by which inline terms can reference their
definition. For example, when <glossentry> topics are used to define acronyms, this reminds authors to create
a key which <abbreviated-form> elements can use to reference the short and expanded versions of that
acronym.

Note that the key value does not need to match the target term or acronym. In fact, using a more qualified value
for the @keyref will reduce conflicts in situations where the same term or acronym may resolve in many ways.
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.

**Inheritance**

+ map/topicref glossref-d/glossref

**Example**

```xml
<map>
    <!-- ... -->
    <topicref href="car-maintenance.dita"/>
    <!-- ... -->
    <glossref keys="cars.abs" href="antiLockBrake.dita"/>
    <!-- ... key declarations for other referenced acronyms ... -->
</map>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Topicref element attributes group* (see page 663), *Link relationship attribute group* (see page 661) (with a narrowed definition of @href, given below), *outputclass* (see page 96), @keyref, and the attributes defined below. This element also uses @collection-type, @locktitle, @chunk, and @search from *Attributes common to many map elements* (see page 655); this element also uses narrowed definitions of @linking, @toc, and @print from *Attributes common to many map elements* (see page 655), given below.

@href

A pointer to a glossary definition, typically a <glossentry> topic. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

@keys (REQUIRED)

Associates one or more space-delimited keys with the target of the glossary reference. See *The keys attribute* (see page 665) for information on using the attribute.

@toc

Specifies whether a topic appears in the table of contents (TOC). If the value is not specified locally, but is specified on an ancestor, the value will cascade from the closest ancestor. On this element the default value for @toc is "no". See *Attributes common to many map elements* (see page 655) for a complete definition of @toc.

@print

Specifies whether the topic should be included in a print-specific rendition, such as PDF. On this element the value defaults to "no". Allowable values are:

- **yes**: Include the topic in the print-oriented file.
- **no**: Do not include the topic in a print-oriented file.
- **printonly**: Only include the topic when rendering the DITA content in a print-oriented context; the topic should not be included in other contexts, such as when rendering as HTML.

@dita-use-conref-target

See *Using the -dita-use-conref-target value* (see page 666) for more information.

**Note:** Beginning with DITA 1.3, the @print attribute is deprecated. It is replaced with a conditional processing attribute: @deliveryTarget. See @deliveryTarget for more details.

@linking
Defines some specific linking characteristics of a glossary topic's current location in the map. On this element the value defaults to "none" in order to keep individual glossary entries from creating links based on their location in the map. Allowable values are:

**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 (see page 666) for more information.

### 4.10.5.2.3 <glossgroup>

The `<glossgroup>` is a specialized topic that may be used to contain multiple `<glossentry>` topics within a single collection.

**Inheritance**
- topic/topic concept/concept glossgroup/glossgroup

#### Example

```xml
<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>
```

**Attributes**

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below), Architectural attribute group (see page 654), and outputclass (see page 0).

#### @id (REQUIRED)

An anchor point. This ID is usually required as part of the @href or @conref syntax when cross referencing or reusing content within the topic; it also enables `<topicref>` elements in DITA maps to optionally reference a specific topic within a DITA document. This attribute is defined with the XML Data Type ID.

### 4.10.6 Bookmap elements

Elements in the bookmap 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.

#### 4.10.6.1 Bookmap content elements

The Bookmap specialization of ditamap supports standard book production for collections of DITA topics.

The OASIS document type for the bookmap specialization also includes substantial book metadata for describing authors, based on the eXtensible Name and Address Language, or xNAL.
4.10.6.1.1 <bookmap>

The `<bookmap>` element is a map specialization used to describe the relationships among a set of DITA topics intended to be configured as a traditional book. Bookmaps 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.

**Inheritance**
- map/map bookmark/bookmap

**Example**

```xml
<bookmap xml:lang="en-us">
  <booktitle>
    <booklibrary>Books about stuff</booklibrary>
    <mainbooktitle>A book about one thing</mainbooktitle>
  </booktitle>
  <bookmeta>
    <bookrights>
      <copyrfirst><year>1994</year></copyrfirst>
      <copyrlast><year>2006</year></copyrlast>
      <bookowner>OASIS</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="updatesToTheBook.dita"/>
    <booklists>
      <trademarklist href="listoftrademarks.dita"/>
      <indexlist/>
    </booklists>
  </backmatter>
</bookmap>
```

**Attributes**

**Comment by robander**

Discussed at TC Jan 14, 2014, should use conref-atts instead of single redefined @conref. Odd set of conrefs here is because we use the common set of attributes (references to groups and narrowed @id), plus @id and @anchorref from map, but we drop the unique-to-map @title.

The following attributes are available on this element: **Universal attribute group** (see page 650) (with a narrowed definition of @id, given below), **Attributes common to many map elements** (see page 655), **Architectural attribute group** (see page 654), **outputclass** (see page 0), and the attributes defined below. This element also uses @type, @scope, and @format from **Link relationship attribute group** (see page 661).

@id
Allows an ID to be specified for the map. Note that maps do not require IDs (unlike topics), and the map ID is not included in references to elements within a map. This attribute is defined with the XML Data Type ID.

@anchorref

Identifies a location within another map file where this map will be anchored at runtime. Resolution of the map is deferred until the final step in the delivery of any rendered content. For example, anchorref="map1.ditamap/a1" causes this map to be pulled into the location of the anchor point "a1" inside map1.ditamap when map1.ditamap is rendered for delivery. This attribute is defined with the XML Data Type CDATA.

4.10.6.1.2 <abbrevlist>

The <abbrevlist> element references a list of abbreviations. It indicates to the processing software that the author wants an abbreviation list generated at the particular location.

Inheritance

- map/topicref bookmap/abbrevlist

Example

<abbrevlist href="abbrev.dita"/>

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), Attributes common to many map elements (see page 655), @navtitle and @copy-to from Topicref element attributes group (see page 663), outputclass (see page 0 ), and @keyref.

@href

References a manual listing for the current element. See The href attribute (see page 663) for detailed information on supported values and processing implications. If no @href is specified, processors may choose to generate an appropriate listing for this element. All of the book listings operate in a similar manner; for example, <toc href="toc.dita"/> references a topic which contains a manual table of contents, while <toc/> indicates that a processor should generate the table of contents.

4.10.6.1.3 <amendments>

The <amendments> element references a list of amendments or updates to the book. It indicates to the processing software that the author wants an amendments list generated at the particular location.

Inheritance

- map/topicref bookmap/amendments

Example

See the example for bookmap (see page 461).

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), Attributes common to many map elements (see page 655), @navtitle and @copy-to from Topicref element attributes group (see page 663), outputclass (see page 0 ), and @keyref.

@href
A pointer to the resource represented by the `<topicref>`. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the `@format` attribute to identify the kind of resource being referenced.

### 4.10.6.1.4 `<appendices>`

The `<appendices>` element is an optional wrapper for `<appendix>` elements within a bookmap.

**Inheritance**

- map/topicref bookmap/appendices

**Example**

```xml
<appendices toc="yes" print="no">
  <topicmeta>
    <navtitle>Appendices</navtitle>
  </topicmeta>
  <appendix href="return-codes.dita"/>
  <appendix href="messages.dita"/>
  <appendix href="extra-info.dita"/>
</appendices>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661) (with a narrowed definition of `@href`, given below), *Attributes common to many map elements* (see page 655), `@navtitle` and `@copy-to` from *Topicref element attributes group* (see page 663), `outputclass` (see page 0), and `@keyref`.

`@href`:

A pointer to the resource represented by the `<topicref>`. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the `@format` attribute to identify the kind of resource being referenced.

### 4.10.6.1.5 `<appendix>`

The `<appendix>` element references a topic as a appendix within a book.

**Inheritance**

- map/topicref bookmap/appendix

**Example**

Appendix topics that include subtopics:

```xml
<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>
```

Appendix that references a ditamap of content:

```xml
<appendix href="intro.ditamap" format="ditamap"/>
```
Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), Attributes common to many map elements (see page 655), @navtitle and @copy-to from Topicref element attributes group (see page 663), outputclass (see page 0 ), and @keyref.

@href

A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

4.10.6.1.6 <backmatter>

The <backmatter> element contains the material that follows the main body of a document and any appendices. It may include items such as a colophon, legal notices, and various types of book lists such as a glossary or an index.

Inheritance

- map/topicref bookmap/backmatter

Example

See the example in bookmap (see page 461).

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Attributes common to many map elements (see page 655), outputclass (see page 0 ), @keyref, and @query from Topicref element attributes group (see page 663). This element also uses @scope, @format, and @type from Link relationship attribute group (see page 661).

4.10.6.1.7 <bibliolist>

The <bibliolist> element references a topic containing a list of bibliographic entries within the book. It indicates to the processing software that the author wants a bibliography, containing links to related books, articles, published papers, or other types of material, generated at the particular location. If no @href attribute is specified on the <bibliolist> element, an external processor may generate a list of bibliographic entries at this location.

Inheritance

- map/topicref bookmap/bibliolist

Example

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

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661) (with a narrowed definition of @href, given below), *Attributes common to many map elements* (see page 655), @navtitle and @copy-to from *Topicref element attributes group* (see page 663), *outputclass* (see page 0), and @keyref.

@href

References a manual listing for the current element. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. If no @href is specified, processors may choose to generate an appropriate listing for this element. All of the book listings operate in a similar manner; for example, `<toc href="toc.dita"/>` references a topic which contains a manual table of contents, while `<toc/>` indicates that a processor should generate the table of contents.

### 4.10.6.1.8 <bookabstract>

The `<bookabstract>` element references a topic used within a bookmap as a brief summary of book content, generally output as part of the book's front matter. It is used to help the reader quickly evaluate the book's purpose.

**Inheritance**

- map/topicref bookmap/bookabstract

**Example**

See the example in *bookmap* (see page 461).

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661) (with a narrowed definition of @href, given below), *Attributes common to many map elements* (see page 655), @navtitle and @copy-to from *Topicref element attributes group* (see page 663), *outputclass* (see page 0), and @keyref.

@href

A pointer to the resource represented by the `<topicref>`. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

### 4.10.6.1.9 <booklibrary>

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

**Inheritance**

- topic/ph bookmap/booklibrary

**Example**

See the example in *bookmap* (see page 461).

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and @keyref.
The `<booklist>` element is a general purpose element, designed for use in specializations, 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, that element should be used instead.

**Inheritance**

- map/topicref bookmap/booklist

**Example**

In this case the `<booklist>` element references a topic that contains a list of authors of topics in this document.

```xml
<booklists>
  <toc/>
  <tablelist/>
  <booklist href="authors.dita" navtitle="List of authors"/>
</booklists>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661) (with a narrowed definition of @href, given below), *Attributes common to many map elements* (see page 655), @navtitle and @copy-to from *Topicref element attributes group* (see page 663), *outputclass* (see page 0 ), and @keyref.

@href

References a manual listing for the current element. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. If no @href is specified, processors may choose to generate an appropriate listing for this element. All of the book listings operate in a similar manner; for example, `<toc href="toc.dita"/>` references a topic which contains a manual table of contents, while `<toc/>` indicates that a processor should generate the table of contents.

The `<booklists>` element references lists of various kinds within the book. For example, it can be used within `<frontmatter>` to reference a `<toc>`, `<tablelist>`, and `<figurelist>`, or within `<backmatter>` to reference a `<glossarylist>`, `<indexlist>`, and `<abbrevlist>`. It indicates to the processing software that the author wants the lists generated at the `<booklists>` location.

**Inheritance**

- map/topicref bookmap/booklists

**Example**

See the example in *bookmark* (see page 461).

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Attributes common to many map elements* (see page 655) (with a narrowed definition of @chunk given below), *Topicref element attributes group* (see page 663), and @keyref. This element also uses @scope, @format, and @type from *Link relationship attribute group* (see page 661).
4.10.6.1.12 <booktitle>

The <booktitle> element contains the title information for a book, including the library title, main title, subtitle, and other titles (as required).

*Inheritance*

- topic/title bookmap/booktitle

*Example*

See the example in *bookmap* (see page 461).

*Attributes*

The following attributes are available on this element: *Universal attribute group* (see page 650) (without the Metadata attribute group) and *outputclass* (see page 0).

4.10.6.1.13 <booktitlealt>

The <booktitlealt> element contains the alternative title, subtitle, or short title for a book. It may be specialized into a specific element for those or other purposes.

*Inheritance*

- topic/ph bookmap/booktitlealt

*Example*

```xml
<bookmap>
    <booktitle>
        <mainbooktitle>This is my big and fancy book</mainbooktitle>
        <booktitlealt>Shorter title</booktitlealt>
    </booktitle>
    <!-- ... -->
</bookmap>
```

*Attributes*

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *@keyref*.

4.10.6.1.14 <chapter>

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

*Inheritance*

- map/topicref bookmap/chapter

*Example*

Chapter topics that include subtopics:

```xml
<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>
```
Chapter that references a ditamap of content:

```xml
<chapter href="intro.ditamap" format="ditamap"/>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661) (with a narrowed definition of @href, given below), *Attributes common to many map elements* (see page 655), @navtitle and @copy-to from *Topicref element attributes group* (see page 663), *outputclass* (see page 0), and @keyref.

@href

A pointer to the resource represented by the `<topicref>`. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

### 4.10.6.1.15 `<colophon>`

The `<colophon>` element references a topic describing how this document was created. In publishing, a colophon describes details of the production of a book. This information generally includes the typefaces used, and often the names of their designers; the paper, ink and details of the binding materials and methods may also receive mention. In the case of technical books, a colophon may specify the software used to prepare the text and diagrams for publication. The `<colophon>` may appear in the `<frontmatter> or `<backmatter>.

**Inheritance**

- map/topicref bookmap/colophon

**Example**

```xml
<bookmap>
  <title>Sample book</title>
  <!-- ... -->
  <backmatter>
    <colophon href="ProdNot.dita" navtitle="Production Notes"/>
  </backmatter>
</bookmap>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661) (with a narrowed definition of @href, given below), *Attributes common to many map elements* (see page 655), @navtitle and @copy-to from *Topicref element attributes group* (see page 663), *outputclass* (see page 0), and @keyref.

@href

A pointer to the resource represented by the `<topicref>`. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

### 4.10.6.1.16 `<dedication>`

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

**Inheritance**

- map/topicref bookmap/dedication
Example

```xml
<frontmatter>
  <dedication href="dtm.dita" navtitle="Dedicated to Mother"/>
</frontmatter>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), Attributes common to many map elements (see page 655), @navtitle and @copy-to from Topicref element attributes group (see page 663), outputclass (see page 0), and @keyref.

@href

A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

4.10.6.1.17 <draftintro>

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

Inheritance

- map/topicref bookmap/draftintro

Example

```xml
<frontmatter>
  <draftintro href="introducing.dita" navtitle="Introduction to this draft"/>
</frontmatter>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), Attributes common to many map elements (see page 655), @navtitle and @copy-to from Topicref element attributes group (see page 663), outputclass (see page 0), and @keyref.

@href

A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

4.10.6.1.18 <figurelist>

The <figurelist> element references a topic containing a list of figures in the book. It indicates to the processing software that the author wants a list of figures generated at the particular location. If no @href attribute is specified on the <figurelist> element, an external processor may generate a list of figures at this location.

Inheritance

- map/topicref bookmap/figurelist
Example
See the example in bookmap (see page 461).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), Attributes common to many map elements (see page 655), outputclass (see page 0), @copy-to from Topicref element attributes group (see page 663), @navtitle and @keyref. This element also uses @query from Link relationship attribute group (see page 661).

4.10.6.19 <frontmatter>

The <frontmatter> element contains the material that precedes the main body of a document. It may include items such as an abstract, a preface, and various types of book lists such as a <toc>, <tablelist>, or <figurelist>.

Inheritance
- map/topicref bookmark/frontmatter

Example
See the example in bookmap (see page 461).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Attributes common to many map elements (see page 655), outputclass (see page 0), @keyref, and @query from Topicref element attributes group (see page 663). This element also uses @scope, @format, and @type from Link relationship attribute group (see page 661).

4.10.6.20 <glossarylist>

The <glossarylist> element references a list of glossary entries within the book. It indicates to the processing software that the author wants a glossary list generated at the particular location.

Inheritance
- map/topicref bookmark/glossarylist

Example
See backmatter (see page 464).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), Attributes common to many map elements (see page 655), @navtitle and @copy-to from Topicref element attributes group (see page 663), outputclass (see page 0), and @keyref.
References a manual listing for the current element. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. If no @href is specified, processors may choose to generate an appropriate listing for this element. All of the book listings operate in a similar manner; for example, `<toc href="toc.dita"/>` references a topic which contains a manual table of contents, while `<toc/>` indicates that a processor should generate the table of contents.

### 4.10.6.1.21 <indexlist>

The `<indexlist>` element indicates to the processing software that the author wants an index at the particular location. If no @href attribute is specified on the `<indexlist>` element, an external processor may generate an index at this location. If the @href attribute references a topic or map, that topic or map should contain a manually created index.

**Inheritance**
- map/topicref bookmap/indexlist

**Example**

See the example in *bookmap* (see page 461).

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661) (with a narrowed definition of @href, given below), *Attributes common to many map elements* (see page 655), `@navtitle` and `@copy-to` from *Topicref element attributes group* (see page 663), `outputclass` (see page 0), and `@keyref`.

@href

References a manual listing for the current element. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. If no @href is specified, processors may choose to generate an appropriate listing for this element. All of the book listings operate in a similar manner; for example, `<toc href="toc.dita"/>` references a topic which contains a manual table of contents, while `<toc/>` indicates that a processor should generate the table of contents.

### 4.10.6.1.22 <mainbooktitle>

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

**Inheritance**
- topic/ph bookmap/mainbooktitle

**Example**

See the example in *bookmap* (see page 461).

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), `outputclass` (see page 0), and `@keyref`.

### 4.10.6.1.23 <notices>

The `<notices>` element references a topic containing special notice information, for example, legal notices about supplementary copyrights and trademarks associated with the book.

**Inheritance**
- map/topicref bookmap/notices
Example

This example references a notices topic that contains legal content.

```xml
<backmatter>
  <notices href="notices.dita" navtitle="Legal notices"/>
  <booklists>
    <!-- Index, glossary, or other lists -->
  </booklists>
</backmatter>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), Attributes common to many map elements (see page 655), @navtitle and @copy-to from Topicref element attributes group (see page 663), outputclass (see page 0 ), and @keyref.

@href

A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

4.10.6.1.24 <part>

The <part> element references a part topic or a map that references part topics for the book. 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.

Inheritance

- map/topicref bookmap/part

Example

Part topics that include chapters and subtopics:

```xml
<part href="guide.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="ref.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>
```

Parts that reference ditamaps of content:

```xml
<part href="intro.ditamap" format="ditamap"/>
<part href="guide.ditamap" format="ditamap"/>
```
Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), Attributes common to many map elements (see page 655), @navtitle and @copy-to from Topicref element attributes group (see page 663), outputclass (see page 0), and @keyref.

@href

A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

4.10.6.1.25 <preface>

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

Inheritance

- map/topicref bookmap/preface

Example

See the example in bookmap (see page 461).

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), Attributes common to many map elements (see page 655), @navtitle and @copy-to from Topicref element attributes group (see page 663), outputclass (see page 0), and @keyref.

@href

A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

4.10.6.1.26 <tablelist>

The <tablelist> element references a topic that contains a list of tables within the book. It indicates to the processing software that the author wants a list of tables generated at the particular location. If no @href attribute is specified on the <tablelist> element, an external processor may generate a list of tables at this location.

Inheritance

- map/topicref bookmap/tablelist

Example

See the example in bookmap (see page 461).

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), Attributes common to many map elements (see page 655), @navtitle and @copy-to from Topicref element attributes group (see page 663), outputclass (see page 0), and @keyref.

@href

A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.
map elements (see page 655), @navtitle and @copy-to from Topicref element attributes group (see page 663), outputclass (see page 0 ), and @keyref.

@href
References a manual listing for the current element. See The href attribute (see page 663) for detailed information on supported values and processing implications. If no @href is specified, processors may choose to generate an appropriate listing for this element. All of the book listings operate in a similar manner; for example, <toc href="toc.dita"> references a topic which contains a manual table of contents, while <toc/> indicates that a processor should generate the table of contents.

4.10.6.1.27 <toc>

The <toc> element indicates to the processing software that the author wants a table of contents generated at the particular location. If no @href attribute is specified on the <toc> element, an external processor may generate table of contents at this location. If the @href attribute references a topic or map, that topic or map should contain a manually created table of contents.

Inheritance
- map/topicref bookmap/toc

Example
See the example in bookmap (see page 461).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), Attributes common to many map elements (see page 655), @navtitle and @copy-to from Topicref element attributes group (see page 663), outputclass (see page 0 ), and @keyref.

@href
References a manual listing for the current element. See The href attribute (see page 663) for detailed information on supported values and processing implications. If no @href is specified, processors may choose to generate an appropriate listing for this element. All of the book listings operate in a similar manner; for example, <toc href="toc.dita"> references a topic which contains a manual table of contents, while <toc/> indicates that a processor should generate the table of contents.

4.10.6.1.28 <trademarklist>

The <trademarklist> element references a topic that contains a list of trademarks within the book. It indicates to the processing software that the author wants a list of trademarks generated at the particular location. If no @href attribute is specified on the <trademarklist> element, an external processor may generate a list of trademarks at this location.

Inheritance
- map/topicref bookmap/trademarklist

Example
See the example in bookmap (see page 461).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), Attributes common to many
map elements (see page 655), @navtitle and @copy-to from Topicref element attributes group (see page 663), outputclass (see page 0), and @keyref.

@href
References a manual listing for the current element. See The href attribute (see page 663) for detailed information on supported values and processing implications. If no @href is specified, processors may choose to generate an appropriate listing for this element. All of the book listings operate in a similar manner; for example, <toc href="toc.dita"/> references a topic which contains a manual table of contents, while <toc/> indicates that a processor should generate the table of contents.

4.10.6.2 Bookmap metadata elements

The Bookmap specialization of ditamap supports standard book production for collections of DITA topics. This section contains the metadata elements used by bookmap to store book-related metadata.

Note: The OASIS document type for the bookmap specialization includes the xNAL domain specialization for describing the author or authors of a document. All elements that appear in that domain (the <authorinformation> element and all of its descendants) are described in the xNAL domain section, rather than in this bookmap metadata section.

4.10.6.2.1 <bookmeta>

The <bookmeta> element contains information about the book that is not considered book content, such as copyright information, author information, and any classifications.

Inheritance
- map/topicmeta bookmap/bookmeta

Example

In this example:

- The <authorinformation> contains the content for a reader's comment form; the <maintainer> element references that author information to create the reader comment form.
- The <prodinfo> element contains the book's product information. This includes the product name, and the version, release, and modification information.
- The <bookid> element contains the book's form number and part number information. The <maintainer> element contains information about the current maintainer of the book; it may reference the <authorinformation> element to provide detailed information about an author that is also the current maintainer.
- The <bookrights> element contains the book's copyright information.

This example makes heavy use of <authorinformation>, which is a domain element that OASIS delivers only with the <bookmap> specialization. However, other DTD, XSD Schema, or RELAX NG implementations of <bookmap> may not include the domain with <authorinformation>; likewise, <authorinformation> may be included in other document types.
Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and the attribute defined below.

@lockmeta

Indicates whether the metadata information should be replaced by the metadata information in the referenced topic. Allowable values are:

yes

The information inside <topicmeta> should not be replaced with information from the topic.

no

The information inside <topicmeta> may be replaced with information from the topic.

-dita-use-conref-target

See Using the -dita-use-conref-target value (see page 666) for more information.

4.10.6.2.2 <approved>

The <approved> element contains information about when and by whom the book was approved during its publication history.

Inheritance

- topic/data bookmap/approved

Example

See the example in bookchangehistory (see page 477).

Attributes

The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).
4.10.6.2.3 <bookchangehistory>

The <bookchangehistory> element contains information about the history of the book's creation and publishing lifecycle, including who wrote, reviewed, edited, and tested the book. It also specifies when these events took place.

Inheritance
- topic/data bookmap/bookchangehistory

Example

```xml
<bookmeta>
  <bookchangehistory>
    <reviewed>
      <started><year>2007</year><month>10</month></started>
    </reviewed>
    <completed><year>2008</year><month>01</month></completed>
  </bookchangehistory>
  <edited>
    <person>Joe T. Editor</person>
    <completed><year>2008</year><month>03</month><day>15</day></completed>
  </edited>
  <tested>
    <organization>OASIS</organization>
    <completed><year>2008</year><month>04</month></completed>
  </tested>
  <approved>
    <organization>OASIS</organization>
    <completed><year>2008</year><month>05</month></completed>
  </approved>
  <bookevent type="Type of Major Event"/>
    <completed><year>2009</year></completed>
  </bookevent>
</bookchangehistory>
</bookmeta>
```

Attributes

The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.6.2.4 <bookevent>

The <bookevent> element indicates a general event in the publication history of a book. This is an appropriate element for specialization if the current set of specific book event types does not meet your needs. If an element already exists to describe a specific type of event, such as <reviewed>, <edited>, or <approved>, that element should be used instead.

Inheritance
- topic/data bookmap/bookevent

Example

See the example in bookchangehistory (see page 477).

Attributes

The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).
4.10.6.2.5 <bookeventtype>

The <bookeventtype> element indicates the specific nature of a <bookevent>, such as updated, indexed, or deprecated. The required @name attribute indicates the event's type.

Inheritance
- topic/data bookmap/bookeventtype

Example
See the example in bookchangehistory (see page 477).

Attributes
The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.6.2.6 <bookid>

The <bookid> element contains the publisher's identification information for the book, such as part number, edition number and ISBN number.

Inheritance
- topic/data bookmap/bookid

Example
See bookmeta (see page 475).

Attributes
The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.6.2.7 <booknumber>

The <booknumber> element contains the book's form number, such as SC21-1920.

Inheritance
- topic/data bookmap/booknumber

Example
In this example, "99F1234" is a part number assigned to this book by the publisher, while SC21-1234-00 is a number that identifies this book among all of the author's works.

```
<bookmeta>
  <bookid>
    <bookpartno>99F1234</bookpartno>
    <booknumber>SC21-1234-00</booknumber>
  </bookid>
</bookmeta>
```
Attributes
The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.6.2.8 <bookowner>

The <bookowner> element specifies the owner of the book's copyright.

Inheritance
- topic/data bookmap/bookowner

Example

```xml
<bookmeta>
  <bookrights>
    <copyrfirst><year>1994</year></copyrfirst>
    <copyrlast><year>2006</year></copyrlast>
    <bookowner>
      <organization>Example Corporation</organization>
    </bookowner>
  </bookrights>
</bookmeta>
```

Attributes
The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.6.2.9 <bookpartno>

The <bookpartno> element contains the book's part number, such as 99F1234. A publisher may use a number like this one to identify a book for tracking purposes.

Inheritance
- topic/data bookmap/bookpartno

Example

In this example, "99F1234" is a part number assigned to this book by the publisher, while SC21-1234-00 is a number that identifies this book among all of the author's works.

```xml
<bookmeta>
  <bookid>
    <bookpartno>99F1234</bookpartno>
    <booknumber>SC21-1234-00</booknumber>
  </bookid>
</bookmeta>
```

Attributes
The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).
4.10.6.2.10 <bookrestriction>

The `<bookrestriction>` element indicates whether the book is classified or restricted in some way. The `@value` attribute indicates any restrictions on the use of the material, such as declaring the information confidential or for licensed use only.

**Inheritance**
- topic/data bookmap/bookrestriction

**Example**

```xml
<bookrights>
  <copyrfirst><year>1994</year></copyrfirst>
  <copyrlast><year>2006</year></copyrlast>
  <bookowner><organization>Example Corporation</organization></bookowner>
  <bookrestriction value="unclassified"/>
</bookrights>
```

**Attributes**

The following attributes are available on this element: `Data element attributes group` (see page 659), `Link relationship attribute group` (see page 661), `Universal attribute group` (see page 650), and `outputclass` (see page 0).

4.10.6.2.11 <bookrights>

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

**Inheritance**
- topic/data bookmap/bookrights

**Example**

See `bookmeta` (see page 475).

**Attributes**

The following attributes are available on this element: `Data element attributes group` (see page 659), `Link relationship attribute group` (see page 661), `Universal attribute group` (see page 650), and `outputclass` (see page 0).

4.10.6.2.12 <completed>

The `<completed>` element indicates a completion date for some type of book event, such as a review, editing, or testing.

**Inheritance**
- topic/ph bookmap/completed

**Example**

See the example in `bookchangehistory` (see page 477).

**Attributes**

The following attributes are available on this element: `Universal attribute group` (see page 650), `outputclass` (see page 0), and `@keyref`. 
4.10.6.2.13 <copyrfirst>

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

**Inheritance**
- topic/data bookmap/copyrfirst

**Example**

See the example in `bookmap` (see page 461).

**Attributes**

The following attributes are available on this element: `Data element attributes group` (see page 659), `Link relationship attribute group` (see page 661), `Universal attribute group` (see page 650), and `outputclass` (see page 0).

4.10.6.2.14 <copyrlast>

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

**Inheritance**
- topic/data bookmap/copyrlast

**Example**

See the example in `bookmap` (see page 461).

**Attributes**

The following attributes are available on this element: `Data element attributes group` (see page 659), `Link relationship attribute group` (see page 661), `Universal attribute group` (see page 650), and `outputclass` (see page 0).

4.10.6.2.15 <day>

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

**Inheritance**
- topic/ph bookmap/day

**Example**

```xml
<bookchangehistory>
  <edited>
    <person>Joe T. Editor</person>
    <completed><year>2008</year><month>10</month><day>13</day></completed>
  </edited>
</bookchangehistory>
```

**Attributes**

The following attributes are available on this element: `Universal attribute group` (see page 650), `outputclass` (see page 0), and `@keyref`.
4.10.6.2.16 <edited>

The <edited> element contains information about when and by whom the book was edited during its publication history.

Inheritance
- topic/data bookmap/edited

Example
See the example in bookchangehistory (see page 477).

Attributes
The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.6.2.17 <edition>

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

Inheritance
- topic/data bookmap/edition

Example

```
<bookmeta>
  <bookid>
    <edition>1</edition>
  </bookid>
</bookmeta>
```

Attributes
The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.6.2.18 <isbn>


Inheritance
- topic/data bookmap/isbn

Example

```
<bookmeta>
  <bookid>
    <isbn>978-0141000039</isbn>
  </bookid>
</bookmeta>
```
4.10.6.2.19 <maintainer>

The <maintainer> element contains information about who maintains the document; the maintainer may be an organization or a person.

**Inheritance**
- topic/data bookmap/maintainer

**Example**
See the example in *bookmeta* (see page 475).

4.10.6.2.20 <month>

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

**Inheritance**
- topic/ph bookmap/month

**Example**

```xml
<bookchangehistory>
    <edited>
        <person>Joe T. Editor</person>
        <completed>
            <year>2008</year>
            <month>10</month>
            <day>13</day>
        </completed>
    </edited>
</bookchangehistory>
```

4.10.6.2.21 <organization>

The <organization> element contains the name of an organization. Note that unlike <organizationname>, the <organization> element is not restricted to use within <authorinformation>; it does not have to contain the name of an authoring organization.

**Inheritance**
- topic/data bookmap/organization

**Example**

```xml
<meta>
    <bookrights>
        <copyrfirst>
            <year>1996</year>
            <year></copyrfirst>
    </bookrights>
</meta>
```
Attributes

The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.6.2.22 <person>

The <person> element contains information about the name of a person. Note that unlike the <personname> element, the <person> element is not restricted to describing the names of authors.

Inheritance

- topic/data bookmap/person

Example

```dita
<bookmeta>
  <bookrights>
    <copyrfirst><year>1977</year></copyrfirst>
    <copyrlast><year>2008</year></copyrlast>
    <bookowner><person>Jane Doe</person></bookowner>
  </bookrights>
</bookmeta>
```

Attributes

The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.6.2.23 <printlocation>

The <printlocation> element indicates the location where the book was printed. Customarily, the content is restricted to the name of the country.

Inheritance

- topic/data bookmap/printlocation

Example

See the example in publisherinformation (see page 485).

Attributes

The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).
4.10.6.2.24 <published>

The <published> element contains information about the person or organization publishing the book, the dates when it was started and completed, and any special restrictions associated with it.

Inheritance
- topic/data bookmap/published

Example
See the example in publisherinformation (see page 485).

Attributes
The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.6.2.25 <publisherinformation>

The <publisherinformation> contains information about what group or person published the book, where it was published, and certain details about its publication history. Other publication history information is found in the <bookchangehistory> element.

Inheritance
- topic/publisher bookmap/publisherinformation

Example

```xml
<bookmeta>
  <publisherinformation>
    <organization>Example Publishers</organization>
    <printlocation>Austin, TX</printlocation>
    <published>
      <publishtype value="general"/>
      <completed><year>1977</year></completed>
    </published>
  </publisherinformation>
</bookmeta>
```

Attributes
The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.6.2.26 <publishtype>

The <publishtype> element indicates whether the book is generally available from the publisher or is restricted in some way. The @value attribute indicates the restrictions, such as beta release, limited availability, or general availability.

Inheritance
- topic/data bookmap/publishtype

Example
See the example in publisherinformation (see page 485).
**Attributes**

The following attributes are available on this element: *Data element attributes group* (see page 659), *Link relationship attribute group* (see page 661), *Universal attribute group* (see page 650), and *outputclass* (see page 0).

### 4.10.6.2.27 <reviewed>

The `<reviewed>` element contains information about when and by whom the book was reviewed during its publication history.

**Inheritance**

- topic/data bookmap/reviewed

**Example**

See the example in *bookchangehistory* (see page 477).

**Attributes**

The following attributes are available on this element: *Data element attributes group* (see page 659), *Link relationship attribute group* (see page 661), *Universal attribute group* (see page 650), and *outputclass* (see page 0).

### 4.10.6.2.28 <revisionid>

The `<revisionid>` element indicates the revision number or revision ID of the book. The processing implementation determines how the level is displayed. Common methods include using a dash, for example "-01", or a period, such as ".01".

**Inheritance**

- topic/ph bookmap/revisionid

**Example**

```xml
<bookchangehistory>
  <edited>
    <person>Joe T. Editor</person>
    <revisionid>1</revisionid>
    <completed><year>2008</year><month>03</month><day>15</day></completed>
  </edited>
  <edited>
    <person>Joe T. Editor</person>
    <revisionid>2</revisionid>
    <completed><year>2008</year><month>10</month><day>13</day></completed>
  </edited>
</bookchangehistory>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0 ), and *@keyref*.

### 4.10.6.2.29 <started>

The `<started>` element indicates a start date for some type of book event, such as a review, editing, or testing.

**Inheritance**

- topic/ph bookmap.started
Example

See the example in bookchangehistory (see page 477).

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and @keyref.

4.10.6.2.30 <summary>

The <summary> element contains a text summary associated with a book event (such as <approved> or <reviewed>) or with the list of copyrights for the book.

Inheritance

- topic/ph bookmap/summary

Example

```xml
<bookchangehistory>
  <edited>
    <person>Joe T. Editor</person>
    <revisionid>1</revisionid>
    <completed><year>2008</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>2008</year><month>10</month><day>13</day></completed>
    <summary>Fixed a few typos</summary>
  </edited>
</bookchangehistory>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and @keyref.

4.10.6.2.31 <tested>

The <tested> element contains information about when and by whom the book was tested during its publication history.

Inheritance

- topic/data bookmap/tested

Example

See the example in bookchangehistory (see page 477).

Attributes

The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0 ).
### 4.10.6.2.32 `<volume>`

The `<volume>` element contains the book's volume number, such as “2” to represent Volume 2.

**Inheritance**
- topic/data bookmap/volume

**Example**

```xml
<bookmeta>
  <bookid><volume>2</volume></bookid>
</bookmeta>
```

**Attributes**

The following attributes are available on this element: *Data element attributes group* (see page 659), *Link relationship attribute group* (see page 661), *Universal attribute group* (see page 650), and *outputclass* (see page 0).

### 4.10.6.2.33 `<year>`

The `<year>` element denotes a year.

**Inheritance**
- topic/ph bookmap/year

**Example**

```xml
<bookchangehistory>
  <edited>
    <person>Joe T. Editor</person>
    <completed><year>2008</year><month>10</month><day>13</day></completed>
  </edited>
</bookchangehistory>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *@keyref*.

### 4.10.7 Technical-content domains elements

Domains in this section include those generally associated with technical content, such as the programming and software domains.

#### 4.10.7.1 Equation domain elements

The elements in the equation domain enable authors to clearly distinguish equations from other type of content. These markup distinctions can enable formatting distinctions, numbering of equations, and more. This domain can be used independently of the MathML domain.

The equation domain elements do not attempt to satisfy all possible requirements for representing equations semantically. More sophisticated users of equations might need to further specialize from these elements or provide an alternative vocabulary for semantic equations.
4.10.7.1.1 <equation-block>

Use the <equation-block> element to represent an equation that is presented as a separate block within a text flow. Block equations can be numbered.

When an <equation-block> element has multiple direct child elements, each child represents an alternative form of the equation. Processors are free to choose the form or forms that they use in deliverables. For example, if there is both an image and MathML markup, an HTML-generating processor could output both the image reference and the MathML with appropriate HTML @class or @id values to enable dynamic showing or hiding of one form or the other based on browser capability.

Inheritance
+ topic/p equation-d/equation-block

Example

In the following example, a block equation uses MathML as its content:

```xml
<p>A block equation using MathML:</p>
<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>
```

In the following example, a block equation uses an image as its content:

```xml
<p>A block equation using an image:</p>
<equation-block>
 <image keyref="equation-image-01">
  <alt>a squared plus b squared.</alt>
 </image>
</equation-block>
```

In the following example, the block equation contains two alternative forms of the same equation:

```xml
<equation-block>
 <image keyref="equation-image-01">
  <alt>a squared plus b squared.</alt>
 </image>
<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>
```
4.10.7.1.2 <equation-figure>

Use the <equation-figure> element to represent an equation that functions as form of figure or display. Display equations are intended to be numbered when numbering is desired.

Since the <equation-figure> element is specialized from <fig>, display equations 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 <fig>, such as paragraphs.

When an <equation-figure> element has multiple direct child <mathml>, <image>, or <pre> elements, each child represents an alternative form of the equation. Processors are free to choose the form or forms that they use in deliverables. For example, if there is both an image and MathML markup, an HTML-generating processor could output both the image reference and the MathML with appropriate HTML @class or @id values to enable dynamic showing or hiding of one form or the other based on browser capability. All other direct-child elements of <equation-figure> are treated normally.

When the intent is to have equations combined with other commentary within an <equation-figure>, the recommended best practice is to use child <equation-block> elements to contain the equations and clearly distinguish them from the commentary.

Inheritance

+ topic/fig equation-d/equation-figure

Example

In the following example, the <equation-figure> element contains a title and MathML:

<equation-figure>
<title>Display equation with a MathML container</title>
<mathml>
<m:math display='block'>
<m:semantics>
<m:mrow>
<m:mfrac>
<m:mrow>
m:mi>n<m:mo>!</m:mo>
</m:mrow>
<m:mrow>
m:mi>r<m:mo>!</m:mo><m:mrow><m:mo>(</m:mo>
m:mi>n<m:mo>&#x2212;m:mi><m:mo>r</m:mo></m:mrow>
</m:mrow>
<m:mo>)!</m:mo></m:mrow><m:mrow><m:mo>!</m:mo></m:mrow>
</m:mfrac>
</m:mrow>
</m:semantics>
</m:math>
</equation-figure>
In the following example, the `<equation-figure>` element contains a title and an `<equation-block>` element that contains MathML and commentary:

```xml
<equation-figure>
  <title>Display equation with a MathML container</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:mo>!</m:mo>
              </m:mrow>
            </m:mfrac>
          </m:mrow>
        </m:semantics>
      </m:math>
    </mathml>
    <p>Where <equation-inline><mathml><m:math><m:mi>r</m:mi></m:math></mathml></equation-inline> is greater than 1.</p>
  </equation-block>
</equation-figure>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Display attribute group* (see page 660), *outputclass* (see page 0), and *spectitle* (see page 0).

4.10.7.1.3 `<equation-inline>`

Use the `<equation-inline>` element to represent an equation that is presented inline within a paragraph or similar context. 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. Processors are free to choose the form or forms that they use in deliverables. For example, if there is both an image and MathML markup, an HTML-generating processor could output both the image reference and the MathML with appropriate HTML `@class` or `@id` values to enable dynamic showing or hiding of one form or the other based on browser capability.

**Inheritance**

+ topic/ph equation-d/equation-inline

**Example**

The following example contains a paragraph that contains an `<equation-inline>` element with MathML markup:

```xml
<p>MathML inline: <equation-inline>
  <mathml>
    <m:math display='inline'>
      <m:msqrt>
        <m:mrow>
          <m:msup>
            <m:mi>a</m:mi>
            <m:mn>2</m:mn>
          </m:msup>
        </m:mrow>
      </m:msqrt>
      <m:mo>+</m:mo>
      <m:msup>
        <m:mi>b</m:mi>
        <m:mn>2</m:mn>
      </m:msup>
    </m:math>
  </mathml>
  <p>Where</p>
</equation-inline></p>
```
The following example contains an `<equation-inline>` element that uses an image as its content:

```xml
<p>An inline equation that contains an image:
<equation-inline>
  <image keyref="equation-image-01">
    <alt>a squared plus b squared.</alt>
  </image>
</equation-inline>
</p>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *@keyref*.

### 4.10.7.2 Markup domain

The markup domain elements are used for the mention of named constructs in markup languages, such as XML.

#### 4.10.7.2.1 `<markupname>`

Use the `<markupname>` element to identify mentions of named markup tokens, such as elements or attributes in XML or SGML, named groups in XSD schemas, or named patterns in RELAX NG schemas. This element also serves as the specialization basis for the elements in the XML mention domain; when that domain is present, its more specific elements should be used instead of `<markupname>` when appropriate.

**Inheritance**

+ topic/keyword markup-d/markupname

#### Example

The `<markupname>p.attributes</markupname>` attribute group defines the allowed attributes for the `<xmlelement>p</xmlelement>` element.

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *@keyref*.

### 4.10.7.3 MathML domain elements

The MathML domain elements enable direct use of MathML markup within DITA documents, as well as use-by-reference of MathML markup that is stored separate, non-DITA documents. MathML is a W3C standard.

For MathML markup that is stored directly 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 XSD or 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 Links**

[Mathematical Markup Language (MathML), Version 3.0](#)
4.10.7.3.1 <mathml>

Use the <mathml> element to contain content that contributes to a semantic equation. Such content includes MathML elements, references to MathML elements held in separate, non-DITA documents, <data>, or <data-about> 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".

Inheritance
+ topic/foreign mathml-d/mathml

Example

In the following example, a <mathml> element contains MathML content:

```xml
<equation-block>
  <mathml>
    <m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
      <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>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.10.7.3.2 <mathmlref>

Use the <mathmlref> element to reference a non-DITA XML document that contains MathML markup. This element enables you to use the 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

Processors SHOULD process the MathML as though the <m:math> element had occurred directly in the content of the containing <mathml> element.
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 <mathml> 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 "equation-02" within a larger document using a key reference, you would define the key in the following way:

```xml
<keydef keys="mathml-equation-02"
    href="math/mathml-equations.xml#equation-02"
    format="mathml"/>
```

You would refer to this key using just the key name:

```xml
<mathml>
  <mathmlref keyref="mathml-equation-02"/>
</mathml>
```

**Inheritance**

+ topic/xref mathml-d/mathmlref

**Example**

Here is a reference to a <mathml> element that is the root element of its containing document:

```xml
<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 `mathml` element sets the MathML namespace as the default namespace, so there are no namespace prefixes on the MathML markup:

```xml
<?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">)</mo>
        </mrow>
        <mi>π</mi>
      </mfrac>
    </mrow>
  </mstyle>
</math>
```

Here is a reference to a specific <mathml> element in a containing XML file:

```xml
<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
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <part>
    <math
      id="timeinday"
```
Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and @keyref. This element also uses @href, @scope, and a narrowed definition of @format (given below) from Link relationship attribute group (see page 661).

@format

Specifies the format (data type) of the referenced resource. For MathML the format should be "mml", which is the default value for @format on this element. This attribute is defined with the XML Data Type CDATA.

4.10.7.4 Programming elements

The programming domain elements are used to define the syntax and to give examples of programming languages.

4.10.7.4.1 <apiname>

The <apiname> element provides the name of an application programming interface (API) such as a Java class name or method name. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

Inheritance

+ topic/keyword pr-d/apiname

Example

<p>Use the <apiname>document.write</apiname> method to create text output in the dynamically constructed view.</p>
Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and @keyref.

4.10.7.4.2 <codeblock>

The <codeblock> element represents lines of program code. Like the <pre> element, line endings and spaces inside the element must be preserved, and the content is typically rendered in a monospaced font. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

Inheritance

+ topic/pre pr-d/codeblock

Example

```
<codeblock>
/* a long sample program */
Do forever
  Say "Hello, World"
End
</codeblock>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Display attribute group (see page 660), outputclass (see page 0 ), xml:space (see page 0 ), and spectitle (see page 0 ).

4.10.7.4.3 <coderef>

The <coderef> element references an external file that contains literal code. When evaluated, the <coderef> element should cause the target code to be displayed inline. If the target contains non-XML characters such as ‘<’ or ‘&’, those will need to be handled in a way that they may be displayed correctly by the final rendering engine. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

Inheritance

+ topic/xref pr-d/coderef

Example

```
<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>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661), outputclass (see page 0 ), and @keyref.
4.10.7.4.4 <codeph>

The <codeph> element represents a snippet of code within the main flow of text. The code phrase is displayed in a monospaced font for emphasis. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

Inheritance
+ topic/ph pr-d/codeph

Example

<p>The second line of the sample program code, <codeph>Do forever</codeph>, represents the start of a loop construct.</p>

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and @keyref.

4.10.7.4.5 <option>

The <option> element describes an option that can be used to modify a command (or something else, like a configuration). This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

Inheritance
+ topic/keyword pr-d/option

Example

<something <option>/modifier</option>

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and @keyref.

4.10.7.4.6 <parmname>

The <parmname> element is used to reference the name of an application programming interface parameter within the text flow of a topic. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

Inheritance
+ topic/keyword pr-d/parmname

Example

<p>Use the <parmname>/env</parmname> parameter of the <cmdname>config</cmdname> command to update the field value.</p>

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and @keyref.
4.10.7.4.7 <parml>

The `<parml>` element contains a list of terms and definitions that describes the parameters in an application programming interface. This is a special kind of definition list that is designed for documenting programming parameters. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

**Inheritance**

+ topic/dl pr-d/parml

**Example**

**Example source:**

```xml
<codeblock>
returnType methodName(pList1, pList2) {
  where
  <plentry>
    <pt>pList1</pt>
    <pd>is the first variable declaration passed to methodName</pd>
  </plentry>
  <plentry>
    <pt>pList2</pt>
    <pd>is the second variable declaration passed to methodName</pd>
  </plentry>
}</codeblock>
```

**Example output:**

This code example is a basic method signature:

```xml
returnType methodName(pList1, pList2) {
  where
  pList1
    is the first variable declaration passed to methodName
  pList2
    is the second variable declaration passed to methodName
}</xml>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *compact* (see page 0 ), *outputclass* (see page 0 ), and *spectitle* (see page 0 ).

4.10.7.4.8 <plentry>

The `<plentry>` element contains one or more parameter terms and definitions (`<pt>` and `<pd>`). This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

**Inheritance**

+ topic/dlentry pr-d/plentry

**Example**

See *parml* (see page 498).
4.10.7.4.9 `<pt>`

The `<pt>` element specifies a parameter term within a parameter list entry. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

**Inheritance**

+ topic/dt pr-d/pt

**Example**

See `parml` (see page 498).

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.10.7.4.10 `<pd>`

The `<pd>` element specifies a parameter definition within a parameter list entry. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

**Inheritance**

+ topic/dd pr-d/pd

**Example**

See `parml` (see page 498).

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and `@keyref`.

4.10.7.4.11 `<synph>`

The `<synph>` (syntax phrase) element is a container for syntax definition elements. It 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. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

**Inheritance**

+ topic/ph pr-d/synph

**Example**

```xml
<synph>
  <kwd>format</kwd>  <var>volumename</var>
</synph>
```
**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.10.7.4.12 `<syntaxdiagram>`

The `<syntaxdiagram>` element is the main container for all the syntax elements that make up a syntax definition. The syntax diagram represents the syntax of a statement from a computer language or a command, function call, or programming language statement. Traditionally, the syntax diagram is formatted with "railroad tracks" that connect the units of the syntax together, but this presentation may differ depending on the output media. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

**Inheritance**

+ topic/fig pr-d/syntaxdiagram

**Example**

```xml
<syntaxdiagram>
<title>CopyFile</title>
<groupseq><var>input-filename</var><kwd>*INFILE</kwd></groupseq>
<groupcomp><var>output-filename</var><kwd>*OUTFILE</kwd></groupcomp>
<groupchoice><var>input-filename</var><kwd>*INFILE</kwd></groupchoice>
<groupchoice><var>output-filename</var><kwd>*OUTFILE</kwd></groupchoice>
</syntaxdiagram>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Display attribute group* (see page 660), and *outputclass* (see page 0).

4.10.7.4.13 `<groupseq>`

The `<groupseq>` element is part of the subset of elements that define syntax diagrams in DITA. A 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. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

**Inheritance**

+ topic/figgroup pr-d/groupseq

**Example**

```xml
<syntaxdiagram frame="bottom">
<title>CopyFile</title>
<groupseq><var>COPYF</var><kwd>*INFILE</kwd></groupseq>
<groupcomp><var>input-filename</var><kwd>*OUTFILE</kwd></groupcomp>
<groupseq><var>output-filename</var><kwd>*OUTFILE</kwd></groupseq>
<groupchoice><var>input-filename</var><kwd>*INFILE</kwd></groupchoice>
<groupchoice><var>output-filename</var><kwd>*OUTFILE</kwd></groupchoice>
</syntaxdiagram>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of ![importance](given below)) and *outputclass* (see page 0).
@importance
The attribute indicates whether this item in a syntax diagram is optional, required, or default. Output processors may indicate this designation in a generated diagram. Allowable values are:

optional
This section of the syntax is optional.

required
This section of the syntax is required.

default
This section of the syntax is used by default or has this default value.

-dita-use-conref-target
See Using the -dita-use-conref-target value (see page 666) for more information.

4.10.7.4.14 <groupchoice>
The <groupchoice> element is part of the subset of elements that define syntax diagrams in DITA. A 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. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

Inheritance
+ topic/figgroup pr-d/groupchoice

Example

```xml
<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>
  <groupchoice><var>input-filename</var><kwd>*INFILE</kwd></groupchoice>
  <groupchoice><var>output-filename</var><kwd>*OUTFILE</kwd></groupchoice>
</syntaxdiagram>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @importance, given below) and outputclass (see page 0).

@importance
The attribute indicates whether this item in a syntax diagram is optional, required, or default. Output processors may indicate this designation in a generated diagram. Allowable values are:

optional
This section of the syntax is optional.

required
This section of the syntax is required.

default
This section of the syntax is used by default or has this default value.

-dita-use-conref-target
See Using the -dita-use-conref-target value (see page 666) for more information.

4.10.7.4.15 <groupcomp>
The <groupcomp> element is part of the subset of elements that define syntax diagrams in DITA. A 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 formatted close together rather than being separated by a horizontal or vertical line, which is the usual formatting method. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

Inheritance
+ topic/figgroup pr-d/groupcomp

Example

```xml
<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>
  <groupchoice><var>input-filename</var><kwd>*INFILE</kwd></groupchoice>
  <groupchoice><var>output-filename</var><kwd>*OUTFILE</kwd></groupchoice>
</syntaxdiagram>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @importance, given below) and outputclass (see page 0).

@importance

The attribute indicates whether this item in a syntax diagram is optional, required, or default. Output processors may indicate this designation in a generated diagram. Allowable values are:

optional
  This section of the syntax is optional.
required
  This section of the syntax is required.
default
  This section of the syntax is used by default or has this default value.
-dita-use-conref-target

See Using the -dita-use-conref-target value (see page 666) for more information.

4.10.7.4.16 <fragment>

The <fragment> element contains a labeled subpart of the syntax within a <syntaxdiagram>. The <fragment> element allows breaking out logical chunks of a large syntax diagram into named fragments. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

Inheritance
+ topic/figgroup pr-d/fragment

Example

```xml
<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>
  <groupchoice><var>input-filename</var><kwd>*INFILE</kwd></groupchoice>
  <groupchoice><var>output-filename</var><kwd>*OUTFILE</kwd></groupchoice>
  <fragment>
    <groupchoice><kwd>*OVERLAP</kwd><kwd>*Prompt</kwd></groupchoice>
  </fragment>
</syntaxdiagram>
```
**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.10.7.4.17 <fragref>

The `<fragref>` element provides a logical reference to a `<fragment>` element within a syntax diagram, so that you can reference a syntax fragment multiple times or pull a large section of syntax out of line for easier reading. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

**Inheritance**

+ topic/xref pr-d/fragref

**Example**

This markup example:

```xml
<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="#syntax/overlay"></fragref>
  <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>
```

may produce output like the following:

CopyFile

```plaintext
>>>-COPYF---input-filename*INFILE---output-filename---*OUTFILE------>

>---| Overlay |-----input-filename-------output-filename---------<
   '---*INFILE-------' '---*OUTFILE-------'

Overlay

|---*OVERLAP-----------------------------|
   '---*Prompt--'
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of `@importance`, given below), *outputclass* (see page 0), and the attributes defined below.

@href

A reference to a syntax diagram `<fragment>` element. The referenced `<fragment>` must be in the same diagram as the `<fragref>` element. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. For this element, processors should assume the equivalent of `scope="local"` and `format="dita"`.

@importance

The attribute indicates whether this item in a syntax diagram is optional or required. Output processors may indicate this designation in a generated diagram. Allowable values are:
optional
   This section of the syntax is optional.
required
   This section of the syntax is required.

-dita-use-conref-target
   See Using the -dita-use-conref-target value (see page 666) for more information.

4.10.7.4.18 <synblk>

The <synblk> (syntax block) element organizes small pieces of a syntax definition into a larger piece. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

Inheritance
+ topic/figgroup pr-d/synblk

Example

```xml
<synblk>
  <groupseq><kwdb>his</kwdb><sep>-</sep><kwdb>is</kwdb><sep>-</sep><kwdb>a</kwdb><sep>-</sep><var>test</var></groupseq>
</synblk>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.10.7.4.19 <synnote>

The <synnote> element contains a note (similar to a footnote) within a syntax definition group or fragment. The syntax note explains aspects of the syntax that cannot be expressed in the markup itself. The note will appear at the bottom of the syntax diagram instead of at the bottom of the page. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

Inheritance
+ topic/fn pr-d/synnote

Example

```xml
<groupcomp><var>one</var><var>two</var><var>three</var></groupcomp>
<synnote>My first syntax note.</synnote>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and the attribute defined below.

@callout
   Specifies what character is used for the footnote link, for example a number or an alpha character. The attribute may also specify a short string of characters. When no @callout value is specified, footnotes are numbered. This attribute is defined with the XML Data Type CDATA.
4.10.7.4.20 <synnoteref>

The <synnoteref> element references a syntax note element (<synnote>) that has already been defined elsewhere in the syntax diagram. The same notation can be used in more than one syntax definition. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

Inheritance
+ topic/xref pr-d/synnoteref

Example

```xml
<synnoteref href="#topicid/mysyn"/>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and the attribute defined below.

@href

A reference to the target syntax note (<synnote>) element. The referenced syntax note must be in the same syntax diagram as the <synnoteref> element. See The href attribute (see page 663) for detailed information on supported values and processing implications. For this element, processors should assume the equivalent of scope="local" and format="dita".

4.10.7.4.21 <kwd>

The <kwd> element defines a keyword within a syntax diagram. A keyword must be typed or output, either by the user or application, exactly as specified in the syntax definition. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

Inheritance
+ topic/keyword pr-d/kwd

Example

```xml
<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>
<groupchoice><var>input-filename</var><kwd>*INFILE</kwd></groupchoice>
<groupchoice><var>output-filename</var><kwd>*OUTFILE</kwd></groupchoice>
</syntaxdiagram>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @importance, given below), outputclass (see page 0), and @keyref.

@importance

The attribute indicates whether this item in a syntax diagram is optional, required, or default. Output processors may indicate this designation in a generated diagram. Allowable values are:

optional

This section of the syntax is optional.
required
   This section of the syntax is required.

default
   This section of the syntax is used by default or has this default value.
-dita-use-conref-target
   See Using the -dita-use-conref-target value (see page 666) for more information.

4.10.7.4.22 <var>

The <var> element defines a variable within a syntax diagram for which the user must supply content, such as their user name or password. Processors typically represent the <var> element in output in an italic font, but are not required to do so. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

Inheritance
+ topic/ph pr-d/var

Example

```xml
<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>
  <groupchoice><var>input-filename</var><kwd>*INFILE</kwd></groupchoice>
  <groupchoice><var>output-filename</var><kwd>*OUTFILE</kwd></groupchoice>
</syntaxdiagram>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @importance, given below) and outputclass (see page 0 ).

   @importance
   The attribute indicates whether this item in a syntax diagram is optional, required, or default. Output processors may indicate this designation in a generated diagram. Allowable values are:

   optional
   This section of the syntax is optional.

   required
   This section of the syntax is required.

   default
   This section of the syntax is used by default or has this default value.
-dita-use-conref-target
   See Using the -dita-use-conref-target value (see page 666) for more information.

4.10.7.4.23 <oper>

The <oper> element defines an operator within a syntax definition. Typical operators are equals (=), plus (+) or multiply (*). This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

Inheritance
+ topic/ph pr-d/oper
Example

```xml
<syntaxdiagram>
  <title>Adding</title>
  <groupseq>
    <kwd>1</kwd><oper>+</oper><var>two</var>
    <delim>=</delim><kwd>something</kwd>
  </groupseq>
</syntaxdiagram>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @importance, given below) and outputclass (see page 0).

@importance

The attribute indicates whether this item in a syntax diagram is optional, required, or default. Output processors may indicate this designation in a generated diagram. Allowable values are:

- optional
  This section of the syntax is optional.
- required
  This section of the syntax is required.
- default
  This section of the syntax is used by default or has this default value.

-dita-use-conref-target

See Using the -dita-use-conref-target value (see page 666) for more information.

4.10.7.4.24 <delim>

The `<delim>` element defines a character marking the beginning or end of a section within a syntax diagram. Typical delimiter characters are the parenthesis, comma, tab, vertical bar or other special characters. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

Inheritance

+ topic/ph pr-d/delim

Example

```xml
<syntaxdiagram>
  <title>Adding</title>
  <groupseq>
    <kwd>1</kwd><oper>+</oper><var>two</var><delim>-</delim>
    <kwd>something</kwd>
  </groupseq>
</syntaxdiagram>
```
This section of the syntax is required.

-dita-use-conref-target

See Using the -dita-use-conref-target value (see page 666) for more information.

4.10.7.4.25 <sep>

The `<sep>` element defines a separator character that is inline with the content of a syntax diagram. The separator occurs between keywords, operators or groups in a syntax definition. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

Inheritance

+ topic/ph pr-d/sep

Example

```xml
  <syntaxdiagram>
    <title>Adding</title>
    <groupseq><kwd>1</kwd><oper>+</oper><sep>)(</sep><var>two</var><sep>)</sep>
    <delim>=</delim><kwd>something</kwd></groupseq>
  </syntaxdiagram>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @importance, given below) and outputclass (see page 0 ).

@importance

The attribute indicates whether this item in a syntax diagram is optional, required, or default. Output processors may indicate this designation in a generated diagram. Allowable values are:

- optional
  This section of the syntax is optional.

- required
  This section of the syntax is required.

- default
  This section of the syntax is used by default or has this default value.

-dita-use-conref-target

See Using the -dita-use-conref-target value (see page 666) for more information.

4.10.7.4.26 <repsep>

The `<repsep>` element provides a repeat separator character within a syntax diagram to indicate that a group of syntax elements can (or should) be repeated. 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. This element is part of the DITA programming domain, a special set of DITA elements designed to document programming tasks, concepts, and reference information.

Inheritance

+ topic/ph pr-d/repsep

Example

In this example, the group may be repeated. When repeated, a comma should be used between selections.

```xml
  <groupchoice>
    <repsep>,</repsep>
  </groupchoice>
```
Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @importance, given below) and outputclass (see page 0).

@importance

The attribute indicates whether this item in a syntax diagram is optional or required. Output processors may indicate this designation in a generated diagram. Allowable values are:

optional

This section of the syntax is optional.

required

This section of the syntax is required.

-dita-use-conref-target

See Using the -dita-use-conref-target value (see page 666) for more information.

4.10.7.5 Release-management domain elements

The release-management domain elements contain human-authored information about the changes that have been made to a DITA topic or map. A processor can retrieve this information and use it to assemble documents or topics that contain release note information.

4.10.7.5.1 <change-completed>

Use the <change-completed> element to indicate the date on which the change was completed.

The recommended best practice is to use date strings that conform to the ISO 8601 standard, unless a Unix-style machine timestamp is used. The string might contain a date and time (for example, 2007-04-05T12:30-02:00) or just a date (for example, 2014-03-04).

Inheritance

- topic/data relmgmt-d/change-completed

Example

See change-historylist (see page 509).

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and name (see page 0).

4.10.7.5.2 <change-historylist>

Use the <change-historylist> element as a container for individual release notes, as represented by the <change-item> element. It appears in the topic prolog or the map <topicmeta> element.

Inheritance

- topic/metadata relmgmt-d/change-historylist
Example

This example shows three simple release notes added to a single topic. This topic is used in documentation for two products, A and B.

```xml
<prolog>
<!-- ... -->
<change-historylist>
  <change-item product="productA productB">
    <change-person>Tom Cihak</change-person>
    <change-organization>JEDEC</change-organization>
    <change-completed>2013-03-23</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>2013-06-07</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-completed>2013-07-20</change-completed>
    <change-summary>Made change 3 to both products</change-summary>
    <data>Details of change 3</data>
  </change-item>
</change-historylist>
<!-- ... -->
</prolog>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and mapkeyref (see page 0).

4.10.7.5.3 <change-item>

Use the `<change-item>` element to represent a record of a single change to a DITA topic or map.

Inheritance

- topic/data relmgmt-d/change-item

Example

See `change-historylist` (see page 509).

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and name (see page 0).

4.10.7.5.4 <change-organization>

Use the `<change-organization>` element to specify the name of an organization that required the change.

Inheritance

- topic/data relmgmt-d//change-organization

Example

See `change-historylist` (see page 509).
**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *name* (see page 0).

4.10.7.5.5 `<change-person>`

Use the `<change-person>` element to specify the name of the person who made the change.

**Inheritance**

- topic/data relmgmt-d/change-person

**Example**

See *change-historylist* (see page 509).

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *name* (see page 0).

4.10.7.5.6 `<change-request-id>`

Use the `<change-request-id>` element to specify an identifier associated with the change request, such as an issue ID or ticket number.

**Inheritance**

- topic/data relmgmt-d//change-request-id

**Example**

See *change-historylist* (see page 509).

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *name* (see page 0).

4.10.7.5.7 `<change-request-reference>`

Use the `<change-request-reference>` element to provide information that links the change to an external tracking system.

**Inheritance**

- topic/metadata relmgmt-d//change-request-reference

**Example**

See *change-historylist* (see page 509).

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *name* (see page 0).
4.10.7.5.8 <change-request-system>

Use the <change-request-system> element to specify the name of an information system that manages or serves the referenced change request, for example, an issue tracking system.

Inheritance
- topic/data relmgmt-d/change-request-system

Example
See change-historylist (see page 509).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and name (see page 0).

4.10.7.5.9 <change-revisionid>

Use the <change-revisionid> element to specify a revision ID string that can identify the change.

Inheritance
- topic/data relmgmt-d/change-revisionid

Example
See change-historylist (see page 509).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and name (see page 0).

4.10.7.5.10 <change-started>

Use the <change-started> element to specify the date on which the change was initiated.

The recommended best practice is to use date strings that conform to the ISO 8601 standard, unless a Unix-style machine timestamp is used. The string might contain a date and time (for example, 2007-04-05T12:30-02:00) or just a date (for example, 2014-03-04).

Inheritance
- topic/data relmgmt-d/change-started

Example
See change-historylist (see page 509).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and name (see page 0).
4.10.7.5.11 <change-summary>

Use the <change-summary> element to summarize the change. It is the portion of the release note that might appear in a document.

Inheritance
- topic/data relmgmt-d/change-summary

Example
See change-historylist (see page 509).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and name (see page 0).

4.10.7.6 Software elements

The software domain elements are used to describe the operation of a software program.

4.10.7.6.1 <msgph>

The <msgph> element is a message phrase that contains the text content of a message produced by an application or program. It can also contain a variable name (<varname>) element to illustrate where variable text content can occur in the message. This element is part of the DITA software domain, a special set of DITA elements designed to document software tasks, concepts and reference information.

Inheritance
+ topic/ph sw-d/msgph

Example

<p>A server log entry of <msgnum>I:0</msgnum> is equivalent to the text message, <msgph>informational: successful</msgph>.</p>

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and @keyref.

4.10.7.6.2 <msgblock>

The <msgblock> element contains a multi-line message or set of messages. The message block can contain multiple message numbers and message descriptions, each enclosed in <msgnum> and <msgph> elements. It can also contain the message content directly. Line breaks and spaces are preserved when the element is rendered. This element is part of the DITA software domain, a special set of DITA elements designed to document software tasks, concepts and reference information.

Inheritance
+ topic/pre sw-d/msgblock

Example

<p>A sequence of failed password attempts generates the following characteristic message stream:</p>
<msgblock>

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *Display attribute group* (see page 660), *outputclass* (see page 0 ), *xml:space* (see page 0 ), and *spectitle* (see page 0 ).

4.10.7.6.3 <msgnum>

The <msgnum> element contains the number of a message produced by an application or program. This element is part of the DITA software domain, a special set of DITA elements designed to document software tasks, concepts and reference information.

Inheritance

+ topic/keyword sw-d/msgnum

Example

```xml
<p>A server log entry of <msgnum>I:0</msgnum> is equivalent to the text message, <msgph>informational: successful</msgph>.</p>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0 ), and @keyref.

4.10.7.6.4 <cmdname>

The <cmdname> element specifies the name of a command when it is part of a software discussion. This element is part of the DITA software domain, a special set of DITA elements designed to document software tasks, concepts and reference information.

Inheritance

+ topic/keyword sw-d/cmdname

Example

```xml
<p>Use the <cmdname>rm</cmdname> command when you wish to remove something forever.</p>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0 ), and @keyref.

4.10.7.6.5 <varname>

The <varname> element specifies a variable that must be supplied to a software application. The variable name element is very similar to the <var> element, but <varname> is available outside of syntax diagrams. This
element is part of the DITA software domain, a special set of DITA elements designed to document software tasks, concepts and reference information.

**Inheritance**

+ topic/keyword sw-d/varname

**Example**

```
<filepath>
  <varname>install-dir</varname>
  \projects\working\<varname>project-dir</varname>
  \source\<varname>filename</varname>.java
</filepath>
```

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0 ), and *@keyref*.

4.10.7.6.6 `<filepath>`

The `<filepath>` element indicates the name of a referenced file, system path, or both. This element is part of the DITA software domain, a special set of DITA elements designed to document software tasks, concepts and reference information.

**Inheritance**

+ topic/ph sw-d/filepath

**Example**

```
<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>
```

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0 ), and *@keyref*.

4.10.7.6.7 `<userinput>`

The `<userinput>` element represents the text a user should input in response to a program or system prompt. This element is part of the DITA software domain, a special set of DITA elements designed to document software tasks, concepts and reference information.

**Inheritance**

+ topic/ph sw-d/userinput

**Example**

```
<p>After you type `<userinput>mealplan dinner</userinput>`, the meal planning program will print a `<systemoutput>For what day?</systemoutput>` message. Reply by typing the day of the week for which you want a meal plan, for example, `<userinput>Thursday</userinput>`.</p>
```
Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and @keyref.

4.10.7.6.8 <systemoutput>

The <systemoutput> element represents computer output or responses to a command or situation. A generalized element, it represents any kind of output from the computer, so the author may wish to choose more specific markup, such as <msgph> for messages from the application. The system output element is part of the DITA software domain, a special set of DITA elements designed to document software tasks, concepts and reference information.

Inheritance
+ topic/ph sw-d/systemoutput

Example

```
<p>After you type <userinput>mealplan dinner</userinput>, the meal planning program will print a <systemoutput>For what day?</systemoutput> message. Reply by typing the day of the week for which you want a meal plan, for example, <userinput>Thursday</userinput>.</p>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and @keyref.

4.10.7.7 SVG elements

The SVG domain elements enable direct use of SVG markup within DITA documents, as well as use-by-reference of SVG markup that is stored in separate, non-DITA 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 XSD or 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 Links
Scalable Vector Graphics (SVG) 1.1 (Second Edition)

4.10.7.7.1 <svg-container>

Use the <svg-container> element to store content that contributes to a scalable vector graphic (SVG). Such content includes SVG elements, references to SVG elements that are stored in separate, non-DITA documents, <data>, or <data-about> elements.

The SVG markup must have a root element of <svg> with the SVG namespace: "http://www.w3.org/2000/svg".

Inheritance
+ topic/foreign svg-d/svg-container
Example

In the following example, an `<svg-container>` element contains inline SVG markup:

```xml
<topic id="svg-test-topic-01">
  <title>SVG Domain Test: Namespace Prefixed SVG Elements</title>
  <body>
    <p>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>
    <p>SVG Directly in body:</p>
    <svg-container>
      <svg:svg width="200" height="200">
        <svg:ellipse cx="100" cy="100" rx="80" ry="80" style="fill:blue; stroke:rgb(0,0,100);stroke-width:2"/>
      </svg:svg>
    </svg-container>
    <fig>
      <title>Figure With SVG Container</title>
      <svg-container>
        <svgref keyref="svg-fragment-01"/>
      </svg-container>
    </fig>
  </body>
</topic>
```

In the following example, the `<svgref>` element uses the `@keyref` attribute to address SVG markup that is stored in a separate, non-DITA document:

```xml
<fig>
  <title>Figure with SVG Container</title>
  <svg-container>
    <svgref keyref="svg-container"/>
  </svg-container>
</fig>
```
The key "svg-fragment-01" is declared in the following way:

```xml
<map>
  ...
  <keydef keys="svg-fragment-01"
    href="media/svg/svg-library.xml#frag-0001"
    format="svg"
  />
  ...
</map>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.10.7.7.2 `<svgref>`

Use the `<svgref>` element to reference a non-DITA XML document that contains scalable vector graphic (SVG) markup. This element enables you to use the markup by reference.

The reference must be to a SVG `<svg>` element. The reference can be one of the following:

- A URI that addresses an XML document; the XML document 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

Processors *SHOULD* process the SVG as though the `<svg>` element had occurred directly in the content of the containing `<svg_container>` element.

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:

```xml
<keydef keys="svg-fragment-0002"
  href="svg/svg-library.xml#svg-fragment-02"
/>
```

You would refer to this key using just the key name:

```xml
<svg_container>
  <svgref keyref="svg-fragment-0002"/>
</svg_container>
```

**Inheritance**

+ topic/xref svg-d/svgref

**Example**

Here is a reference to an `<svg>` element that is the root element of its containing document:

```xml
<fig>
  <title>Figure With SVG Container</title>
  <svg_container>
    <svgref
      href="media/svg/svg-graphic-01.xml"
      format="svg"
    />
  </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
<?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)="/n</svg>
</fig>

Here is a reference to a specific `<svg>` element in a containing XML file:

```xml
<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
<?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)="/n</svg>
    </part>
    <svg id="frag-0002"
         width="4in" height="6in" version="1.1">
      <circle cx="150" cy="200" r="100" fill="url(#grad_blue)"
             animate attributeName="r" begin="Go.click" end="Stop.click" dur="4s"
             values="100; 0; 100" repeatCount="indefinite"/>
    </circle>
    <radialGradient id="grad_blue" cx="20%" cy="20%" r="100%" fx="30%" fy="30%">
      <stop stop-color="white" offset="0="/n</radialGradient>
```

DITA Version 1.3 Specification

DITA XML Reference

Copyright © OASIS Open 2005, 2014. All Rights Reserved.
Page 519 of 747
Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and @keyref. This element also uses Link relationship attribute group (see page 661), with narrowed definitions of @href and @format (given below).

@href

Reference to the <svg> element to be used. 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 to be used. This attribute is not required, but must be specified if @keyref is not specified. See The href attribute (see page 663) for detailed information on supported values and processing implications.

@format

The data format of the resource. For SVG the format should be "svg", which is the default value for @format on this element. This attribute is defined with the XML Data Type CDATA.

4.10.7.8 Task requirements domain

The task requirements domain contains elements for use in describing tasks that involve machines or other pieces of hardware.

4.10.7.8.1 <prelreqs>

The <prelreqs> element contains information about preliminary requirements – the things the user needs to know or do before starting a task. This element contains information about personnel requirements, safety conditions, support equipment, supplies, and spare parts.

Inheritance

+ topic/section task/prereq taskreq-d/prelreqs

Example

<prelreqs>
  <reqconds>
    <reqcond>Rear Oil Seal replacement</reqcond>
    <reqcontp>Motor Oil Guide</reqcontp>
  </reqconds>
  <reqpers>
    <personnel>2</personnel>
    <perscat>Mechanic</perscat>
    <perskill>Expert</persskill>
    <esttime>2 hours</esttime>
  </reqpers>
  <supequip>
    <supequi>Driver handle</supequi>
    <supequi>Slide hammer</supequi>
  </supequip>
</prelreqs>
Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and spectitle (see page 0).

4.10.7.8.2 <closereqs>

The <closereqs> element contains information about closing requirements – steps or tasks that the user should perform after completing a task, for example, "Check around the vehicle for any drips or leaks."

Inheritance
+ topic/section task/postreq taskreq-d/closereqs

Example

```
<closereqs>
  <reqconds>
    <reqcond>Run the engine and then check the dipstick to ensure the vehicle has enough oil. </reqcond>
    <reqcond>Check around the vehicle for any drips or leaks. </reqcond>
  </reqconds>
</closereqs>
```
Attributes
The following attributes are available on this element: Universal attribute group (see page 650), compact (see page 0), outputclass (see page 0), and spectitle (see page 0).

4.10.7.8.4 <reqcond>
The <reqcond> element specifies a condition that must be fulfilled before performing a task, for example, "Rear Oil Seal replacement."

Inheritance
+ topic/li task/li taskreq-d/reqcond

Example

```xml
<reqconds>
  <reqcond>The system placed on an accessible, flat surface.</reqcond>
  <reqcond>All anti-static discharge wrist-straps available.</reqcond>
</reqconds>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.10.7.8.5 <noconds>
The <noconds> element specifies that there are no conditions to be fulfilled before performing a task.

Inheritance
+ topic/li task/li taskreq-d/noconds

Example

```xml
<reqconds>
  <noconds/>
</reqconds>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.10.7.8.6 <reqcontp>
The <reqcontp> element specifies a technical publication that can be used to fulfill a condition before performing a task. It may also specify a publication that is required in order to fulfill the condition, such as a list of local regulations.

Inheritance
+ topic/li task/li taskreq-d/reqcontp

Example

```xml
<reqconds>
  <reqcontp>USDOD-43109: Preparing your Patriot Missile Control Unit for Railroad Portage.</reqcontp>
  <reqcond>The system placed on an accessible, flat surface.</reqcond>
</reqconds>
```
Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.10.7.8.7 <reqpers>

The <reqpers> element contains information about the personnel who are required to perform a task. This information might specify the number of workers, the type and skill level of the workers, and the length of time that they will need to perform the task.

Inheritance
+ topic/ol task/ol taskreq-d/reqpers

Example

```xml
<reqpers>
  <personnel>2</personnel>
  <perscat>Mechanic</perscat>
  <perskill>Expert</perskill>
  <esttime>2 hours</esttime>
</reqpers>
```

Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650), *compact* (see page 0), *outputclass* (see page 0), and *spectitle* (see page 0).

4.10.7.8.8 <personnel>

The <personnel> element specifies the minimum number of workers who are required to perform a task.

Inheritance
+ topic/li task/li taskreq-d/personnel

Example

```xml
<reqpers>
  <personnel>2</personnel>
  <perscat>Mechanic</perscat>
  <perskill>Expert</perskill>
  <esttime>2 hours</esttime>
</reqpers>
```

Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.10.7.8.9 <perscat>

The <perscat> element specifies the type or category of workers that is required by a task.

Inheritance
+ topic/li task/li taskreq-d/perscat
Example

```
<reqpers>
  <personnel>2</personnel>
  <perscat>Mechanic</perscat>
  <perskill>Expert</perskill>
  <esttime>2 hours</esttime>
</reqpers>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

### 4.10.7.8.10 <perskill>

The `<perskill>` element specifies the skill level of the workers who must perform the task.

**Inheritance**

+ topic/li task/li taskreq-d/perskill

Example

```
<reqpers>
  <personnel>2</personnel>
  <perscat>Mechanic</perscat>
  <perskill>Expert</perskill>
  <esttime>2 hours</esttime>
</reqpers>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

### 4.10.7.8.11 <esttime>

The `<esttime>` element provides an estimate of the time that is required to perform a task.

**Inheritance**

+ topic/li task/li taskreq-d/esttime

Example

```
<reqpers>
  <personnel>2</personnel>
  <perscat>Mechanic</perscat>
  <perskill>Expert</perskill>
  <esttime>2 hours</esttime>
</reqpers>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).
4.10.7.8.12 <supeqli>

The <supeqli> element contains a list of the tools, support equipment, or monitoring equipment that is required to perform a task. These pieces of support equipment need to be assembled prior to beginning the task.

Inheritance
+ topic/ul task/ul taskreq-d/supeqli

Example

```xml
<supequip>
  <supeqli>
    <supequi>Driver handle</supequi>
    <supequi>Slide hammer</supequi>
    <supequi>Axle seal installer</supequi>
  </supeqli>
</supequip>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), compact (see page 0), outputclass (see page 0), and spectitle (see page 0).

4.10.7.8.13 <supequi>

The <supequi> element specifies a tool, a piece of support equipment, or a piece of monitoring equipment that is needed to perform a task, such as a slide hammer.

Inheritance
+ topic/li task/li taskreq-d/supequi

Example

```xml
<supequip>
  <supequi>Driver handle</supequi>
  <supequi>Slide hammer</supequi>
  <supequi>Axle seal installer</supequi>
</supequip>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.10.7.8.14 <supequip>

The <supequip> element contains information about the support equipment that is required to perform a task. This element either contains children elements that specify particular items of support equipment or a <nosupeq> element that specifies that no support equipment is required.

Inheritance
+ topic/p task/p taskreq-d/supequip

Example

```xml
<supequip>
  <supequi>Driver handle</supequi>
</supequip>
```
Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.10.7.8.15 <nosupeq>

The <nosupeq> element indicates that there is no support equipment required to perform a task.

Inheritance
+ topic/data task/data taskreq-d/nosupeq

Example

```xml
<supeq>
  <nosupeq/>
</supeq>
```

Attributes
The following attributes are available on this element: *Data element attributes group* (see page 659), *Link relationship attribute group* (see page 661), *Universal attribute group* (see page 650), @keyref, and *outputclass* (see page 0).

4.10.7.8.16 <supplies>

The <supplies> element contains information about the supplies or parts that are required to perform a task. These supplies or parts need to be available prior to beginning the task. This element either contains child elements that specify particular supplies or a <nosupply> element that indicates that no supplies are needed.

Inheritance
+ topic/p task/p taskreq-d/supplies

Example

```xml
<supplies>
  <supplyli>
    <supply>gasket</supply>
    <supply>engine oil</supply>
  </supplyli>
</supplies>
```

Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.10.7.8.17 <supply>

The <supply> element contains information about a single supply in a list of supplies that are needed to perform a task.

Inheritance
+ topic/li task/li taskreq-d/supply
Example

```xml
<supplies>
  <supplyli>
    <supply>gasket</supply>
    <supply>engine oil</supply>
  </supplyli>
</supplies>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.10.7.8.18 <supplyli>
The <supplyli> element specifies a list of supplies needed to perform a task.

Inheritance
+ topic/ul task/ul taskreq-d/supplyli

Example

```xml
<supplies>
  <supplyli>
    <supply>gasket</supply>
    <supply>engine oil</supply>
  </supplyli>
</supplies>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), compact (see page 0), outputclass (see page 0), and spectitle (see page 0).

4.10.7.8.19 <nosupply>
The <nosupply> element specifies that no supplies are needed to perform a task.

Inheritance
+ topic/data task/data taskreq-d/nosupply

Example

```xml
<supplies>
  <nosupply/>
</supplies>
```

Attributes
The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), @keyref, and outputclass (see page 0).
4.10.7.8.20 <spare>

The <spare> element specifies a particular spare part that is required to perform a task, for example, a "dipstick."

Inheritance
+ topic/li task/li taskreq-d/spare

Example

```xml
<spares>
  <sparesli>
    <spare>dipstick</spare>
    <spare>engine oil</spare>
  </sparesli>
</spares>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0 ).

4.10.7.8.21 <spares>

The <spares> element contains information about the spare parts that are needed for a task. This information might specify particular spare parts or it might state that no spare parts are required.

Inheritance
+ topic/p task/p taskreq-d/spares

Example

```xml
<spares>
  <sparesli>
    <spare>dipstick</spare>
    <spare>engine oil</spare>
  </sparesli>
</spares>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0 ).

4.10.7.8.22 <sparesli>

The <sparesli> element contains information about the spare parts that are required to perform a task.

Inheritance
+ topic/ul task/ul taskreq-d/sparesli

Example

```xml
<spares>
  <sparesli>
    <spare>dipstick</spare>
    <spare>engine oil</spare>
  </sparesli>
</spares>
```
**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *compact* (see page 0 ), *outputclass* (see page 0 ), and *spectitle* (see page 0 ).

### 4.10.7.8.23 `<nospares>`

The `<nospares>` element specifies that no spare parts are needed to perform a task.

**Inheritance**

+ topic/data task/data taskreq-d/nospares

**Example**

```
<sares>
  <nospares/>
</sares>
```

**Attributes**

The following attributes are available on this element: *Data element attributes group* (see page 659), *Link relationship attribute group* (see page 661), *Universal attribute group* (see page 650), @keyref, and *outputclass* (see page 0 ).

### 4.10.7.8.24 `<nosafety>`

The `<nosafety>` element specifies that there are no safety conditions that must be considered.

**Inheritance**

+ topic/li task/li taskreq-d/nosafety

**Example**

```
<safety>
  <nosafety/>
</safety>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0 ).

### 4.10.7.8.25 `<safecond>`

The `<safecond>` element specifies a safety condition that must be considered prior to completing a task. It may also contain a complete hazard statement.

**Inheritance**

+ topic/li task/li taskreq-d/safecond

**Example**

```
<safety>
  <safecond>All power sources disconnected from the system.</safecond>
  <safecond>All networking cables disconnected from the system.</safecond>
</safety>
```
Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.10.7.8.26 <safety>

The `<safety>` element contains information about safety conditions. This element either contains children elements that describe safety conditions or a `<nosafety>` element that indicates that there are no safety conditions that must be considered.

Inheritance

+ topic/ol task/ol taskreq-d/safety

Example

```
<safety>
  <safecond>All power sources disconnected from the system.</safecond>
  <safecond>All networking cables disconnected from the system.</safecond>
</safety>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *compact* (see page 0), *outputclass* (see page 0), and *spectitle* (see page 0).

4.10.7.9 User interface elements

The user interface domain elements are used to describe the user interface of a software program.

4.10.7.9.1 <uicontrol>

The `<uicontrol>` element is used to mark up user interface controls, such as names of buttons, entry fields, menu items, or other objects that allow the user to control the interface. Use the `<uicontrol>` element inside a `<menucascade>` element to identify a sequence of menu choices in a nested menu, such as File > New. This element is part of the DITA user interface domain, a special set of DITA elements designed to document user interface tasks, concepts and reference information.

Inheritance

+ topic/ph ui-d/uicontrol

Example

```
<p>Press the <uicontrol>OK</uicontrol> button.</p>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *@keyref*.

4.10.7.9.2 <wintitle>

The `<wintitle>` element can be used to mark up names of windows or dialogs, or other user interface elements at the same level of grouping, including wizard titles, wizard page titles, and window pane titles. This element is part of the DITA user interface domain, a special set of DITA elements designed to document user interface tasks, concepts and reference information.
Inheritance
+ topic/keyword ui-d/wintitle

Example

<pre>
<step>
  <cmd>Click <uicontrol>Configure</uicontrol>.</cmd>
  <stepresult>The <wintitle>Configuration Options</wintitle> window opens with your last set of selections highlighted.</stepresult>
</step>
</pre>

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and @keyref.

4.10.7.9.3 <menucascade>

The <menucascade> element is used to document a series of menu choices. The <menucascade> element contains one or more user interface control (<uicontrol>) elements, for example: Start > Programs > Accessories > Notepad. If there is more than one <uicontrol> element, the formatter shows connecting characters between the menu items to represent the menu cascade. This element is part of the DITA user interface domain, a special set of DITA elements designed to document user interface tasks, concepts and reference information.

Comment by robander, 25 February 2014
Wondering at the wording here - "the formatter shows connecting characters". Better phrasing with the same meaning would be "formatters must show connecting characters" (or even formatters "should") but this may be adding a new RFC-level requirement that didn't previously exist. Potentially better to change to "formatters may show connecting..."

Inheritance
+ topic/ph ui-d/menucascade

Example

This example:

<pre>
<menucascade>
  <uicontrol>Start</uicontrol>
  <uicontrol>Programs</uicontrol>
  <uicontrol>Accessories</uicontrol>
  <uicontrol>Notepad</uicontrol>
</menucascade>
</pre>

May produce output similar to the following: Start > Programs > Accessories > Notepad

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and @keyref.
4.10.7.9.4 <shortcut>

The <shortcut> element identifies a keyboard shortcut for a menu or window action. This element is part of the DITA user interface domain, a special set of DITA elements designed to document user interface tasks, concepts and reference information.

Inheritance
+ topic/keyword ui-d/shortcut

Example

This example:

```dita
<menucascade>
  <uicontrol>Start</uicontrol>
  <uicontrol><shortcut>P</shortcut>rogram</uicontrol>
</menucascade>
```

May produce output similar to the following: Start > Programs

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and @keyref.

4.10.7.9.5 <screen>

The <screen> element contains or refers to a textual representation of a computer screen or user interface panel (window).

Use <screen> to contain representations of text-based online panels, text consoles ("term" or "curses" windows, for example), or other text-based user interface components. The default print representation is to enclose the screen within a box, suggesting a computer display screen. In contrast to graphical screen captures normally used to represent GUI parts (see the <image> element description), this element specifically supports constructions for which text is the primary content.

This element is part of the DITA user interface domain, a special set of DITA elements designed to document user interface tasks, concepts and reference information.

Inheritance
+ topic/pre ui-d/screen

Example

```
<p>Type "edit" after the command line prompt and press Enter. The following editing interface will be displayed.</p>
<screen>
  File  Edit  Search  View  Options  Help
  +----------------------------------------------------------+
  |                                                             |
  |                                                             |
  |                                                             |
  |                                                             |
  |  Line:1    Col:1  F1=Help                                   |
  +----------------------------------------------------------+
</screen>
```

Figure 104: Using the <screen> element to represent a DOS edit session

Figure 105: Potential result from an rendering process

Type "edit" after the command line prompt and press Enter. The following editing interface will be displayed.
4.10.7.10 XML mention domain

Use the XML-mention domain elements for mentions of named XML constructs, including elements, attributes, entities, processing instructions, and document-type declaration components. These elements enable specific typographic effects for different construct types, precise search and retrieval of specific constructs, and automatic indexing of different constructs. This domain is intended to support the description and documentation of XML document types and XML applications.

Note: Although the original XML 1.0 Recommendation specifically 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 semantically precise prefix "xml" for the XML mention domain.

4.10.7.10.1 <numcharref>

Use the <numcharref> element to identify mentions of XML numeric character references, for example, "&#10;" or "&#x0a;". This element enables more precise semantic searching of the DITA source; it also enables distinct output formatting. The content of the <numcharref> element should be the numeric value without any leading or trailing characters added as style, for example, "10" or "x0a". This element is part of the XML mention domain.

Inheritance
+ topic/keyword markup-d/markupname xml-d/numcharref

Example

<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: a-acute.</p>

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), xml:space (see page 0 ), and spectitle (see page 0 ).
entity name. The content of the `<parameterentity>` element should be the entity name without the leading percentage sign or trailing semi-colon. This element is part of the XML mention domain.

**Inheritance**

+ topic/keyword markup-d/markupname xml-d/parameterentity

**Examples**

```xml
<p>... The `<parameterentity>p.content</parameterentity>` parameter entity defines the content model for the `<xmlelement>p</xmlelement>` element.</p>

<p>To include the XML-mention domain in a DTD document-type shell, you must declare the `<parameterentity>xml-d-dec</parameterentity>` parameter entity.</p>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), `outputclass` (see page 0), and `@keyref`.

### 4.10.7.10.3 `<textentity>`

Use the `<textentity>` element to identify mentions of XML text entities, for example, "&xml-d-att;". This element enables more precise semantic searching of the DITA source; it also enables distinct formatting, such as adding an ampersand (&) as a leading character and a semi-colon (;) as a trailing character. The content of the `<textentity>` element should be the entity name without the ampersand and semi-colon delimiters. This element is part of the XML mention domain.

**Inheritance**

+ topic/keyword markup-d/markupname xml-d/textentity p

**Example**

```xml
The `<textentity>hi-d-att</textentity>` entity holds the contribution for the `<xmlatt>domains</xmlatt>` attribute.
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), `outputclass` (see page 0), and `@keyref`.

### 4.10.7.10.4 `<xmlatt>`

Use the `<xmlatt>` element to identify mentions of XML attributes, for example, `@audience`. This element enables more precise semantic searching of the DITA source; it also enables distinct formatting, such as preceding the name of the attribute with a commercial at (@) sign. The content of the `<xmlatt>` element should be the attribute name without commercial at or equals (=) characters. This element is part of the XML mention domain.

**Inheritance**

+ topic/keyword markup-d/markupname xml-d/xmlatt
Example

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.

**Attributes**

The following attributes are available on this element: Universal attribute group (see page 650), `outputclass` (see page 0), and `@keyref`.

### 4.10.7.10.5 `<xmllelement>`

Use the `<xmllelement>` element to identify mentions of XML element types. This element enables more precise semantic searching of the DITA source; it also enables distinct formatting, such as surrounding the name of the element with angle brackets. The content of the `<xmllelement>` element should be the element type name without leading or trailing angle brackets. This element is part of the XML mention domain.

**Inheritance**

+ topic/keyword markup-d/markupname xml-d/xmllelement

**Example**

```xml
<p>Use the `<xmllelement>uicontrol</xmllelement>` (user interface control) 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>
```

**Attributes**

The following attributes are available on this element: Universal attribute group (see page 650), `outputclass` (see page 0), and `@keyref`.

### 4.10.7.10.6 `<xmlnsname>`

Use the `<xmlnsname>` element to identify mentions of namespace names, for example, "http://dita.oasis-open.org/architecture/2005/". This element enables more precise semantic searching of the DITA source; it also enables distinct formatting. This element is part of the XML mention domain.

**Inheritance**

+ topic/keyword markup-d/markupname xml-d/xmlnsname

**Example**

The namespace name for XHTML is `<xmlnsname>http://www.w3.org/1999/xhtml</xmlnsname>`.

**Attributes**

The following attributes are available on this element: Universal attribute group (see page 650), `outputclass` (see page 0), and `@keyref`. 
4.10.7.10.7 <xmlpi>

Use the `<xmlpi>` element to identify mentions of processing instruction names, for example "<?piname pi value?>"). This element enables more precise semantic searching of the DITA source; it also enables distinct formatting. This element is part of the XML mention domain.

**Inheritance**

+ topic/keyword markup-d/markupname xml-d/xmlpi

**Example**

While DITA does not define any processing instructions, some applications may use some DocBook processing instructions, such as `<xmlpi>dbhtmlbgcolor</xmlpi>`.

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 687), and `@keyref`.

4.10.7.11 xNAL domain elements

The xNAL domain elements represent a subset of the Extensible Name and Address Standard. It is used to encode information about the author or authors of DITA information. The domain can be included in any DITA document type shell that requires additional metadata for names and addresses, although the implementations provided by OASIS only include it in the bookmap document type.

4.10.7.11.1 <authorinformation>

The `<authorinformation>` element contains detailed information about the author or authoring organization.

**Inheritance**

+ topic/author xnal-d/authorinformation

**Example**

```
<authorinformation>
  <personinfo>
    <namedetails>
      <personname>
        <firstname>Derek</firstname>
        <middlename>L.</middlename>
        <lastname>Singleton</lastname>
        <generationidentifier>Jr.</generationidentifier>
        <otherinfo>noted psychologist</otherinfo>
      </personname>
    </namedetails>
    <addressdetails>
      <thoroughfare>123 Yellow Brick Road</thoroughfare>
      <locality>Emerald City</locality>
      <administrativearea>Kansas</administrativearea>
      <country>USA</country>
    </addressdetails>
    <contactnumbers>
      <contactnumber>123-555-4678</contactnumber>
    </contactnumbers>
    <emailaddresses>
      <emailaddress>wizard@example.org</emailaddress>
    </emailaddresses>
  </personinfo>
</authorinformation>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661) (with a narrowed definition for `@type`, given below), and `@keyref`.

@type
Describes the target of a reference. See *The type attribute* (see page 674) for detailed information on supported values and processing implications. Note that this differs from the `@type` attribute on many other DITA elements. Beginning with DITA 1.2, values in this attribute are not limited to a small number of choices; the following values were used in DITA 1.0 and DITA 1.1, and are also recognized for the `<author>` element (and its specializations):

- **creator**
  The primary or original author of the content.

- **contributor**
  An additional author who is not primary.

- **-dita-use-conref-target**
  See *Using the -dita-use-conref-target value* (see page 666) for more information.

### 4.10.7.11.2 `<addressdetails>`

The `<addressdetails>` element contains information about the address of the author or authoring group.

#### Inheritance

+ topic/ph xnal-d/addressdetails

##### Example

```
<personinfo>
  <namedetails>
    <firstname>Derek</firstname>
    <middlename>L.</middlename>
    <lastname>Singleton</lastname>
    <generationidentifier>Jr.</generationidentifier>
    <otherinfo>noted psychologist</otherinfo>
  </namedetails>
  <addressdetails>
    <thoroughfare>123 Yellow Brick Road</thoroughfare>
    <locality>Emerald City</locality>
    <administrativearea>Kansas</administrativearea>
    <country>USA</country>
  </addressdetails>
  <contactnumbers>
    <contactnumber>123-555-4678</contactnumber>
  </contactnumbers>
  <emailaddresses>
    <emailaddress>wizard@example.org</emailaddress>
  </emailaddresses>
</personinfo>
```

#### Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), `outputclass` (see page 0), and `@keyref`.

### 4.10.7.11.3 `<administrativearea>`

The `<administrativearea>` element contains information about a county, state, or province.

#### Inheritance

+ topic/ph xnal-d/administrativearea

##### Example

```
<addressdetails>
  <thoroughfare>123 Yellow Brick Road</thoroughfare>
  <locality>Emerald City</locality>
  <administrativearea>Kansas</administrativearea>
  <country>USA</country>
</addressdetails>
```
**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *@keyref.*

4.10.7.11.4 `<contactnumber>`

A `<contactnumber>` element contains the contact number of a person or organization, such as a telephone number, mobile phone number, or fax number.

**Inheritance**
+ topic/data xnal-d/contactnumber

**Example**

```xml
<personinfo>
  <namedetails><personname>
    <firstname>Derek</firstname>
    <middlename>L.</middlename>
    <lastname>Singleton</lastname>
    <generationidentifier>Jr.</generationidentifier>
    <otherinfo>noted psychologist</otherinfo>
  </personname></namedetails>
  <addressdetails>
    <thoroughfare>123 Yellow Brick Road</thoroughfare>
    <locality>Emerald City</locality>
    <administrativearea>Kansas</administrativearea>
    <country>USA</country>
  </addressdetails>
  <contactnumbers><contactnumber>123-555-4678</contactnumber></contactnumbers>
  <emailaddresses><emailaddress>wizard@example.org</emailaddress></emailaddresses>
</personinfo>
```

**Attributes**
The following attributes are available on this element: *Data element attributes group* (see page 659), *Link relationship attribute group* (see page 661), *Universal attribute group* (see page 650), and *outputclass* (see page 0).

4.10.7.11.5 `<contactnumbers>`

The `<contactnumbers>` element contains a list of telephone and fax numbers.

**Inheritance**
+ topic/data xnal-d/contactnumbers

**Example**

```xml
<personinfo>
  <namedetails><personname>
    <firstname>Derek</firstname>
    <middlename>L.</middlename>
    <lastname>Singleton</lastname>
    <generationidentifier>Jr.</generationidentifier>
    <otherinfo>noted psychologist</otherinfo>
  </personname></namedetails>
  <addressdetails>
    <thoroughfare>123 Yellow Brick Road</thoroughfare>
    <locality>Emerald City</locality>
    <administrativearea>Kansas</administrativearea>
    <country>USA</country>
  </addressdetails>
  <contactnumbers><contactnumber>123-555-4678</contactnumber></contactnumbers>
</personinfo>
```
Attributes
The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.7.11.6 <country>

The <country> element contains the name of a country.

Inheritance
+ topic/ph xnal-d/country

Example

```xml
<addressdetails>
  <thoroughfare>123 Yellow Brick Road</thoroughfare>
  <locality>Emerald City</locality>
  <administrativearea>Kansas</administrativearea>
  <country>USA</country>
</addressdetails>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and @keyref.

4.10.7.11.7 <emailaddress>

The <emailaddress> element contains an e-mail address.

Inheritance
+ topic/data xnal-d/emailaddress

Example

```xml
<personinfo>
  <namedetails>
    <personname>
      <firstname>Derek</firstname>
      <middlename>L.</middlename>
      <lastname>Singleton</lastname>
      <generationidentifier>Jr.</generationidentifier>
      <otherinfo>noted psychologist</otherinfo>
    </personname>
  </namedetails>
  <addressdetails>
    <thoroughfare>123 Yellow Brick Road</thoroughfare>
    <locality>Emerald City</locality>
    <administrativearea>Kansas</administrativearea>
    <country>USA</country>
  </addressdetails>
  <contactnumbers>
    <contactnumber>123-555-4678</contactnumber>
  </contactnumbers>
  <emailaddresses>
    <emailaddress>wizard@example.org</emailaddress>
  </emailaddresses>
</personinfo>
```
Attributes
The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.7.11.8 <emailaddresses>

The <emailaddresses> element contains a list of e-mail addresses.

Inheritance
+ topic/data xnal-d/emailaddresses

Example

```xml
<personinfo>
  <namedetails>
    <personname>
      <firstname>Derek</firstname>
      <middlename>L.</middlename>
      <lastname>Singleton</lastname>
      <generationidentifier>Jr.</generationidentifier>
      <otherinfo>noted psychologist</otherinfo>
    </personname>
    <addressdetails>
      <thoroughfare>123 Yellow Brick Road</thoroughfare>
      <locality>Emerald City</locality>
      <administrativearea>Kansas</administrativearea>
      <country>USA</country>
    </addressdetails>
    <contactnumbers>
      <contactnumber>123-555-4678</contactnumber>
    </contactnumbers>
    <emailaddresses>
      <emailaddress>wizard@example.org</emailaddress>
    </emailaddresses>
  </namedetails>
</personinfo>
```

Attributes
The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.7.11.9 <firstname>

The <firstname> element contains the person's first name.

Inheritance
+ topic/data xnal-d/firstname

Example

```xml
<namedetails>
  <personname>
    <honorific>Dr.</honorific>
    <firstname>Derek</firstname>
    <middlename>L.</middlename>
    <lastname>Singleton</lastname>
    <generationidentifier>Jr.</generationidentifier>
    <otherinfo>noted psychologist</otherinfo>
  </personname>
</namedetails>
```
Attributes

The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.7.11.10 <generationidentifier>

The <generationidentifier> element contains information about the person's generation, such as: Jr, III, or VIII.

Inheritance

+ topic/data xnal-d/generationidentifier

Example

<namedetails><personname>
  <firstname>Derek</firstname>
  <middlename>L.</middlename>
  <lastname>Singleton</lastname>
  <generationidentifier>Jr.</generationidentifier>
  <otherinfo>noted psychologist</otherinfo>
</personname></namedetails>

Attributes

The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.7.11.11 <honorific>

The <honorific> element contains the person's title, such as: Dr., Mr., Ms., or HRH.

Inheritance

+ topic/data xnal-d/honorific

Example

<namedetails><personname>
  <honorific>Dr.</honorific>
  <firstname>Derek</firstname>
  <middlename>L.</middlename>
  <lastname>Singleton</lastname>
  <generationidentifier>Jr.</generationidentifier>
  <otherinfo>noted psychologist</otherinfo>
</personname></namedetails>

Attributes

The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.7.11.12 <lastname>

The <lastname> element contains the person's last name.

Inheritance

+ topic/data xnal-d/lastname
Example

<namedetails><personname>
  <honorific>Dr.</honorific>
  <firstname>Derek</firstname>
  <middlename>L.</middlename>
  <lastname>Singleton</lastname>
  <generationidentifier>Jr.</generationidentifier>
  <otherinfo>noted psychologist</otherinfo>
</personname></namedetails>

Attributes

The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.7.11.13 <locality>

The <locality> element contains information about the city and postal or ZIP code. It can contain the information directly, or by acting as a wrapper for <localityname> and <postalcode>.

Inheritance

+ topic/ph xnal-d/locality

Example

 ADDRESS DETAILS

<addressdetails>
  <thoroughfare>123 Yellow Brick Road</thoroughfare>
  <locality>
    <localityname>Emerald City</localityname>
    <postalcode>66780</postalcode>
  </locality>
  <administrativearea>Kansas</administrativearea>
  <country>USA</country>
</addressdetails>

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and @keyref.

4.10.7.11.14 <localityname>

The <localityname> element contains the name of the locality or city.

Inheritance

+ topic/ph xnal-d/localityname

Example

 ADDRESS DETAILS

<addressdetails>
  <thoroughfare>123 Yellow Brick Road</thoroughfare>
  <locality>
    <localityname>Emerald City</localityname>
    <postalcode>66780</postalcode>
  </locality>
  <administrativearea>Kansas</administrativearea>
  <country>USA</country>
</addressdetails>
Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0 ), and *@keyref*.

4.10.7.11.15 <middlename>

The `<middlename>` element contains the person's middle name or initial.

Inheritance

+ topic/data xnal-d/middlename

Example

```xml
<namedetails><personname>
  <honorific>Dr.</honorific>
  <firstname>Derek</firstname>
  <middlename>L.</middlename>
  <lastname>Singleton</lastname>
  <generationidentifier>Jr.</generationidentifier>
  <otherinfo>noted psychologist</otherinfo>
</personname></namedetails>
```

Attributes

The following attributes are available on this element: *Data element attributes group* (see page 659), *Link relationship attribute group* (see page 661), *Universal attribute group* (see page 650), and *outputclass* (see page 0 ).

4.10.7.11.16 <namedetails>

The `<namedetails>` element contains information about the name of the author or the authoring organization.

Inheritance

+ topic/data xnal-d/namedetails

Example

```xml
<personinfo>
  <namedetails><personname>
    <firstname>Derek</firstname>
    <middlename>L.</middlename>
    <lastname>Singleton</lastname>
    <generationidentifier>Jr.</generationidentifier>
    <otherinfo>noted psychologist</otherinfo>
  </personname></namedetails>
  <addressdetails>
    <thoroughfare>123 Yellow Brick Road</thoroughfare>
    <locality>Emerald City</locality>
    <administrativearea>Kansas</administrativearea>
    <country>USA</country>
  </addressdetails>
  <contactnumbers><contactnumber>123-555-4678</contactnumber></contactnumbers>
  <emailaddresses><emailaddress<wizard@example.org</emailaddress></emailaddresses>
</personinfo>
```

Attributes

The following attributes are available on this element: *Data element attributes group* (see page 659), *Link relationship attribute group* (see page 661), *Universal attribute group* (see page 650), and *outputclass* (see page 0 ).
4.10.7.11.17 <organizationinfo>

The `<organizationinfo>` element contains detailed information about an authoring organization.

Inheritance

+ `topic/data xnal-d/organizationinfo`

Example

```xml
<organizationinfo>
  <namedetails>
    <organizationnamedetails>
      <organizationname>WizardWorks, Inc.</organizationname>
      <otherinfo>'Best wizard in Oz'</otherinfo>
    </organizationnamedetails>
  </namedetails>
  <addressdetails>
    <thoroughfare>123 Yellow Brick Road</thoroughfare>
    <locality>Emerald City</locality>
    <administrativearea>Kansas</administrativearea>
    <country>USA</country>
  </addressdetails>
  <contactnumbers>
    <contactnumber>123-555-4678</contactnumber>
  </contactnumbers>
  <emailaddresses>
    <emailaddress>wizard@example.org</emailaddress>
  </emailaddresses>
  <urls>
    <url>www.wizardworks.example.org</url>
  </urls>
</organizationinfo>
```

Attributes

The following attributes are available on this element: `Data element attributes group` (see page 659), `Link relationship attribute group` (see page 661), `Universal attribute group` (see page 650), and `outputclass` (see page 0).

4.10.7.11.18 <organizationname>

The `<organizationname>` element contains name information about the authoring organization.

Inheritance

+ `topic/ph xnal-d/organizationname`

Example

```xml
<organizationinfo>
  <namedetails>
    <organizationnamedetails>
      <organizationname>WizardWorks, Inc.</organizationname>
      <otherinfo>'Best wizard in Oz'</otherinfo>
    </organizationnamedetails>
  </namedetails>
  <addressdetails>
    <thoroughfare>123 Yellow Brick Road</thoroughfare>
    <locality>Emerald City</locality>
    <administrativearea>Kansas</administrativearea>
    <country>USA</country>
  </addressdetails>
  <contactnumbers>
    <contactnumber>123-555-4678</contactnumber>
  </contactnumbers>
  <emailaddresses>
    <emailaddress>wizard@example.org</emailaddress>
  </emailaddresses>
  <urls>
    <url>www.wizardworks.example.org</url>
  </urls>
</organizationinfo>
```

Attributes

The following attributes are available on this element: `Universal attribute group` (see page 650), `outputclass` (see page 0), and `@keyref.`
4.10.7.11.19 <organizationnamedetails>

The `<organizationnamedetails>` element contains information about the name of an authoring organization.

*Inheritance*

+ topic/ph xnal-d/organizationnamedetails

*Example*

```xml
<organizationinfo>
  <namedetails>
    <organizationname>WizardWorks, Inc.</organizationname>
    <otherinfo>'Best wizard in Oz'</otherinfo>
  </namedetails>
  <addressdetails>
    <thoroughfare>123 Yellow Brick Road</thoroughfare>
    <locality>Emerald City</locality>
    <administrativearea>Kansas</administrativearea>
    <country>USA</country>
  </addressdetails>
  <contactnumbers>
    <contactnumber>123-555-4678</contactnumber>
  </contactnumbers>
  <emailaddresses>
    <emailaddress>wizard@example.org</emailaddress>
  </emailaddresses>
  <urls>
    <url>www.wizardworks.example.org</url>
  </urls>
</organizationinfo>
```

*Attributes*

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0 ), and *@keyref*.

4.10.7.11.20 <otherinfo>

The `<otherinfo>` element contains other name information about the author or authoring organization.

*Inheritance*

+ topic/data xnal-d/otherinfo

*Example*

```xml
<organizationinfo>
  <namedetails>
    <organizationname>WizardWorks, Inc.</organizationname>
    <otherinfo>'Best wizard in Oz'</otherinfo>
  </namedetails>
  <addressdetails>
    <thoroughfare>123 Yellow Brick Road</thoroughfare>
    <locality>Emerald City</locality>
    <administrativearea>Kansas</administrativearea>
    <country>USA</country>
  </addressdetails>
  <contactnumbers>
    <contactnumber>123-555-4678</contactnumber>
  </contactnumbers>
  <emailaddresses>
    <emailaddress>wizard@example.org</emailaddress>
  </emailaddresses>
  <urls>
    <url>www.wizardworks.example.org</url>
  </urls>
</organizationinfo>
```

*Attributes*

The following attributes are available on this element: *Data element attributes group* (see page 659), *Link relationship attribute group* (see page 661), *Universal attribute group* (see page 650), and *outputclass* (see page 0 ).
4.10.7.11.21 <personinfo>

The <personinfo> element is a wrapper containing all relevant data about a person, including name, address, and contact information.

Inheritance
+ topic/data xnal-d/personinfo

Example

```
<personinfo>
  <namedetails>
    <personname>
      <firstname>Derek</firstname>
      <middlename>L.</middlename>
      <lastname>Singleton</lastname>
      <generationidentifier>Jr.</generationidentifier>
      <otherinfo>noted psychologist</otherinfo>
    </personname>
  </namedetails>
  <addressdetails>
    <thoroughfare>123 Yellow Brick Road</thoroughfare>
    <locality>Emerald City</locality>
    <administrativearea>Kansas</administrativearea>
    <country>USA</country>
  </addressdetails>
  <contactnumbers>
    <contactnumber>123-555-4678</contactnumber>
  </contactnumbers>
  <emailaddresses>
    <emailaddress>wizard@example.org</emailaddress>
  </emailaddresses>
</personinfo>
```

Attributes

The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.7.11.22 <personname>

The <personname> element contains name information about the author.

Inheritance
+ topic/data xnal-d/personname

Example

```
<personinfo>
  <namedetails>
    <personname>
      <firstname>Derek</firstname>
      <middlename>L.</middlename>
      <lastname>Singleton</lastname>
      <generationidentifier>Jr.</generationidentifier>
      <otherinfo>noted psychologist</otherinfo>
    </personname>
  </namedetails>
  <addressdetails>
    <thoroughfare>123 Yellow Brick Road</thoroughfare>
    <locality>Emerald City</locality>
    <administrativearea>Kansas</administrativearea>
    <country>USA</country>
  </addressdetails>
  <contactnumbers>
    <contactnumber>123-555-4678</contactnumber>
  </contactnumbers>
  <emailaddresses>
    <emailaddress>wizard@example.org</emailaddress>
  </emailaddresses>
</personinfo>
```
**Attributes**

The following attributes are available on this element: *Data element attributes group* (see page 659), *Link relationship attribute group* (see page 661), *Universal attribute group* (see page 650), and *outputclass* (see page 0).

4.10.7.11.23 `<postalcode>`

The `<postalcode>` element contains information about the postal code or the ZIP code.

**Inheritance**

+ topic/ph xnal-d/postalcode

**Example**

```xml
<addressdetails>
  <thoroughfare>123 Yellow Brick Road</thoroughfare>
  <locality>
    <localityname>Emerald City</localityname>
    <postalcode>66780</postalcode>
  </locality>
  <administrativearea>Kansas</administrativearea>
  <country>USA</country>
</addressdetails>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and `@keyref`.

4.10.7.11.24 `<thoroughfare>`

The `<thoroughfare>` element contains information about the thoroughfare - for example, the street, avenue, or boulevard - on which an address is located.

**Inheritance**

+ topic/ph xnal-d/thoroughfare

**Example**

```xml
<addressdetails>
  <thoroughfare>123 Yellow Brick Road</thoroughfare>
  <locality>
    <localityname>Emerald City</localityname>
    <postalcode>66780</postalcode>
  </locality>
  <administrativearea>Kansas</administrativearea>
  <country>USA</country>
</addressdetails>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and `@keyref`. 
4.10.7.11.25 <url>

The <url> element contains a Uniform Resource Locator (URL), such as a person's or company's internet address.

Inheritance
+ topic/data xnal-d/url

Example

```xml
<organizationinfo>
  <namedetails>
    <organizationnamedetails>
      <organizationname>WizardWorks, Inc.</organizationname>
      <otherinfo>'Best wizard in Oz'</otherinfo>
    </organizationnamedetails>
  </namedetails>
  <thoroughfare>123 Yellow Brick Road</thoroughfare>
  <locality>Emerald City</locality>
  <administrativearea>Kansas</administrativearea>
  <country>USA</country>
</addressdetails>
<contactnumbers><contactnumber>123-555-4678</contactnumber></contactnumbers>
<emailaddresses><emailaddress>wizard@example.org</emailaddress></emailaddresses>
<urls><url>www.wizardworks.example.org</url></urls>
</organizationinfo>
```

Attributes

The following attributes are available on this element: Data element attributes group (see page 659), Link relationship attribute group (see page 661), Universal attribute group (see page 650), and outputclass (see page 0).

4.10.7.11.26 <urls>

The <urls> element contains a list of Uniform Resource Locators (URLs).

Inheritance
+ topic/data xnal-d/urls

Example

```xml
<organizationinfo>
  <namedetails>
    <organizationnamedetails>
      <organizationname>WizardWorks, Inc.</organizationname>
      <otherinfo>'Best wizard in Oz'</otherinfo>
    </organizationnamedetails>
  </namedetails>
  <thoroughfare>123 Yellow Brick Road</thoroughfare>
  <locality>Emerald City</locality>
  <administrativearea>Kansas</administrativearea>
  <country>USA</country>
</addressdetails>
<contactnumbers><contactnumber>123-555-4678</contactnumber></contactnumbers>
<emailaddresses><emailaddress>wizard@example.org</emailaddress></emailaddresses>
<urls><url>www.wizardworks.example.org</url></urls>
</organizationinfo>
```
4.11 Learning and training elements

4.11.1 Learning and training topic elements

Use the learning and training topic types to provide the instructional content, according to the needs identified by the learning goals and objectives.

4.11.1.1 <learningOverview>

A Learning Overview topic identifies the learning objectives, includes other information helpful to the learner, such as prerequisites, duration, intended audience, and can include information and strategies that seeks to gain attention and stimulate recall of prior learning.

Inheritance
- topic/topic learningBase/learningBase learningOverview/learningOverview

Example

```xml
<learningOverview id="understanding_the_basics" xml:lang="en-us">
  <title>Overview: Understanding the basics</title>
  <shortdesc>Mail basics start from the inbox, viewing and opening messages you receive, and moving them to appropriate mail folders for easy access and retrieval.</shortdesc>
  <lcAudience>The intended audience includes new users of the company email system and anyone wanting a refresher on the basic features.</lcAudience>
  <lcDuration>
    <title>Expected duration</title>
    <lcTime value="00:30">It should take you no more than 30 minutes to complete this module.</lcTime>
  </lcDuration>
  <lcObjectives>
    <lcObjectivesStem>When you complete this lesson, you'll know how to perform the following mail basics:</lcObjectivesStem>
    <lcObjectivesGroup>
      <lcObjective>Viewing the inbox</lcObjective>
      <lcObjective>Opening a message</lcObjective>
      <lcObjective>Moving messages to a folder</lcObjective>
    </lcObjectivesGroup>
  </lcObjectives>
</learningOverviewbody>
</learningOverview>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below), Architectural attribute group (see page 654), and outputclass (see page 0 ).

@id (REQUIRED)

An anchor point. This ID is usually required as part of the @href or @conref syntax when cross referencing or reusing content within the topic; it also enables <topicref> elements in DITA maps to optionally reference a specific topic within a DITA document. This attribute is defined with the XML Data Type ID.
4.11.1.2 <learningOverviewbody>

The <learningOverviewbody> element is the main body-level element in a <learningOverview> topic. A <learningOverviewbody> has a very specific structure, with the following elements in this order: <lcIntro>, <lcAudience>, <lcDuration>, <lcPrereqs>, <lcObjectives>, <lcResources>, followed by zero or more <section> elements. Each of the <learningOverviewbody> sections are optional.

Inheritance
- topic/body learningBase/learningBasebody learningOverview/learningOverviewbody

Example
See learningOverview.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.1.3 <learningContent>

A Learning Content topic provides the learning content itself, and enables direct use of content from DITA task, concept, and reference topics, as well as additional content of any topic type that supports specific objectives declared in the Learning Overview topic type.

A learning content topic comprises a set of self-contained content about a single terminal learning objective supported by zero or more enabling learning objectives.

A learning content topic should be rendered as single result component (for example, a single HTML page) when it has subordinate topics. The subordinate topics either can be direct child elements or associated through a map. This rendering result can be requested by specifying a value of "to-content" for the @chunk attribute of <topicref> elements that reference <learningContent> topics. This is the default value for the <learningContentRef> element that is provided by the learning map domain.

Inheritance
- topic/topic learningBase/learningBase learningContent/learningContent

Example

```xml
<learningContent id="learningcontent">
  <title>Your Mail Inbox</title>
  <learningContentbody>
    <lcObjectives>
      <lcObjectivesStem>When you complete this topic, you'll understand:</lcObjectivesStem>
      <lcObjectivesGroup>
        <lcObjective>How to use the mail inbox.</lcObjective>
      </lcObjectivesGroup>
    </lcObjectives>
  </learningContentbody>
</learningContent>
```

DITA Version 1.3 Specification

TBD

Copyright © OASIS Open 2005, 2014. All Rights Reserved.
**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of @id, given below), *Architectural attribute group* (see page 654), and *outputclass* (see page 0).

@id (REQUIRED)

An anchor point. This ID is usually required as part of the @href or @conref syntax when cross referencing or reusing content within the topic; it also enables <topicref> elements in DITA maps to optionally reference a specific topic within a DITA document. This attribute is defined with the XML Data Type ID.

### 4.11.1.4 <learningContentbody>

The <learningContentbody> element is the main body-level element in a <learningContent> topic.

**Inheritance**

- topic/body learningBase/learningBasebody learningContent/learningContentbody

**Example**

See learningContent.

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

### 4.11.1.5 <learningSummary>

A Learning Summary topic recaps and provides context for the achievement or accomplishment of learning objectives, provides guidance to reinforce learning and long-term memory, and may pose questions to enhance encoding and verification of the learning content.

**Inheritance**

- topic/topic learningBase/learningBasebody learningSummary/learningSummary

**id="learningSummary_ex"**

**Example**

```xml
<learningSummary id="learningsummary">
  <title>Summary: Understanding mail basics</title>
  <learningSummarybody>
    <lcObjectives>
      <lcObjectivesStem>You now know how to perform the following mail basics:</lcObjectivesStem>
      <lcObjectivesGroup>
        <lcObjective>Viewing the inbox</lcObjective>
        <lcObjective>Opening a message</lcObjective>
        <lcObjective>Moving messages to a folder</lcObjective>
      </lcObjectivesGroup>
    </lcObjectives>
  </learningSummarybody>
</learningSummary>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of @id, given below), *Architectural attribute group* (see page 654), and *outputclass* (see page 0).
@id (REQUIRED)
An anchor point. This ID is usually required as part of the @href or @conref syntax when cross referencing or reusing content within the topic; it also enables <topicref> elements in DITA maps to optionally reference a specific topic within a DITA document. This attribute is defined with the XML Data Type ID.

4.11.1.6 <learningSummarybody>
The <learningSummarybody> element is the main body-level element in a <learningSummary> topic.

Inheritance
- topic/body learningBase/learningBasebody learningSummary/learningSummarybody

Example
See learningSummary.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.1.7 <learningAssessment>
A Learning Assessment topic presents questions or interactions that measure progress, encourage recollection, and stimulate reinforcement of the learning content, and can be presented before the content as a pre-assessment or as a post-assessment test. The interactions use a sub-set of the Question-Test Interoperability (QTI) specification, implemented as a DITA domain specialization.

Inheritance
- topic/topic learningBase/learningBase learningAssessment/learningAssessment

Example

```xml
<learningAssessment id="testAssess">
  <title>Certification Test</title>
  <shortdesc>Pass this test, and you are a certified genius.</shortdesc>
  <learningAssessmentbody>
    <lcIntro>Here's your test, folks. Good luck!</lcIntro>
    <lcInteraction>
      <lcSingleSelect id="asdf">
        <title>Multiple Choice - IEEE standards trivia</title>
        <lcQuestion>Which one of the listed standards committees is responsible for developing the token ring specification?</lcQuestion>
        <lcAnswerOptionGroup>
          <lcAnswerOption>
            <lcAnswerContent>IEEE 802.3</lcAnswerContent>
          </lcAnswerOption>
          <lcAnswerOption>
            <lcAnswerContent>IEEE 802.5</lcAnswerContent>
            <lcCorrectResponse/>
          </lcAnswerOption>
          <lcAnswerOption>
            <lcAnswerContent>IEEE 802.6</lcAnswerContent>
          </lcAnswerOptionGroup>
        </lcSingleSelect>
        <lcInteraction>
          <title>Summary</title>
          <shortdesc>You are now certified.</shortdesc>
        </lcInteraction>
      </lcInteraction>
    </lcInteraction>
  </learningAssessmentbody>
</learningAssessment>
```
Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below), Architectural attribute group (see page 654), and outputclass (see page 0).

@id (REQUIRED)

An anchor point. This ID is usually required as part of the @href or @conref syntax when cross referencing or reusing content within the topic; it also enables <topicref> elements in DITA maps to optionally reference a specific topic within a DITA document. This attribute is defined with the XML Data Type ID.

4.11.1.9 <learningPlan>

A Learning Plan topic describes learning needs and goals, instructional design models, task analyses, learning taxonomies, and other information necessary to the lesson planning process.

Inheritance

- topic/topic learningBase/learningBase learningPlan/learningPlan

Example

See learningAssessment.

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.1.8 <learningAssessmentbody>

The <learningAssessmentbody> element is the main body-level element in a <learningAssessment> topic.

Inheritance

- topic/body learningBase/learningBasebody learningAssessment/learningAssessmentbody

Example

See learningAssessment.
<lcPlanObjective>Describe the related plan objective for this gap.</lcPlanObjective>
<lcJtaItem>Provide job task analysis information about the gap.</lcJtaItem>
<lcGapItemDelta>Describe the gap item.</lcGapItemDelta>
<title>and another GapItem for you to consider</title>
<lcPlanObjective>Describe the related plan objective for this gap.</lcPlanObjective>
<lcJtaItem>Provide job task analysis information about the gap.</lcJtaItem>
<lcGapItemDelta>Describe the gap item.</lcGapItemDelta>
</lcGapItem>
</lcGapAnalysis>
<lcIntervention>
<title>lcIntervention</title>
<lcInterventionItem>
<title>lcInterventionItem</title>
<lcLearnStrat>Describe the manner in which the learning content will be instructed. This should be a high level design that applies instructional-design theories and models.</lcLearnStrat>
<lcPlanObjective>Describe the objective to be addressed by a gap analysis or intervention.</lcPlanObjective>
<lcAssessment>Describe assessment plans.</lcAssessment>
<lcDelivery>Describe the delivery method for this learning content.</lcDelivery>
</lcInterventionItem>
<title>another lcInterventionItem (and more if you want 'em)</title>
<lcLearnStrat>Describe the manner in which the learning content will be instructed. This should be a high level design that applies instructional-design theories and models.</lcLearnStrat>
<lcPlanObjective>Describe the objective to be addressed by a gap analysis or intervention.</lcPlanObjective>
<lcAssessment>Describe assessment plans.</lcAssessment>
<lcDelivery>Describe the delivery method for this learning content.</lcDelivery>
</lcInterventionItem>
</lcIntervention>
<lcTechnical>
<title>lcTechnical</title>
<lcLMS>
<title>lcLMS</title>
<p>Describe characteristics of the learning management system to be used.</p>
</lcLMS>
<lcHandouts>
<title>lcHandouts</title>
<p>Describe any handouts.</p>
</lcHandouts>
<lcClassroom>
<title>lcClassroom</title>
<p>Provide information about the classroom, if used.</p>
</lcClassroom>
<lcOJT>
<title>lcOJT</title>
<p>Describe the on-the-job training.</p>
</lcOJT>
<lcConstraints>
<title>lcConstraints</title>
<p>Describe any constraints.</p>
</lcConstraints>
<lcW3C>
<title>lcW3C</title>
<p>Provide information about any related web standards.</p>
</lcW3C>
<lcPlayers>
<title>lcPlayers</title>
<p>Detail the tools and plugins used for time-sequenced display at runtime.</p>
</lcPlayers>
<lcViewers>
<title>lcViewers</title>
<p>Detail the viewers used for time-sequenced display at runtime.</p>
</lcViewers>
<lcResolution>
<title>lcResolution</title>

DITA Version 1.3 Specification  TBD
Copyright © OASIS Open 2005, 2014. All Rights Reserved.  Page 555 of 747
Provide a list of related resources and information about them, such as related articles or samples on the web.

</lcResolution>
</lcFileSizeLimitations>
</lcDownloadTime>
</lcFileSizeLimitations>
</lcDownloadTime>
</lcSecurity>
</lcTechnical>
</learningPlanbody>
</learningPlan>

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below), Architectural attribute group (see page 654), and outputclass (see page 0).

@id (REQUIRED)

An anchor point. This ID is usually required as part of the @href or @conref syntax when cross referencing or reusing content within the topic; it also enables <topicref> elements in DITA maps to optionally reference a specific topic within a DITA document. This attribute is defined with the XML Data Type ID.

4.11.1.10 <learningPlanbody>

The <learningPlanbody> element is the main body-level element in a <learningPlan> topic.

Inheritance

- topic/topic learningBase/learningBase learningPlan/learningPlanbody

Example

See learningPlan.

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.1.11 <learningBase>

The <learningBase> topic type is not used to deliver any actual learning content, but instead provides a set of common elements for use in the other more specific learning content types: <learningOverview>, <learningContent>, <learningSummary>, <learningAssessment>, and <learningPlan>.

Inheritance

- topic/topic learningBase/learningBase
**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of @id, given below), *Architectural attribute group* (see page 654), and *outputclass* (see page 0).

@id (REQUIRED)

An anchor point. This ID is usually required as part of the @href or @conref syntax when cross referencing or reusing content within the topic; it also enables <topicref> elements in DITA maps to optionally reference a specific topic within a DITA document. This attribute is defined with the XML Data Type ID.

### 4.11.1.12 <learningBasebody>

The <learningBasebody> element is the main body-level element in a <learningBase> topic. The <learningBase> topic provides a set of base elements for use in the other specialized learning types. It is not generally intended for creating actual content. As such, each of the body sections in <learningBase> are optional.

**Inheritance**

- topic/body learningBase/learningBasebody

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

### 4.11.2 Learning and training map types

Use the learning and training map types to aggregate learning content into individual learning objects, groups of learning objects, or higher-level aggregations of learning groups and objects, according to the needs identified by the learning goals and objectives.

#### 4.11.2.1 <learningGroupMap>

Use a <learningGroupMap> to create standalone aggregations of <learningObject> and <learningGroup> elements, as well as overviews, summaries, and assessments, according to the needs identified by the learning goals and objectives.

**Description and Semantics**

The <learningGroupMap> is a map specialization which supports the DITA Learning and Training Map domain. It is used to aggregate <learningObject> and <learningGroup> elements, together with their supporting files and assessments, into a single standalone map type. The <learningGroupMap> references one and only one <learningGroup>. Any <topicref> elements that occur before the <learningGroup> are intended to be resource-only references such as key definitions.

Recursion is provided to support complex aggregations of learning content and can be used as necessary to support the creation of complex learning structures, such as learning units, sections, and courses.

**Inheritance**

- map/map learningGroupMap/learningGroupMap

**Example**

```xml
<learningGroupMap id="learningGroupMap1">
  <title>Sample learningGroup map</title>
  <learningGroup>
```

DITA Version 1.3 Specification

Copyright © OASIS Open 2005, 2014. All Rights Reserved.

Page 557 of 747
Attributes

The following attributes are available on this element: **Universal attribute group** (see page 650) (with a narrowed definition of @id, given below), **Attributes common to many map elements** (see page 655), **Architectural attribute group** (see page 654), **outputclass** (see page 0), and the attributes defined below. This element also uses @type, @scope, and @format from **Link relationship attribute group** (see page 661).

- **@id**
  
  Allows an ID to be specified for the map. Note that maps do not require IDs (unlike topics), and the map ID is not included in references to elements within a map. This attribute is defined with the XML Data Type ID.

- **@anchorref**
  
  Identifies a location within another map file where this map will be anchored at runtime. Resolution of the map is deferred until the final step in the delivery of any rendered content. For example, anchorref="map1.ditamap/a1" causes this map to be pulled into the location of the anchor point "a1" inside map1.ditamap when map1.ditamap is rendered for delivery. This attribute is defined with the XML Data Type CDATA.

- **@title**
  
  An identifying title for the map. May be used or ignored, depending on the capabilities of the display mechanism. Note that beginning with DITA 1.1, the map can include a <title> element, which is preferred over the @title attribute. This attribute is defined with the XML Data Type CDATA.

4.11.2.2 <learningObjectMap>

Use a <learningObjectMap> to create standalone aggregations of <learningContent> together with its supporting plan, overview, summary, and assessments, according to the needs identified by the learning goals and objectives. The <learningObjectMap> references one and only one <learningObject>. Any <topicref> elements that occur before the <learningObject> are intended to be resource-only references such as key definitions.

Description and Semantics

The <learningObjectMap> is a map specialization which supports the DITA Learning and Training Map domain. It is used to aggregate <learningContent>, together with its supporting files and assessments, into a single standalone map type.

Inheritance

- map/map learningObjectMap/learningObjectMap

Example

<learningObjectMap id="leaningObjectMap1">
  <title>Sample learningObject map</title>
  <learningobject>
    <learningPlanRef href="learningPlan.dita" scope="local" type="learningPlan" format="dita"/>
    <learningOverviewRef href="learningOverview.dita" scope="local" type="learningOverview" format="dita"/>
  </learningobject>
</learningObjectMap>
Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below), Attributes common to many map elements (see page 655), Architectural attribute group (see page 654), outputclass (see page 60), and the attributes defined below. This element also uses @type, @scope, and @format from Link relationship attribute group (see page 661).

@id

Allows an ID to be specified for the map. Note that maps do not require IDs (unlike topics), and the map ID is not included in references to elements within a map. This attribute is defined with the XML Data Type ID.

@anchorref

Identifies a location within another map file where this map will be anchored at runtime. Resolution of the map is deferred until the final step in the delivery of any rendered content. For example, anchorref="map1.ditamap/a1" causes this map to be pulled into the location of the anchor point "a1" inside map1.ditamap when map1.ditamap is rendered for delivery. This attribute is defined with the XML Data Type CDATA.

@title

An identifying title for the map. May be used or ignored, depending on the capabilities of the display mechanism. Note that beginning with DITA 1.1, the map can include a <title> element, which is preferred over the @title attribute. This attribute is defined with the XML Data Type CDATA.

4.11.3 Learning and training map domain elements

Use the learning and training map domain to organize groups of topics as learning objects. Defined as a map domain, these elements are available for use within any DITA map, not just a learning-specific DITA map. For example, you could include learning objects in a DITA generic map or in a DITA bookmark.

4.11.3.1 <learningGroup>

Use a <learningGroup> to structure learning objects into higher-level organizations, such as course-level, module-level, or lesson-level.

In addition to organizing <learningObject> elements, a <learningGroup> may include topic references to <learningPlan>, <learningOverview>, and <learningSummary> topics, and may also include references to <learningAssessment> topics. A <learningGroup> can also contain other <learningGroup> elements, allowing you to organize learning content at course, module, or other higher levels of hierarchy.

Inheritance

+ map/topicref learningmap-d/learningGroup

Example

```xml
<learningGroup href="course_top_Overview.dita" type="learningOverview">
  <!-- a learningGroup can appear anywhere in a map hierarchy but always follows a consistent information pattern -->
  <!-- a course-level learning group -->
</learningGroup>
```
<learningPlanRef href="course_plan_topic.dita" type="learningPlan" />
<learningOverviewRef href="course_Overview.dita" type="learningOverview"/>

<!-- module-level learning groups -->
<learningGroup href="module1_Overview.dita" type="learningOverview">
  <!-- This module has two lesson-level groups of learning objects -->
  <learningOverviewRef href="lesson1_Overview.dita" type="learningOverview"/>
  <learningOverviewRef href="lesson2_Overview.dita" type="learningOverview"/>
  <learningObject collection-type="sequence" href="lo1_Overview.dita" type="learningOverview">
  <!-- . . . -->
  </learningObject>
  <learningObject collection-type="sequence" href="lo2_Overview.dita" type="learningOverview">
  <!-- . . . -->
  </learningObject>
  <learningAssessmentRef href="lesson1_Assessment.dita" type="learningAssessment"/>
  <learningSummaryRef href="lesson1_Summary.dita" type="learningSummary"/>
</learningGroup>

<learningGroup href="module2_Overview.dita" type="learningOverview">
  <learningOverviewRef href="lesson1_Overview.dita" type="learningOverview"/>
  <learningObject collection-type="sequence" href="loA_Overview.dita" type="learningOverview">
  <!-- . . . -->
  </learningObject>
  <learningObject collection-type="sequence" href="loB_Overview.dita" type="learningOverview">
  <!-- . . . -->
  </learningObject>
  <learningObject collection-type="sequence" href="loC_Overview.dita" type="learningOverview">
  <!-- . . . -->
  </learningObject>
  <learningAssessmentRef href="module2_Assessment.dita" type="learningAssessment"/>
  <learningSummaryRef href="module2_Summary.dita" type="learningSummary"/>
</learningGroup>

<learningAssessmentRef href="course_Assessment.dita" type="learningAssessment"/>
<learningSummaryRef href="course_Summary.dita" type="learningSummary"/>
</map>

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), Attributes common to many map elements (see page 655), Topicref element attributes group (see page 663), outputclass (see page 0 ), @keys, and @keyref.

@href
A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

4.11.3.2 <learningObject>

learningObject organizes topic references to DITA learning topics as a learning object.

A <learningObject> follows a specific sequence of topic references to learning content, starting with a learning plan, then a learning overview or learning pre-assessment, one or more learning content topics, a learning summary, and one or more learning post-assessment topics.

By default, the topic references in a <learningObject> are expected to use the learning-specific topic types. However, the @href can point to content of any type that you want to structure as a learning object. In this way, you can structure any existing DITA content as a learning object, and then take advantage of the learning-specific processing available for building and packaging that content for delivery in a learning system or learning context.

Inheritance

+ map/topicref learningmap-d/learningObject

Example

Comment by WEK
Added example of using learningContentComponentRef within learningContentRef.

```xml
<map title="learningObject_example">
  <!-- learningObject clusters can appear anywhere in a map hierarchy but always follow a consistent information pattern within the LO -->
  <learningObject collection-type="sequence" href="topOverview.dita" type="learningOverview">
    <learningPlanRef href="testlearningPlan.dita" type="learningPlan"/>
    <learningOverviewRef href="testlearningOverview.dita" type="learningOverview"/>
    <learningContentRef href="testlearningContent.dita" type="learningContent">
      <learningContentComponentRef href="termA.dita" type="glossentry"/>
      <learningContentComponentRef href="termB.dita" type="glossentry"/>
    </learningContentRef>
    <learningAssessmentRef href="testlearningAssess.dita" type="learningAssessment"/>
    <learningSummaryRef href="testlearningSummary.dita" type="learningSummary"/>
  </learningObject>
</map>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with a narrowed definition of @href, given below), Attributes common to many map elements (see page 655), Topicref elements attributes group (see page 663), outputclass (see page 0 ), keys, and keyref.

@href

A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.
4.11.3.3 <learningObjectMapRef>

Use a <learningObjectMapRef> element to include a map reference to a learning object map as part of a DITA learning map, learning bookmap, or learning group map.

Description and Semantics

The <learningObjectMapRef> is a <topicref> specialization included in the DITA Learning and Training Map domain. It is used to reference a single <learningObjectMap>, either from a learning map, learning bookmap, learning group map or <learningGroup> element.

Inheritance

+ map/topicref learningmap-d/learningObjectMapRef

Example

```xml
<map id="map_1234">
  <title>Learning map with learningObject map references</title>
  <!-- As child of map -->
  <learningObjectMapRef href="learningObjectMap1.ditamap" type="learningObjectMap" format="ditamap" scope="local"/>
  <!-- ... -->
  <learningGroup navtitle="Top-Level Learning Group">
    <!-- As component of learningGroup -->
    <learningObjectMapRef href="learningObjectMap2.ditamap" type="learningObjectMap" format="ditamap" scope="local"/>
    <!-- ... -->
  </learningGroup>
</map>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with narrowed definitions of @href and @format, given below), Attributes common to many map elements (see page 665), Topicref element attributes group (see page 663), outputclass (see page 0 ), @keys, and @keyref.

@href

A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

@format

On this element the @format attribute sets a default value of "ditamap", because the purpose of the element is to reference a ditamap document. Otherwise, the attribute is the same as described in Link relationship attribute group (see page 661).

4.11.3.4 <learningGroupMapRef>

Use a <learningGroupMapRef> element to include a map reference to a learning group map as part of a DITA learning map, learning bookmap, learning group map, or <learningGroup> element.

Description and Semantics

The <learningGroupMapRef> is a <topicref> specialization included in the DITA Learning and Training Map domain. It is used to reference a single learning group map, either from a learning map, learning bookmap, learning group map or <learningGroup> element.
Inheritance
+ map/topicref learningmap-d/learningGroupMapRef

Example

```xml
<map id="map_1234">
  <title>Learning map with learningGroup map references</title>
  <!-- .... -->
  <learningGroupMapRef href="learningGroupMap1.ditamap" type="learningGroupMap" format="ditamap" scope="local"/>
  <!-- .... -->
  <learningGroup navtitle="Top-Level Learning Group">
    <!-- .... -->
    <learningGroupMapRef href="learningGroupMap2.ditamap" type="learningGroupMap" format="ditamap" scope="local"/>
    <!-- .... -->
  </learningGroup>
</map>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with narrowed definitions of @href and @format, given below), Attributes common to many map elements (see page 655), Topicref element attributes group (see page 663), outputclass (see page 0), @keys, and @keyref.

@href

A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

@format

On this element the @format attribute sets a default value of "ditamap", because the purpose of the element is to reference a ditamap document. Otherwise, the attribute is the same as described in Link relationship attribute group (see page 661).

4.11.3.5 <learningPlanRef>

Use a <learningPlanRef> element to include a topic reference to a learning plan topic as part of a <learningObject>.

By default, a <learningPlanRef> is expected to link to a <learningPlan> topic. However, the @href in a <learningPlanRef> can link to content of any type that you want to use as a learning plan in a learning object. In this way, you can structure any existing DITA content as a learning object, and then take advantage of the learning-specific processing available for building and packaging that content for delivery in a learning system or learning context.

Comment by robander, 18 February 2014

As part of the update to 1.3, John Hunt suggested removing defaults for @type. So, I've removed the following paragraph:

When you include a reference to a DITA topic that is not learningPlan, change the type attribute to match the type of topic being referenced.

Inheritance
+ map/topicref learningmap-d/learningPlanref
Example
See learningObject.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with narrowed definitions of @href and @format, given below), Attributes common to many map elements (see page 655) (without the @collection-type attribute, and with a narrowed definition of @chunk, given below), Topicref element attributes group (see page 663), outputclass (see page 0), @keys, and @keyref.

@href
A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

@format
On this element, the @format attribute has a default value of "dita", because it usually links to DITA learning topics. If linking to something other than DITA, set the @format attribute as described in The format attribute (see page 676).

@chunk
On this element, the @chunk attribute has a default value of "to-content", which causes the current branch of content to be published as a single unit. See Chunking (see page 102) for more information and for details on other supported values.

4.11.3.6 <learningPreAssessmentRef>

Use a <learningPreAssessmentRef> element to include a topic reference to a learning assessment topic as part of a DITA <learningObject>.

By default, a <learningPreAssessmentRef> is expected to link to a <learningAssessment> topic. However, the @href in a <learningPreAssessmentRef> can link to content of any type that you want to use as a learning assessment in a learning object. In this way, you can structure any existing DITA content as a learning object, and then take advantage of the learning-specific processing available for building and packaging that content for delivery in a learning system or learning context.

Comment by robander, 18 February 2014
As part of the update to 1.3, John Hunt suggested removing defaults for @type. So, I've removed the following paragraph:

When you include a reference to a DITA topic that is not learningAssessment, change the type attribute to match the type of topic being referenced.

Inheritance
+ map/topicref learningmap-d/learningPreAssessmentRef

Example
See learningObject.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with narrowed definitions of @href and @format, given below), Attributes common to many map elements (see page 655) (without the @collection-type attribute, and with
a narrowed definition of @chunk, given below), *Topicref element attributes group* (see page 663), *outputclass* (see page 0), @keys, and @keyref.

@href
A pointer to the resource represented by the `<topicref>`. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

@format
On this element, the @format attribute has a default value of "dita", because it usually links to DITA learning topics. If linking to something other than DITA, set the @format attribute as described in *The format attribute* (see page 676).

@chunk
On this element, the @chunk attribute has a default value of "to-content", which causes the current branch of content to be published as a single unit. See *Chunking* (see page 102) for more information and for details on other supported values.

### 4.11.3.7 `<learningOverviewRef>`

Use a `<learningOverviewRef>` element to include a topic reference to a learning overview topic as part of a `<learningObject>` element.

By default, a `<learningOverviewRef>` is expected to link to a `<learningOverview>` topic. However, the @href in a `<learningOverviewRef>` can link to content of any type that you want to use as a learning overview in a learning object. In this way, you can structure any existing DITA content as a learning object, and then take advantage of the learning-specific processing available for building and packaging that content for delivery in a learning system or learning context.

---

**Comment by robander, 18 February 2014**

As part of the update to 1.3, John Hunt suggested removing defaults for @type. So, I've removed the following paragraph:

*When you include a reference to a DITA topic that is not learningOverview, change the @type attribute to match the type of topic being referenced.*

---

### Inheritance

+ map/topicref learningmap-d/learningOverviewRef

### Example

See *learningObject*.

### Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661) (with narrowed definitions of @href and @format, given below), *Attributes common to many map elements* (see page 655) (without the @collection-type attribute, and with a narrowed definition of @chunk, given below), *Topicref element attributes group* (see page 663), *outputclass* (see page 0), @keys, and @keyref.

@href
A pointer to the resource represented by the `<topicref>`. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

@format
On this element, the @format attribute has a default value of "dita", because it usually links to DITA learning topics. If linking to something other than DITA, set the @format attribute as described in The format attribute (see page 676).

@chunk

On this element, the @chunk attribute has a default value of "to-content", which causes the current branch of content to be published as a single unit. See Chunking (see page 102) for more information and for details on other supported values.

4.11.3.8 <learningContentRef>

Use a <learningContentRef> element to include a topic reference to a learning content topic as part of a DITA <learningObject> element.

By default, a <learningContentRef> is expected to link to a <learningContent> topic. However, a <learningContentRef> can link to content of any type that you want to use as a learning content in a learning object. In this way, you can structure any existing DITA content as a learning object, and then take advantage of the learning-specific processing available for building and packaging that content for delivery in a learning system or learning context.

The presentation intent for <learningContent> is that the topic be presented as an atomic unit of presentation, for example, as required by SCORM delivery requirements. Thus the default value for the @chunk attribute is "to-content". You can use <learningContentComponentRef> elements to include subordinate topics.

Comment by robander, 18 February 2014

As part of the update to 1.3, John Hunt suggested removing defaults for @type. So, I've removed the following paragraph:

When you include a reference to a DITA topic that is not learningContent, change the type attribute value to match the type of topic being referenced.

Inheritance

+ map/topicref learningmap-d/learningContentRef

Example

See learningObject.

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with narrowed definitions of @href and @format, given below), Attributes common to many map elements (see page 655) (without the @collection-type attribute, and with a narrowed definition of @chunk, given below), Topicref element attributes group (see page 663), outputclass (see page 0 ), @keys, and @keyref.

@href

A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

@format

On this element, the @format attribute has a default value of "dita", because it usually links to DITA learning topics. If linking to something other than DITA, set the @format attribute as described in The format attribute (see page 676).

@chunk
On this element, the @chunk attribute has a default value of "to-content", which causes the current branch of content to be published as a single unit. See Chunking (see page 102) for more information and for details on other supported values.

4.11.3.9 <learningContentComponentRef>

Use a <learningContentComponentRef> element to include a topic reference to a topic that acts as a subtopic of a <learningContent> topic.

The learning and training architecture defines the child topics of <learningObject> as being atomic units of presentation. However, these topics may include subordinate topics. The <learningContentComponentRef> reference may be used to refer to any topic type. The presentation intent is that such topics are presented as integral parts of their parent topics. This intent is reflected in the default @chunk attribute value of "to-content" for the elements <learningContentRef>, <learningOverviewRef>, <learningPlanRef>, <learningPreAssessmentRef>, and <learningPostAssessmentRef>.

Inheritance
+ map/topicref learningmap-d/learningContentRef

Example

See learningObject.

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with narrowed definitions of @href and @format, given below), Attributes common to many map elements (see page 655) (without the @collection-type attribute), Topicref element attributes group (see page 663), outputclass (see page 0 ), @keys, and @keyref.

@href

A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

@format

On this element, the @format attribute has a default value of "dita", because it usually links to DITA learning topics. If linking to something other than DITA, set the @format attribute as described in The format attribute (see page 676).

4.11.3.10 <learningSummaryRef>

Use a <learningSummaryRef> to include topic reference to a learning summary topic in a DITA learning object.

By default, a <learningSummaryRef> is expected to link to a <learningSummary> topic. However, the @href attribute in a <learningSummaryRef> can link to content of any type that you want to use as a learning summary in a learning object. In this way, you can structure any existing DITA content as a learning object, and then take advantage of the learning-specific processing available for building and packaging that content for delivery in a learning system or learning context.

Comment by robander, 18 February 2014
As part of the update to 1.3, John Hunt suggested removing defaults for @type. So, I've removed the following paragraph:
When you include a reference to a DITA topic that is not learningSummary, change the type attribute to match the type of topic being referenced.
Inheritance
+ map/topicref learningmap-d/learningSummaryRef

Example
See learningObject.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Link relationship attribute group (see page 661) (with narrowed definitions of @href and @format, given below), Attributes common to many map elements (see page 655) (without the @collection-type attribute, and with a narrowed definition of @chunk, given below), Topicref element attributes group (see page 663), outputclass (see page 0 ), @keys, and @keyref.

@href
A pointer to the resource represented by the <topicref>. See The href attribute (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

@format
On this element, the @format attribute has a default value of "dita", because it usually links to DITA learning topics. If linking to something other than DITA, set the @format attribute as described in The format attribute (see page 676).

@chunk
On this element, the @chunk attribute has a default value of "to-content", which causes the current branch of content to be published as a single unit. See Chunking (see page 102) for more information and for details on other supported values.

4.11.3.11 <learningPostAssessmentRef>

Use a <learningPostAssessmentRef> element to include a topic reference to a learning assessment topic as part of a DITA <learningObject>.

By default, a <learningPostAssessmentRef> is expected to link to a <learningAssessment> topic. However, the @href in a <learningPostAssessmentRef> can link to content of any type that you want to use as a learning assessment in a learning object. In this way, you can structure any existing DITA content as a learning object, and then take advantage of the learning-specific processing available for building and packaging that content for delivery in a learning system or learning context.

Comment by robander , 18 February 2014
As part of the update to 1.3, John Hunt suggested removing defaults for @type. So, I've removed the following paragraph:
When you include a reference to a DITA topic that is not learningAssessment, change the type attribute to match the type of topic being referenced.

Inheritance
+ map/topicref learningmap-d/learningPostAssessmentRef

Example
See learningObject.
**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Link relationship attribute group* (see page 661) (with narrowed definitions of @href and @format, given below), *Attributes common to many map elements* (see page 655) (without the @collection-type attribute, and with a narrowed definition of @chunk, given below), *Topicref element attributes group* (see page 663), *outputclass* (see page 0 ), @keys, and @keyref.

@href

A pointer to the resource represented by the <topicref>. See *The href attribute* (see page 663) for detailed information on supported values and processing implications. References to DITA content cannot be below the topic level: that is, you cannot reference individual elements inside a topic. References to content other than DITA topics should use the @format attribute to identify the kind of resource being referenced.

@format

On this element, the @format attribute has a default value of "dita", because it usually links to DITA learning topics. If linking to something other than DITA, set the @format attribute as described in *The format attribute* (see page 676).

@chunk

On this element, the @chunk attribute has a default value of "to-content", which causes the current branch of content to be published as a single unit. See *Chunking* (see page 102) for more information and for details on other supported values.

4.11.4 Learning2 domain

4.11.4.1 <lcAnswerContent2>

The <lcAnswerContent2> element in a learning assessment interaction provides the content for an answer option, which the learner can select as correct or incorrect. This element replaces the <lcAnswerContent> element in DITA 1.2.

**Inheritance**

topic/div learningInteractionBase2-d/div learning2-d/lcAnswerContent2

**Example**

```dita
<!--True - False Interaction -->
<lcTrueFalse2>
  <lcInteractionLabel2>True - False sun rising</lcInteractionLabel2>
  <lcQuestion2>On Earth, the sun rises in the West and sets in the East.</lcQuestion2>
  <lcAnswerOptionGroup2>
    <lcAnswerOption2>
      <lcAnswerContent2>True</lcAnswerContent2>
    </lcAnswerOption2>
    <lcAnswerOption2>
      <lcAnswerContent2>False</lcAnswerContent2>
      <lcCorrectResponse2/>
    </lcAnswerOption2>
  </lcAnswerOptionGroup2>
  <lcFeedbackIncorrect2>No, look to the East.</lcFeedbackIncorrect2>
  <lcFeedbackCorrect2>Yes, look to the East.</lcFeedbackCorrect2>
</lcTrueFalse2>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0 ).
4.11.4.2 <lcAnswerOption2>

The <lcAnswerOption2> element in an assessment interaction provides the content and feedback for a question option, and can indicate the correct option. This element replaces the <lcAnswerOption> element in DITA 1.2.

Inheritance
+ topic/li learningInteractionBase2-d/li learning2-d/lcAnswerOption2

Example

```xml
<!--Single select Interaction -->
<lcSingleSelect2 id="singleselect">
  <lcInteractionLabel2>Multiple Choice - IEEE standards trivia</lcInteractionLabel2>
  <lcQuestion2>Which one of the listed standards committees is responsible for developing the token ring specification?</lcQuestion2>
  <lcAnswerOptionGroup2>
    <lcAnswerOption2>
      <lcAnswerContent2>IEEE 802.3</lcAnswerContent2>
      <lcFeedback2>Sorry. A little low.</lcFeedback2>
    </lcAnswerOption2>
    <lcAnswerOption2>
      <lcAnswerContent2>IEEE 802.5</lcAnswerContent2>
      <lcCorrectResponse2/>
      <lcFeedback2>That's the one.</lcFeedback2>
    </lcAnswerOption2>
    <lcAnswerOption2>
      <lcAnswerContent2>IEEE 802.6</lcAnswerContent2>
      <lcFeedback2>Nope. Too high.</lcFeedback2>
    </lcAnswerOption2>
    <lcAnswerOption2>
      <lcAnswerContent2>IEEE 802.11</lcAnswerContent2>
      <lcFeedback2>Nope. Way too high.</lcFeedback2>
    </lcAnswerOption2>
  </lcAnswerOptionGroup2>
</lcSingleSelect2>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.4.3 <lcAnswerOptionGroup2>

The <lcAnswerOptionGroup2> element provides a container for the options for a true-false, single-select, or multiple-select assessment interaction. This element replaces the <lcAnswerOptionGroup> element in DITA 1.2.

Inheritance
+ topic/ul learningInteractionBase2-d/ul learning2-d/lcAnswerOptionGroup2

Example

```xml
<!--Single select Interaction -->
<lcSingleSelect2 id="singleselect">
  <lcInteractionLabel2>Multiple Choice - IEEE standards trivia</lcInteractionLabel2>
  <lcQuestion2>Which one of the listed standards committees is responsible for developing the token ring specification?</lcQuestion2>
  <lcAnswerOptionGroup2>
    <lcAnswerOption2>
      <lcAnswerContent2>IEEE 802.3</lcAnswerContent2>
    </lcAnswerOption2>
    <!-- More options here... -->
  </lcAnswerOptionGroup2>
</lcSingleSelect2>
```
Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.11.4.4 <lcArea2>

A `<lcArea2>` defines an area of a hotspot image that contains a correct or incorrect choice in a hotspot assessment interaction. This element replaces the `<lcArea>` element in DITA 1.2.

Inheritance

+ topic/figgroup learningInteractionBase2-d/figgroup learning2-d/lcArea2

Example

```xml
<lcHotspot2>
  <lcInteractionLabel2>Team logos hotspot</lcInteractionLabel2>
  <lcQuestion2>Which of the following is the logo for the Cleveland Indians?</lcQuestion2>
  <lcHotspotMap2>
    <image href="hotlogos.gif"/>
    <lcArea2 id="indians">
      <lcAreaShape2>rect</lcAreaShape2>
      <lcAreaCoords2>0,0,50,50</lcAreaCoords2>
      <lcCorrectResponse2 value="indians"/>
      <lcFeedback2>Yes.</lcFeedback2>
    </lcArea2>
    <lcArea2 id="giants">
      <lcAreaShape2>rect</lcAreaShape2>
      <lcAreaCoords2>50,0,100,50</lcAreaCoords2>
      <lcFeedback2>No.</lcFeedback2>
    </lcArea2>
    <lcArea2 id="cardinals">
      <lcAreaShape2>rect</lcAreaShape2>
      <lcAreaCoords2>0,50,50,100</lcAreaCoords2>
      <lcFeedback2>No.</lcFeedback2>
    </lcArea2>
    <lcArea2 id="orioles">
      <lcAreaShape2>rect</lcAreaShape2>
      <lcAreaCoords2>50,50,100,100</lcAreaCoords2>
      <lcFeedback2>No.</lcFeedback2>
    </lcArea2>
  </lcHotspotMap2>
</lcHotspot2>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).
4.11.4.5 <lcAreaCoords2>

The <lcAreaCoords2> element specifies the coordinates of a linkable hotspot in a learning image. This element replaces the <lcAreaCoords> element in DITA 1.2.

This element contains text data representing coordinate data for learning images with linkable hotspots. Pixels are the recommended units for describing coordinates. The syntax of the coordinate data depends on the shape described by the coordinates, and is based on the image map definition in HTML. It uses the following data for the appropriate shapes:

**Shape**

**Data format**

- **rect** left-x, top-y, right-x, bottom-y
- **circle** center-x, center-y, radius
- **poly** x1, y1, x2, y2, ..., xN, yN. The first x and y coordinate pair and the last should be the same to close the polygon.

**Inheritance**

+ topic/ph learningInteractionBase2-d/ph learning2-d/lcAreaCoords2

**Example**

```xml
<!--Hotspot Interaction -->
<lcHotspot2>
    <lcInteractionLabel2>Team logos hotspot</lcInteractionLabel2>
    <lcQuestion2>Which of the following is the logo for the Cleveland Indians?</lcQuestion2>
    <lcHotspotMap2>
        <image href="hotlogos.gif"/>
        <lcArea2 id="indians">
            <lcAreaShape2>rect</lcAreaShape2>
            <lcAreaCoords2>0,0,50,50</lcAreaCoords2>
            <lcCorrectResponse2 value="indians"/>
            <lcFeedback2>Yes.</lcFeedback2>
        </lcArea2>
        <lcArea2 id="giants">
            <lcAreaShape2>rect</lcAreaShape2>
            <lcAreaCoords2>50,0,100,50</lcAreaCoords2>
            <lcFeedback2>No.</lcFeedback2>
        </lcArea2>
        <lcArea2 id="cardinals">
            <lcAreaShape2>rect</lcAreaShape2>
            <lcAreaCoords2>0,50,50,100</lcAreaCoords2>
            <lcFeedback2>No.</lcFeedback2>
        </lcArea2>
        <lcArea2 id="orioles">
            <lcAreaShape2>rect</lcAreaShape2>
            <lcAreaCoords2>50,50,100,100</lcAreaCoords2>
            <lcFeedback2>No.</lcFeedback2>
        </lcArea2>
    </lcHotspotMap2>
</lcHotspot2>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of @translate, given below), *outputclass* (see page 0), and @keyref. @translate
Indicates whether the content of the element should be translated or not. For this element the default value is "no". Setting to "yes" will override the default. The DITA architectural specification contains a list of each OASIS DITA element and its common processing default for the translate value; because this element uses an actual default, it will always be treated as translate="no" unless overridden as described. Available values are:

- **no**
  The content of this element is not translateable.

- **yes**
  The content of this element is translateable.

- **-dita-use-conref-target**
  See *Using the -dita-use-conref-target value* (see page 666) for more information.

### 4.11.4.6 `<lcAreaShape2>`

The `<lcAreaShape2>` element defines the shape of a linkable hotspot in a learning image. This element replaces the `<lcAreaShape>` element in DITA 1.2.

The `<lcAreaShape2>` element supports these values:

- **rect**
  Define a rectangular region. If you leave the `<lcAreaShape2>` element blank, this is assumed.

- **circle**
  Define a circular region.

- **poly**
  Define a polygonal region.

- **default**
  Indicates the entire diagram.

**Inheritance**

+ topic/keyword learningInteractionBase2-d/keyword learning2-d/lcAreaShape2

**Example**

```xml
<!--Hotspot Interaction -->
<lcHotspot2>
  <lcInteractionLabel2>Team logos hotspot</lcInteractionLabel2>
  <lcQuestion2>Which of the following is the logo for the Cleveland Indians?</lcQuestion2>
  <image href="hotlogos.gif"/>
  <lcArea2 id="indians">
    <lcAreaShape2>rect</lcAreaShape2>
    <lcAreaCoords2>0,0,50,50</lcAreaCoords2>
    <lcCorrectResponse2 value="indians"/>
    <lcFeedback2>Yes.</lcFeedback2>
  </lcArea2>
  <lcArea2 id="giants">
    <lcAreaShape2>rect</lcAreaShape2>
    <lcAreaCoords2>50,0,100,50</lcAreaCoords2>
    <lcFeedback2>No.</lcFeedback2>
  </lcArea2>
  <lcArea2 id="cardinals">
    <lcAreaShape2>rect</lcAreaShape2>
    <lcAreaCoords2>0,50,50,100</lcAreaCoords2>
    <lcFeedback2>No.</lcFeedback2>
  </lcArea2>
  <lcArea2 id="orioles">
    <lcAreaShape2>rect</lcAreaShape2>
    <lcAreaCoords2>50,50,100,100</lcAreaCoords2>
    <lcFeedback2>No.</lcFeedback2>
  </lcArea2>
</lcHotspot2>
```
Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @translate, given below), outputclass (see page 0), and @keyref.

@translate

Indicates whether the content of the element should be translated or not. For this element the default value is "no". Setting to "yes" will override the default. The DITA architectural specification contains a list of each OASIS DITA element and its common processing default for the translate value; because this element uses an actual default, it will always be treated as translate="no" unless overridden as described. Available values are:

no

The content of this element is not translateable.

yes

The content of this element is translateable.

-dita-use-conref-target

See Using the -dita-use-conref-target value (see page 666) for more information.

4.11.4.7 <lcCorrectResponse2>

The <lcCorrectResponse2> element in an assessment interaction indicates a correct response. This element replaces the <lcCorrectResponse> element in DITA 1.2.

In a <lcHotspot2> interaction, indicate the correct hotspot by setting the @value attribute of <lcCorrectResponse2> to the id of the "correct" area in the image map.

Inheritance

+ topic/data learningInteractionBase2-d/data learning2-d/lcCorrectResponse2

Example

<!--True - False Interaction -->
<lcTrueFalse2>
<lcInteractionLabel2>True - False sun rising</lcInteractionLabel2>
<lcQuestion2>On Earth, the sun rises in the West and sets in the East.</lcQuestion2>
<lcAnswerOptionGroup2>
<lcAnswerOption2>
<lcAnswerContent2>True</lcAnswerContent2>
</lcAnswerOption2>
<lcAnswerOption2>
<lcAnswerContent2>False</lcAnswerContent2>
</lcAnswerOption2>
</lcAnswerOptionGroup2>
<lcFeedbackIncorrect2>No, look to the East.</lcFeedbackIncorrect2>
<lcFeedbackCorrect2>Yes, look to the East.</lcFeedbackCorrect2>
</lcTrueFalse2>

Example with the <lcHotspot2> element

<!--Hotspot Interaction -->
<lcHotspot2>
<lcInteractionLabel2>Team logos hotspot</lcInteractionLabel2>
<lcQuestion2>Which of the following is the logo for the Cleveland Indians?</lcQuestion2>
<lcHotspotMap2>
<image href="hotlogos.gif"/>
<lcArea2 id="indians">
  <lcAreaShape2>rect</lcAreaShape2>
  <lcAreaCoords2>0,0,50,50</lcAreaCoords2>
  <lcCorrectResponse2 value="indians"/>
  <lcFeedback2>Yes.</lcFeedback2>
</lcArea2>

<lcArea2 id="giants">
  <lcAreaShape2>rect</lcAreaShape2>
  <lcAreaCoords2>50,0,100,50</lcAreaCoords2>
  <lcFeedback2>No.</lcFeedback2>
</lcArea2>

<lcArea2 id="cardinals">
  <lcAreaShape2>rect</lcAreaShape2>
  <lcAreaCoords2>0,50,50,100</lcAreaCoords2>
  <lcFeedback2>No.</lcFeedback2>
</lcArea2>

<lcArea2 id="orioles">
  <lcAreaShape2>rect</lcAreaShape2>
  <lcAreaCoords2>50,50,100,100</lcAreaCoords2>
  <lcFeedback2>No.</lcFeedback2>
</lcArea2>
</lcHotspotMap2>
</lcHotspot2>

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and the attributes defined below.

@name

The name used to indicate this data. The default value for this attribute is "lcCorrectResponse2".

@value (REQUIRED)

The value. In an <lcHotspot2> interaction, set the value to the id corresponding to the correct area in the imagemap.

4.11.4.8 <lcFeedback2>

The <lcFeedback2> element in an assessment interaction provides information to the learner about a correct or incorrect response. This element replaces the <lcFeedback> element in DITA 1.2.

Inheritance

+ topic/div learningInteractionBase2-d/div learning2-d/lcFeedback2

Example

<!--Single select Interaction -->
<lcSingleSelect2 id="singleselect">
  <lcInteractionLabel2>Multiple Choice - IEEE standards trivia</lcInteractionLabel2>
  <lcQuestion2>Which one of the listed standards committees is responsible for developing the token ring specification?</lcQuestion2>
  <lcAnswerOptionGroup2>
    <lcAnswerOption2>
      <lcAnswerContent2>IEEE 802.3</lcAnswerContent2>
      <lcFeedback2>Sorry. A little low.</lcFeedback2>
    </lcAnswerOption2>
    <lcAnswerOption2>
      <lcAnswerContent2>IEEE 802.5</lcAnswerContent2>
      <lcCorrectResponse2/>
      <lcFeedback2>That's the one.</lcFeedback2>
    </lcAnswerOption2>
    <lcAnswerOption2>
      <lcAnswerContent2>IEEE 802.6</lcAnswerContent2>
      <lcFeedback2>Nope. Too high.</lcFeedback2>
    </lcAnswerOption2>
    <lcAnswerOption2>
      <lcAnswerContent2>IEEE 802.11</lcAnswerContent2>
      <lcFeedback2>Nope. Way too high.</lcFeedback2>
    </lcAnswerOption2>
  </lcAnswerOptionGroup2>
</lcSingleSelect2>
**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

### 4.11.4.9 `<lcFeedbackCorrect2>`

The `<lcFeedbackCorrect2>` element in an assessment interaction provides feedback to the learner about a correct response. This element replaces the `<lcFeedbackCorrect>` element in DITA 1.2.

**Inheritance**

+ topic/div learningInteractionBase2-d/div learning2-d/lcFeedbackCorrect2

**Example**

```xml
<!--True - False Interaction        -->
<lctruefalse2>
<lciinteractionlabel2>True - False sun rising</lciinteractionlabel2>
<lciquestion2>On Earth, the sun rises in the West and sets in the East.</lciquestion2>
<lciansweroptiongroup2>
<lciansweroption2>
<lcianswercontent2>True</lcianswercontent2>
</lcansweroption2>
<lciansweroption2>
<lcianswercontent2>False</lcianswercontent2>
<lccorrectresponse2/>
</lcansweroption2>
</lcansweroptiongroup2>
<lcfedbackincorrect2>No, look to the East.</lcfedbackincorrect2>
<lcfedbackcorrect2>Yes, look to the East.</lcfedbackcorrect2>
</lctruefalse2>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

### 4.11.4.10 `<lcFeedbackIncorrect2>`

The `<lcFeedbackIncorrect2>` element in an assessment interaction provides feedback about incorrect response. This element replaces the `<lcFeedbackIncorrect>` element in DITA 1.2.

**Inheritance**

+ topic/div learningInteractionBase2-d/div learning2-d/lcFeedbackIncorrect2

**Example**

```xml
<!--True - False Interaction        -->
<lctruefalse2>
<lciinteractionlabel2>True - False sun rising</lciinteractionlabel2>
<lciquestion2>On Earth, the sun rises in the West and sets in the East.</lciquestion2>
<lciansweroptiongroup2>
<lciansweroption2>
<lcianswercontent2>True</lcianswercontent2>
</lcansweroption2>
<lciansweroption2>
<lcianswercontent2>False</lcianswercontent2>
</lcansweroption2>
<lcfedbackincorrect2>No, look to the East.</lcfedbackincorrect2>
</lciansweroptiongroup2>
```

DITA Version 1.3 Specification  
TBD  
Copyright © OASIS Open 2005, 2014. All Rights Reserved.
<lcAnswerContent2>False</lcAnswerContent2>
<lcCorrectResponse2/>
</lcAnswerOption2>
</lcAnswerOptionGroup2>
<lcFeedbackIncorrect2>No, look to the East.</lcFeedbackIncorrect2>
<lcFeedbackCorrect2>Yes, look to the East.</lcFeedbackCorrect2>
</lcTrueFalse2>

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.4.11 <lcHotspot2>

In a <lcHotspot2> interaction, the learner clicks on a region of the screen to indicate a choice. This element replaces the <lcHotspot> element in DITA 1.2.

Inheritance
+ topic/div learningInteractionBase2-d/lcInteractionBase2 learning2-d/lcHotspot2

Example

<!--Hotspot Interaction -->
<lcHotspot2>
  <lcInteractionLabel2>Team logos hotspot</lcInteractionLabel2>
  <lcQuestion2>Which of the following is the logo for the Cleveland Indians?</lcQuestion2>
  <lcHotspotMap2>
    <image href="hotlogos.gif"/>
    <lcArea2 id="indians">
      <lcAreaShape2>rect</lcAreaShape2>
      <lcAreaCoords2>0,0,50,50</lcAreaCoords2>
      <lcCorrectResponse2 value="indians"/>
      <lcFeedback2>Yes.</lcFeedback2>
    </lcArea2>
    <lcArea2 id="giants">
      <lcAreaShape2>rect</lcAreaShape2>
      <lcAreaCoords2>50,0,100,50</lcAreaCoords2>
      <lcFeedback2>No.</lcFeedback2>
    </lcArea2>
    <lcArea2 id="cardinals">
      <lcAreaShape2>rect</lcAreaShape2>
      <lcAreaCoords2>0,50,50,100</lcAreaCoords2>
      <lcFeedback2>No.</lcFeedback2>
    </lcArea2>
    <lcArea2 id="orioles">
      <lcAreaShape2>rect</lcAreaShape2>
      <lcAreaCoords2>50,50,100,100</lcAreaCoords2>
      <lcFeedback2>No.</lcFeedback2>
    </lcArea2>
  </lcHotspotMap2>
</lcHotspot2>

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below) and outputclass (see page 0).

@id (REQUIRED)
Defines an ID by which the element may be referenced. This attribute is defined with the XML Data Type NMTOKEN.
4.11.4.12 <lcHotspotMap2>

A <lcHotspotMap2> interaction lets you designate an action area or region over an image, allowing a click in that region to get scored as correct or incorrect in response to an interaction question. This element replaces the <lcHotspotMap> element in DITA 1.2.

Inheritance
+ topic/fig learningInteractionBase2-d/figgroup learning2-d/lcHotspotMap2

Example

```xml
<!--Hotspot Interaction -->
<lcHotspot2>
  <lcInteractionLabel2>Team logos hotspot</lcInteractionLabel2>
  <lcQuestion2>Which of the following is the logo for the Cleveland Indians?</lcQuestion2>
  <lcHotspotMap2>
    <image href="hotlogos.gif"/>
    <lcArea2 id="indians">
      <lcAreaShape2>rect</lcAreaShape2>
      <lcAreaCoords2>0,0,50,50</lcAreaCoords2>
      <lcCorrectResponse2 value="indians"/>
      <lcFeedback2>Yes.</lcFeedback2>
    </lcArea2>
    <lcArea2 id="giants">
      <lcAreaShape2>rect</lcAreaShape2>
      <lcAreaCoords2>50,0,100,50</lcAreaCoords2>
      <lcFeedback2>No.</lcFeedback2>
    </lcArea2>
    <lcArea2 id="cardinals">
      <lcAreaShape2>rect</lcAreaShape2>
      <lcAreaCoords2>0,50,50,100</lcAreaCoords2>
      <lcFeedback2>No.</lcFeedback2>
    </lcArea2>
    <lcArea2 id="orioles">
      <lcAreaShape2>rect</lcAreaShape2>
      <lcAreaCoords2>50,50,100,100</lcAreaCoords2>
      <lcFeedback2>No.</lcFeedback2>
    </lcArea2>
  </lcHotspotMap2>
</lcHotspot2>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.4.13 <lcInstructornote2>

Use the <lcInstructornote2> element to provide information or notes you want to provide to the course instructor. These notes can be conditionalized out of content you intend to deliver to the learner. This element replaces the <lcInstructornote> element in DITA 1.2.

Inheritance
+ topic/note learningInteractionBase2-d/note learning2-d/lcInstructornote2

Example

```xml
<lintro>
  <title>Introduction</title>
  <p>If you need an introduction, you would include it here.</p>
  <note>You can have notes, tables, all kinds of things like that here, if you desire.</note>
</lintro>
```
Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.4.14 <lcInteractionBase2>

Use <lcInteractionBase2> as the base for more specialized assessment types. This element replaces the <lcInteractionBase> element in DITA 1.2.

Inheritance
+ topic/div learningInteractionBase2-d/lcInteractionBase2

Example
<lcInteractionBase2> should only be used as a base for further specialization.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below) and outputclass (see page 0).

@id (REQUIRED)
Defines an ID by which the element may be referenced. This attribute is defined with the XML Data Type NMTOKEN.

4.11.4.15 <lcItem2>

The <lcItem2> element in an assessment interaction provides the content for an item that matches the match item in a match table. This element replaces the <lcItem> element in DITA 1.2.

Inheritance
+ topic/div learningInteractionBase2-d/lcInteractionBase2

Example
See lcMatching2 (see page 579).

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.4.16 <lcMatching2>

In a <lcMatching2> interaction, the learner identifies the correct choice that matches another choice and, optionally, any feedback for the pair or for correct or incorrect matches. This element replaces the <lcMatching> element in DITA 1.2.

Inheritance
+ topic/div learningInteractionBase2-d/lcInteractionBase2 learning2-d/lcMatching2
Example

```xml
<!--Matching Interaction -->
<lcmatching2>
  <lcinteractionlabel2>Matching teams with cities</lcinteractionlabel2>
  <lcquestion2>Match the team with the city.</lcquestion2>
  <lcmatchtable2>
    <lcmatchingheader2>
      <lcitem2>Team</lcitem2>
      <lcmatchingitem2>City</lcmatchingitem2>
    </lcmatchingheader2>
    <lcmatchingpair2>
      <lcitem2>Boston</lcitem2>
      <lcmatchingitem2>Red Sox</lcmatchingitem2>
      <lcmatchingitemfeedback2>
        <lcfedbackcorrect2>The Red Sox play in Boston's Fenway Park</lcfedbackcorrect2>
        <lcfedbackincorrect2>That is not Boston's home team</lcfedbackincorrect2>
      </lcmatchingitemfeedback2>
    </lcmatchingpair2>
    <lcmatchingpair2>
      <lcitem2>San Francisco</lcitem2>
      <lcmatchingitem2>Giants</lcmatchingitem2>
      <lcmatchingitemfeedback2>
        <lcfedbackcorrect2>The Giants play in San Francisco's Candlestick Park</lcfedbackcorrect2>
        <lcfedbackincorrect2>That is not San Francisco's home team</lcfedbackincorrect2>
      </lcmatchingitemfeedback2>
    </lcmatchingpair2>
    <lcmatchingpair2>
      <lcitem2>Chicago</lcitem2>
      <lcmatchingitem2>Cubs</lcmatchingitem2>
      <lcmatchingitemfeedback2>
        <lcfedbackcorrect2>The Cubs play in Chicago's Wrigley Field</lcfedbackcorrect2>
        <lcfedbackincorrect2>That is not Chicago's home team</lcfedbackincorrect2>
      </lcmatchingitemfeedback2>
    </lcmatchingpair2>
    <lcmatchingpair2>
      <lcitem2>Toronto</lcitem2>
      <lcmatchingitem2>Blue Jays</lcmatchingitem2>
      <lcmatchingitemfeedback2>
        <lcfedbackcorrect2>The Blue Jays play in Toronto's SkyDome</lcfedbackcorrect2>
        <lcfedbackincorrect2>That is not Toronto's home team</lcfedbackincorrect2>
      </lcmatchingitemfeedback2>
    </lcmatchingpair2>
  </lcmatchtable2>
</lcmatching2>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below) and outputclass (see page 0).

@id (REQUIRED)

Defines an ID by which the element may be referenced. This attribute is defined with the XML Data Type NMTOKEN.

4.11.4.17 <lcMatchingItem2>

The <lcMatchingItem2> element in an assessment interaction provides the content for the matching side of a matching pair of items in a match table interaction. This element replaces the <lcMatchingItem> element in DITA 1.2.

Inheritance

+ topic/div learningInteractionBase2-d/stentry learning2-d/lcMatchingItem2

Example

See lcMatching2 (see page 579).
4.11.4.18 <lcMatchingItemFeedback2>

The <lcMatchingItemFeedback2> element in an assessment interaction provides feedback on the match as a whole or on correct and incorrect matches or both. This element replaces the <lcMatchingItemFeedback> element in DITA 1.2.

Inheritance

+ topic/div learningInteractionBase2-d/stentry learning2-d/lcMatchingItemFeedback2

Example

See lcMatching2 (see page 579).

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.4.19 <lcMatchingPair2>

The <lcMatchingPair2> element in an assessment interaction provides a table row with the pair of items that comprise a correct match in a matching interaction and, optionally, feedback on the pair or correct and incorrect matches. This element replaces the <lcMatchingPair> element in DITA 1.2.

Inheritance

+ topic/div learningInteractionBase2-d/strow learning2-d/lcMatchingPair2

Example

See lcMatching2 (see page 579).

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.4.20 <lcMatchTable2>

The <lcMatchTable2> element in an assessment interaction provides a format for matching items. This element replaces the <lcMatchTable> element in DITA 1.2.

Inheritance

+ topic/div learningInteractionBase2-d/simpletable learning2-d/lcMatchTable2

Example

See lcMatching2 (see page 579).
**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

### 4.11.4.21 `<lcMultipleSelect2>`

In a `<lcMultipleSelect2>` interaction, the learner must indicate two or more correct answers from a list of choices. This element replaces the `<lcMultipleSelect>` element in DITA 1.2.

**Inheritance**

+ topic/div learningInteractionBase2-d/lcInteractionBase2 learning2-d/lcMultipleSelect2

**Example**

```xml
<!|--Multiple select Interaction -->
<lcMultipleSelect2>
  <lcInteractionLabel2>Finding Major League Baseball logos</lcInteractionLabel2>
  <lcQuestion2>Which one of the following is a logo of a Major League Baseball team? (You may choose more than one.)</lcQuestion2>
  <lcAnswerOptionGroup2>
    <lcAnswerOption2 id="A">
      <lcAnswerContent2>
        <image href="logo1.gif"/>
      </lcAnswerContent2>
      <lcCorrectResponse2/>
      <lcFeedback2>Yes, that's one.</lcFeedback2>
    </lcAnswerOption2>
    <lcAnswerOption2 id="B">
      <lcAnswerContent2>
        <image href="logo2.gif"/>
      </lcAnswerContent2>
      <lcCorrectResponse2/>
      <lcFeedback2>Yes, that's one.</lcFeedback2>
    </lcAnswerOption2>
    <lcAnswerOption2 id="C">
      <lcAnswerContent2>
        <image href="logo3.gif"/>
      </lcAnswerContent2>
      <lcFeedback2>No, not that one. Sorry!</lcFeedback2>
    </lcAnswerOption2>
    <lcAnswerOption2 id="D">
      <lcCorrectResponse2/>
      <lcAnswerContent2>
        <image href="logo4.gif"/>
      </lcAnswerContent2>
      <lcFeedback2>Yes, that's one.</lcFeedback2>
    </lcAnswerOption2>
  </lcAnswerOptionGroup2>
</lcMultipleSelect2>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of `@id`, given below) and *outputclass* (see page 0).

### `@id` *(REQUIRED)*

Defines an ID by which the element may be referenced. This attribute is defined with the XML Data Type `NMTOKEN`.

---

DITA Version 1.3 Specification

Copyright © OASIS Open 2005, 2014. All Rights Reserved.

Page 582 of 747
4.11.4.22 <lcOpenAnswer2>

Use <lcOpenAnswer2> to provide a suggested answer for an <lcOpenQuestion2> interaction. This element replaces the <lcOpenAnswer> element in DITA 1.2.

Inheritance
+ topic/div learningInteractionBase2-d/div learning2-d/lcOpenAnswer2

Example

```xml
<lcOpenQuestion2 id="oq1">
  <lcInteractionLabel2>Cows and moon jumps</lcInteractionLabel2>
  <lcQuestion2>Describe how it might be possible for a cow to jump over the moon.</lcQuestion2>
  <lcOpenAnswer2>Cows can only jump over the moon in nursery rhymes.</lcOpenAnswer2>
</lcOpenQuestion2>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.4.23 <lcOpenQuestion2>

Use <lcOpenQuestion2> to pose an open-ended question in an assessment interaction. This element replaces the <lcOpenQuestion> element in DITA 1.2.

Inheritance
+ topic/div learningInteractionBase2-d/lcInteractionBase2 learning2-d/lcOpenQuestion2

Example

```xml
<lcOpenQuestion2 id="oq1">
  <lcInteractionLabel2>Cows and moon jumps</lcInteractionLabel2>
  <lcQuestion2>Describe how it might be possible for a cow to jump over the moon.</lcQuestion2>
  <lcOpenAnswer2>Cows can only jump over the moon in nursery rhymes.</lcOpenAnswer2>
</lcOpenQuestion2>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below) and outputclass (see page 0).

@id (REQUIRED)
Defines an ID by which the element may be referenced. This attribute is defined with the XML Data Type NMTOKEN.
4.11.4.24 <lcQuestion2>

Use the <lcQuestion2> element in an interaction to ask the question. This element replaces the <lcQuestion> element in DITA 1.2.

Inheritance
+ topic/div learningInteractionBase2-d/lcQuestionBase2 learning2-d/lcQuestion2

Example

```xml
<!-- True - False Interaction -->
<lcTrueFalse2>
  <lcInteractionLabel2>True - False sun rising</lcInteractionLabel2>
  <lcQuestion2>On Earth, the sun rises in the West and sets in the East.</lcQuestion2>
  <lcAnswerOptionGroup2>
    <lcAnswerOption2>
      <lcAnswerContent2>True</lcAnswerContent2>
    </lcAnswerOption2>
    <lcAnswerOption2>
      <lcAnswerContent2>False</lcAnswerContent2>
      <lcCorrectResponse2/>
    </lcAnswerOption2>
  </lcAnswerOptionGroup2>
  <lcFeedbackIncorrect2>No, look to the East.</lcFeedbackIncorrect2>
  <lcFeedbackCorrect2>Yes, look to the East.</lcFeedbackCorrect2>
</lcTrueFalse2>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.4.25 <lcQuestionBase2>

The <lcQuestionBase2> element is the base for <lcQuestion2> in the learning2Domain domain. This is an "abstract" element type intended only for use as a base for specialization. This element replaces the <lcQuestion> element in DITA 1.2.

Inheritance
+ topic/div learningInteractionBase2-d/lcQuestionBase2

Example

See <lcQuestion2>.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.4.26 <lcSequence2>

The <lcSequence2> element in an assessment interaction provides the position of a sequence option in a sequence. This element replaces the <lcSequence> element in DITA 1.2.

Inheritance
+ topic/data learningInteractionBase2-d/data learning2-d/lcSequence2
Order the following U.S. cities according to population, from largest to smallest.

```
<lcSequenceOptionGroup2>
  <lcSequenceOption2>
    <lcAnswerContent2>Portland, Oregon</lcAnswerContent2>
    <lcSequence2 name="lcSequence2" value="2"/>
  </lcSequenceOption2>
  <lcSequenceOption2>
    <lcAnswerContent2>Chicago, Illinois</lcAnswerContent2>
    <lcSequence2 name="lcSequence2" value="1"/>
  </lcSequenceOption2>
  <lcSequenceOption2>
    <lcAnswerContent2>Portland, Maine</lcAnswerContent2>
    <lcSequence2 name="lcSequence2" value="4"/>
  </lcSequenceOption2>
  <lcSequenceOption2>
    <lcAnswerContent2>Syracuse, New York</lcAnswerContent2>
    <lcSequence2 name="lcSequence2" value="3"/>
  </lcSequenceOption2>
</lcSequenceOptionGroup2>
```

No, try again, please.
Very good.

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and the attributes defined below.

- **@name**
  The name used to indicate this data. The default value for this attribute is "lcSequence2".

- **@value (REQUIRED)**
  The position of an option in a sequence of options. The value *MUST* be a number.

- **@datatype (REQUIRED)**
  The datatype for this data. Use numbers for sequence.

---

**Comment by robander, 14 February 2014**

Read this while converting attribute table. There is something wrong here. The attribute table says "required", but I cannot tell from the description what is supposed to go in the attribute. The example above shows elements that do not use this attribute. I've sent email to the TC to clarify.

---

**4.11.4.27 <lcSequenceOption2>**

The `<lcSequenceOption2>` element in an assessment interaction provides the contents of an item in a sequence interaction. This element replaces the `<lcSequenceOption>` element in DITA 1.2.

**Inheritance**

+ topic/li learningInteractionBase2-d/lcSequenceOption2

**Example**

```
<lcSequencing2>
  <lcInteractionLabel2>Sequencing City Populations in the U.S.</lcInteractionLabel2>
  <lcQuestion2>Order the following U.S. cities according to population, from largest to smallest.</lcQuestion2>
  <lcSequenceOptionGroup2>
    <lcSequenceOption2>
      <lcAnswerContent2>Portland, Oregon</lcAnswerContent2>
    </lcSequenceOption2>
  </lcSequenceOptionGroup2>
</lcSequencing2>
```
Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.11.4.28 `<lcSequenceOptionGroup2>`

The `<lcSequenceOptionGroup2>` element provides the options for an assessment sequence interaction. This element replaces the `<lcSequenceOptionGroup>` element in DITA 1.2.

Inheritance

+ topic/ol learningInteractionBase2-d/ol learning2-d/lcSequenceOptionGroup2

Example

```xml
<lcSequencing2>
  <lcInteractionLabel2>Sequencing City Populations in the U.S.</lcInteractionLabel2>
  <lcQuestion2>Order the following U.S. cities according to population, from largest to smallest.</lcQuestion2>
  <lcSequenceOptionGroup2>
    <lcSequenceOption2>
      <lcAnswerContent2>Portland, Oregon</lcAnswerContent2>
      <lcSequence2 name="lcSequence2" value="2="/>
    </lcSequenceOption2>
    <lcSequenceOption2>
      <lcAnswerContent2>Chicago, Illinois</lcAnswerContent2>
      <lcSequence2 name="lcSequence2" value="1="/>
    </lcSequenceOption2>
    <lcSequenceOption2>
      <lcAnswerContent2>Portland, Maine</lcAnswerContent2>
      <lcSequence2 name="lcSequence2" value="4="/>
    </lcSequenceOption2>
    <lcSequenceOption2>
      <lcAnswerContent2>Syracuse, New York</lcAnswerContent2>
      <lcSequence2 name="lcSequence2" value="3="/>
    </lcSequenceOption2>
  </lcSequenceOptionGroup2>
  <lcFeedbackIncorrect2>No, try again, please.</lcFeedbackIncorrect2>
  <lcFeedbackCorrect2>Very good.</lcFeedbackCorrect2>
</lcSequencing2>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).
4.11.4.29 `<lcSequencing2>`

A `<lcSequencing2>` interaction asks the learner to arrange a list of choices into a predefined order, such as small to large. This element replaces the `<lcSequencing>` element in DITA 1.2.

**Inheritance**

+ `topic/div learningInteractionBase2-d/lcInteractionBase2 learning2-d/lcSequencing2`  

**Example**

```xml
<lcSequencing2>
  <lcInteractionLabel2>Sequencing City Populations in the U.S.</lcInteractionLabel2>
  <lcQuestion2>Order the following U.S. cities according to population, from largest to smallest.</lcQuestion2>
  <lcSequenceOptionGroup2>
    <lcSequenceOption2>
      <lcAnswerContent2>Portland, Oregon</lcAnswerContent2>
      <lcSequence2 name="lcSequence2" value="2"/>
    </lcSequenceOption2>
    <lcSequenceOption2>
      <lcAnswerContent2>Chicago, Illinois</lcAnswerContent2>
      <lcSequence2 name="lcSequence2" value="1"/>
    </lcSequenceOption2>
    <lcSequenceOption2>
      <lcAnswerContent2>Portland, Maine</lcAnswerContent2>
      <lcSequence2 name="lcSequence2" value="4"/>
    </lcSequenceOption2>
    <lcSequenceOption2>
      <lcAnswerContent2>Syracuse, New York</lcAnswerContent2>
      <lcSequence2 name="lcSequence2" value="3"/>
    </lcSequenceOption2>
  </lcSequenceOptionGroup2>
  <lcFeedbackIncorrect2>No, try again, please.  
  </lcFeedbackIncorrect2>
  <lcFeedbackCorrect2>Very good.</lcFeedbackCorrect2>
</lcSequencing2>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of `@id`, given below) and *outputclass* (see page 0).

`@id` *(REQUIRED)*  
Defines an ID by which the element may be referenced. This attribute is defined with the XML Data Type `NMTOKEN`.

4.11.4.30 `<lcSingleSelect2>`

A `<lcSingleSelect2>` interaction presents three or more choices, only one of which is correct. This element replaces the `<lcSingleSelect>` element in DITA 1.2.

**Inheritance**

+ `topic/div learningInteractionBase2-d/lcInteractionBase2 learning2-d/lcSingleSelect2`  

**Example**

```xml
<!-- Single select Interaction                -->
<lcSingleSelect2 id="singelselect">  
  <lcInteractionLabel2>Multiple Choice - IEEE standards trivia</lcInteractionLabel2>
  <lcQuestion2>Which one of the listed standards committees is responsible for developing the token ring specification?</lcQuestion2>
  <lcAnswerOptionGroup2>
    <lcAnswerOption2>
      <lcAnswerContent2>IEEE 802.5</lcAnswerContent2>
    </lcAnswerOption2>
    <lcAnswerOption2>
      <lcAnswerContent2>IEEE 802.3</lcAnswerContent2>
    </lcAnswerOption2>
    <lcAnswerOption2>
      <lcAnswerContent2>IEEE 802.11</lcAnswerContent2>
    </lcAnswerOption2>
    <lcAnswerOption2>
      <lcAnswerContent2>IEEE 802.9</lcAnswerContent2>
    </lcAnswerOption2>
  </lcAnswerOptionGroup2>
</lcSingleSelect2>
```
Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of @id, given below) and *outputclass* (see page 0).

@id (REQUIRED)
Defines an ID by which the element may be referenced. This attribute is defined with the XML Data Type NMTOKEN.

4.11.4.31 <lcTrueFalse2>
A `<lcTrueFalse2>` interaction presents the learner with two choices, one correct, the other incorrect, often presented as true/false or yes/no responses. This element replaces the `<lcTrueFalse>` element in DITA 1.2.

Inheritance
+ topic/div learningInteractionBase2-d/lcInteractionBase2 learning2-d/lcTrueFalse2

Example

```
<!-- True - False Interaction -->
<lcTrueFalse2>
  <lcInteractionLabel2>True - False sun rising</lcInteractionLabel2>
  <lcQuestion2>On Earth, the sun rises in the West and sets in the East.</lcQuestion2>
  <lcAnswerOptionGroup2>
    <lcAnswerOption2>
      <lcAnswerContent2>True</lcAnswerContent2>
      <lcCorrectResponse2/>
    </lcAnswerOption2>
    <lcAnswerOption2>
      <lcAnswerContent2>False</lcAnswerContent2>
      <lcFeedbackIncorrect2>No, look to the East.</lcFeedbackIncorrect2>
    </lcAnswerOption2>
  </lcAnswerOptionGroup2>
</lcTrueFalse2>
```

Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of @id, given below) and *outputclass* (see page 0).

@id (REQUIRED)
Defines an ID by which the element may be referenced. This attribute is defined with the XML Data Type NMTOKEN.

4.11.5 Learning and training interactions domain elements

Use the learning interactions to construct a basic set of question and response interaction. The domain also provides an lcInstructornote element, which you can use to provide instructor-specific information in the learning topics.

4.11.5.1 <lcInstructornote>

The <lcInstructornote> was deprecated in DITA version 1.3. It was replaced with the <lcInstructornote2> element. It was originally intended to provide information or notes you want to provide to the course instructor. These notes can be conditionalized out of content you intend to deliver to the learner.

Inheritance

+ topic/note learningInteractionBase-d/note learning-d/lcInstructornote

Example

```xml
<lcIntro>
<title>Introduction</title>
<p>If you need an introduction, you would include it here.</p>
<note>You can have notes, tables, all kinds of things like that here, if you desire.</note>
<lcInstructornote>You can provide tips for instructors in an instructor note, like this one.</lcInstructornote>
</lcIntro>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.5.2 DITA learning interaction base domain elements

The learning interaction base 2 domain defines an "abstract" base type for all learning assessments. This element replaces the learning interaction base domain in DITA 1.2. This base type enables recognition of elements as interactions as distinct from other topic/div elements. The lcInteractionBase2 element is intended to be used only as a base for specialization. It should not be used directly as an element type in DITA documents.

4.11.5.2.1 <lcInteractionBase>

The <lcInteractionBase> element was deprecated in DITA version 1.3. It was replaced with the <lcInteractionBase2> element. It was originally intended as the base for more specialized assessment types.

Inheritance

+ topic/fig lcInteractionBase-d/lcInteractionBase

lcInteractionBase should only be used as a base for further specialization.

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below) and outputclass (see page 0).
@id (REQUIRED)

Defines an ID by which the element may be referenced. This attribute is defined with the XML Data Type NMTOKEN.

4.11.5.2.2 <lcQuestionBase>

The <lcQuestionBase> element was deprecated in DITA version 1.3. It was replaced with the <lcQuestionBase2> element. It was originally intended as the base for lcQuestion in the learningDomain domain. This is an "abstract" element type intended only for use as a base for specialization.

Inheritance
+ topic/p learningInteractionBase-d/lcQuestionBase

Example
See lcQuestion.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.5.2.3 <lcOpenQuestion>

The <lcOpenQuestion> element was deprecated in DITA version 1.3. It was replaced with the <lcOpenQuestion2> element. It was originally intended to pose an open-ended question in an assessment interaction.

Inheritance
+ topic/fig learningInteractionBase-d/lcInteractionBase learning-d/lcOpenQuestion

Example

<!--Open Question sample -->
<lcOpenQuestion id="oq1">
  <title>Cows and moon jumps</title>
  <lcQuestion>Describe how it might be possible for a cow to jump over the moon.</lcQuestion>
  <lcOpenAnswer>Cows can only jump over the moon in nursery rhymes.</lcOpenAnswer>
</lcOpenQuestion>

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below) and outputclass (see page 0).

@id (REQUIRED)

Defines an ID by which the element may be referenced. This attribute is defined with the XML Data Type NMTOKEN.
4.11.5.2.4 <lcTrueFalse>

The element <lcTrueFalse> was deprecated in DITA version 1.3. It was replaced with the <lcTrueFalse2> element. It was originally intended to present the learner with two choices, one correct, the other incorrect, often presented as true/false or yes/no responses.

**Inheritance**

+ topic/fig learningInteractionBase-d/lcInteractionBase learning-d/lcTrueFalse

**Example**

```xml
<!-- True - False Interaction -->
<lcTrueFalse id="tf1">
  <title>True - False sun rising</title>
  <lcQuestion>On Earth, the sun rises in the West and sets in the East.</lcQuestion>
  <lcAnswerOptionGroup>
    <lcAnswerOption>
      <lcAnswerContent>True</lcAnswerContent>
    </lcAnswerOption>
    <lcAnswerOption>
      <lcAnswerContent>False</lcAnswerContent>
      <lcCorrectResponse/>
    </lcAnswerOption>
  </lcAnswerOptionGroup>
  <lcFeedbackIncorrect>No, look to the East.</lcFeedbackIncorrect>
  <lcFeedbackCorrect>Yes, look to the East.</lcFeedbackCorrect>
</lcTrueFalse>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of @id, given below) and *outputclass* (see page 0).

@id *(REQUIRED)*

Defines an ID by which the element may be referenced. This attribute is defined with the XML Data Type NMTOKEN.

4.11.5.2.5 <lcSingleSelect>

The <lcSingleSelect> element was deprecated in DITA version 1.3. It was replaced with the <lcSingleSelect2> element. It was originally intended to present three or more choices, only one of which is correct.

**Inheritance**

+ topic/fig learningInteractionBase-d/lcInteractionBase learning-d/lcSingleSelect

**Example**

```xml
<!-- Single select Interaction -->
<lcSingleSelect id="singleselect">
  <title>Multiple Choice - IEEE standards trivia</title>
  <lcQuestion>Which one of the listed standards committees is responsible for developing the token ring specification?</lcQuestion>
  <lcAnswerOptions>
    <lcAnswerOption>
      <lcAnswerContent>IEEE 802.3</lcAnswerContent>
      <lcFeedback>Sorry. A little low.</lcFeedback>
    </lcAnswerOption>
    <lcAnswerOption>
      <lcAnswerContent>IEEE 802.5</lcAnswerContent>
      <lcFeedbackResponse/>
    </lcAnswerOption>
    <lcAnswerOption>
      <lcAnswerContent>IEEE 802.6</lcAnswerContent>
      <lcFeedback>That's the one.</lcFeedback>
    </lcAnswerOption>
  </lcAnswerOptions>
</lcSingleSelect>
```
Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below) and outputclass (see page 0).

@id (REQUIRED)

Defines an ID by which the element may be referenced. This attribute is defined with the XML Data Type NMTOKEN.

4.11.5.2.6 <lcMultipleSelect>

The <lcMultipleSelect> element was deprecated in DITA version 1.3. It was replaced with the <lcMultipleSelect2> element. It was originally intended to provide an interaction where the learner must indicate two or more correct answers from a list of choices.

Inheritance

+ topic/fig learningInteractionBase-d/lcInteractionBase learning-d/lcMultipleSelect

Example

<!--Multiple select Interaction -->
<lcMultipleSelect
  id="ms1">
  <title>Finding Major League Baseball logos</title>
  <lcQuestion>Which one of the following is a logo of a Major League Baseball team? (You may choose more than one.)</lcQuestion>
  <lcAnswerOptionGroup>
    <lcAnswerOption id="A">
      <lcAnswerContent>
        <image href="logo1.gif"/>
      </lcAnswerContent>
      <lcCorrectResponse/>
      <lcFeedback>Yes, that's one.</lcFeedback>
    </lcAnswerOption>
    <lcAnswerOption id="B">
      <lcAnswerContent>
        <image href="logo2.gif"/>
      </lcAnswerContent>
      <lcCorrectResponse/>
      <lcFeedback>Yes, that's one.</lcFeedback>
    </lcAnswerOption>
    <lcAnswerOption id="C">
      <lcAnswerContent>
        <image href="logo3.gif"/>
      </lcAnswerContent>
      <lcFeedback>No, not that one. Sorry!</lcFeedback>
    </lcAnswerOption>
    <lcAnswerOption id="D">
      <lcAnswerContent>
        <image href="logo4.gif"/>
      </lcAnswerContent>
      <lcFeedback>No, not that one. Sorry!</lcFeedback>
    </lcAnswerOption>
  </lcAnswerOptionGroup>
</lcMultipleSelect>
Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of @id, given below) and *outputclass* (see page 0).

@id (REQUIRED)

Defines an ID by which the element may be referenced. This attribute is defined with the XML Data Type NMTOKEN.

4.11.5.2.7 <lcSequencing>

The `<lcSequencing>` element was deprecated in DITA version 1.3. It was replaced with the `<lcSequencing2>` element. It was originally intended to ask the learner to arrange a list of choices into a predefined order, such as small to large.

Inheritance

+ topic/fig learningInteractionBase-d/lcInteractionBase learning-d/lcSequencing

Example

```xml
<lcSequencing id="sequencing">
  <title>Sequencing City Populations in the U.S.</title>
  <lcQuestion>Order the following U.S. cities according to population, from largest to smallest.</lcQuestion>
  <lcSequenceOptionGroup>
    <lcSequenceOption>
      <lcAnswerContent>Portland, Oregon</lcAnswerContent>
      <lcSequence name="lcSequence" value="2"/>
    </lcSequenceOption>
    <lcSequenceOption>
      <lcAnswerContent>Chicago, Illinois</lcAnswerContent>
      <lcSequence name="lcSequence" value="1"/>
    </lcSequenceOption>
    <lcSequenceOption>
      <lcAnswerContent>Portland, Maine</lcAnswerContent>
      <lcSequence name="lcSequence" value="4"/>
    </lcSequenceOption>
    <lcSequenceOption>
      <lcAnswerContent>Syracuse, New York</lcAnswerContent>
      <lcSequence name="lcSequence" value="3"/>
    </lcSequenceOption>
  </lcSequenceOptionGroup>
  <lcFeedbackIncorrect>No, try again, please.</lcFeedbackIncorrect>
  <lcFeedbackCorrect>Very good.</lcFeedbackCorrect>
</lcSequencing>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of @id, given below) and *outputclass* (see page 0).

@id (REQUIRED)

Defines an ID by which the element may be referenced. This attribute is defined with the XML Data Type NMTOKEN.
4.11.5.2.8 `<lcMatching>`

The `<lcMatching>` element was deprecated in DITA version 1.3. It was replaced with the `<lcMatching2>` element. It was originally intended to allow the learner to identify the correct choice that matches another choice and, optionally, any feedback for the pair or for correct or incorrect matches.

*Inheritance*

+ `topic/fig learningInteractionBase-d/lcInteractionBase learning-d/lcMatching`

*Example*

```xml
<!--Matching Interaction -->
<lcMatching id="matching">
  <lcQuestion>Match the team with the city.</lcQuestion>
  <lcMatchTable>
    <lcMatchingHeader>
      <lcItem>Team</lcItem>
      <lcMatchingItem>City</lcMatchingItem>
    </lcMatchingHeader>
    <lcMatchingPair>
      <lcItem>Boston</lcItem>
      <lcMatchingItem>Red Sox</lcMatchingItem>
      <lcMatchingItemFeedback>
        <lcFeedbackCorrect>The Red Sox play in Boston's Fenway Park</lcFeedbackCorrect>
        <lcFeedbackIncorrect>That is not Boston's home team</lcFeedbackIncorrect>
      </lcMatchingItemFeedback>
    </lcMatchingPair>
    <lcMatchingPair>
      <lcItem>San Francisco</lcItem>
      <lcMatchingItem>Giants</lcMatchingItem>
      <lcMatchingItemFeedback>
        <lcFeedbackCorrect>The Giants play in San Francisco's Candlestick Park</lcFeedbackCorrect>
        <lcFeedbackIncorrect>That is not San Francisco's home team</lcFeedbackIncorrect>
      </lcMatchingItemFeedback>
    </lcMatchingPair>
    <lcMatchingPair>
      <lcItem>Chicago</lcItem>
      <lcMatchingItem>Cubs</lcMatchingItem>
      <lcMatchingItemFeedback>
        <lcFeedbackCorrect>The Cubs play in Chicago's Wrigley Field</lcFeedbackCorrect>
        <lcFeedbackIncorrect>That is not Chicago's home team</lcFeedbackIncorrect>
      </lcMatchingItemFeedback>
    </lcMatchingPair>
    <lcMatchingPair>
      <lcItem>Toronto</lcItem>
      <lcMatchingItem>Blue Jays</lcMatchingItem>
      <lcMatchingItemFeedback>
        <lcFeedbackCorrect>The Blue Jays play in Toronto's SkyDome</lcFeedbackCorrect>
        <lcFeedbackIncorrect>That is not Toronto's home team</lcFeedbackIncorrect>
      </lcMatchingItemFeedback>
    </lcMatchingPair>
  </lcMatchTable>
</lcMatching>
```

*Attributes*

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of `@id`, given below) and *outputclass* (see page 0).

`@id` *(REQUIRED)*

Defines an ID by which the element may be referenced. This attribute is defined with the XML Data Type `NMTOKEN`.
4.11.5.2.9 <lcHotspot>

The <lcHotspot> element was deprecated in DITA version 1.3. It was replaced with the <lcHotspot2> element. It was originally intended to support an interaction where the learner clicks on a region of the screen to indicate a choice.

Inheritance
+ topic/fig learningInteractionBase-d/lcInteractionBase learning-d/lcHotspot

Example

```xml
<!--Hotspot Interaction -->
<lcHotspot id="hotspots">
  <title>Team logos hotspot</title>
  <lcQuestion>Which of the following is the logo for the Cleveland Indians?</lcQuestion>
  <lcHotspotMap>
    <lcArea id="indians">
      <lcAreaShape>rect</lcAreaShape>
      <lcAreaCoords>0,0,50,50</lcAreaCoords>
      <lcCorrectResponse value="indians"/>
      <lcFeedback>Yes.</lcFeedback>
    </lcArea>
    <lcArea id="giants">
      <lcAreaShape>rect</lcAreaShape>
      <lcAreaCoords>50,0,100,50</lcAreaCoords>
      <lcFeedback>No.</lcFeedback>
    </lcArea>
    <lcArea id="cardinals">
      <lcAreaShape>rect</lcAreaShape>
      <lcAreaCoords>0,50,50,100</lcAreaCoords>
      <lcFeedback>No.</lcFeedback>
    </lcArea>
    <lcArea id="orioles">
      <lcAreaShape>rect</lcAreaShape>
      <lcAreaCoords>50,50,100,100</lcAreaCoords>
      <lcFeedback>No.</lcFeedback>
    </lcArea>
  </lcHotspotMap>
</lcHotspot>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) (with a narrowed definition of @id, given below) and outputclass (see page 0).

@id (REQUIRED)

Defines an ID by which the element may be referenced. This attribute is defined with the XML Data Type NMTOKEN.

4.11.6 Learning and training metadata elements

Use the learning and training metadata to describe specific characteristics of the learning content. The learning and training metadata elements provide a robust metadata model for learning objects, built upon industry standards and best practices. Defined as a domain, these elements are available for use in both the topic prolog and the map topicmeta.

4.11.6.1 <lcLom>

The <lcLom> provides a set of specialized data elements to use to specify metadata in the learning topics and learning map domain.

Inheritance

Comment by robander, 17 February 2014
DITA 1.2 defined the ancestry as "topic/data", but "metadata" is correct. I've fixed this in the spec, and notified
the TC of the change - this should be considered an Errata item.

+ topic/metadata learning-d/lom

**LOM Examples**

```xml
<learningContent id="learningcontent">
  <title>LOM samples</title>
  <shortdesc>Here are samples of LOM metadata.</shortdesc>
  <prolog>
    <lom>
      <lomStructure value="collection"></lomStructure>
      <lomCoverage>This course was first offered in ancient Rome, with no updates
needed since.</lomCoverage>
      <lomAggregationLevel value="1"></lomAggregationLevel>
      <lomTechRequirement value="ms-windows"></lomTechRequirement>
      <lomInstallationRemarks>No installation is required for this content.</lomInstallationRemarks>
      <lomOtherPlatformRequirements>no other platform requirements</lomOtherPlatformRequirements>
      <lomInteractivityType value="expositive"></lomInteractivityType>
      <lomLearningResourceType value="narrativetext"></lomLearningResourceType>
      <lomInteractivityLevel value="medium"></lomInteractivityLevel>
      <lomSemanticDensity value="medium"></lomSemanticDensity>
      <lomIntendedUserRole value="teacher"></lomIntendedUserRole>
      <lomContext value="other"></lomContext>
      <lomTypicalAgeRange value="18-22"></lomTypicalAgeRange>
      <lomTypicalDifficulty value="easy"></lomTypicalDifficulty>
      <lomTypicalLearningTime value="00:30"></lomTypicalLearningTime>
    </lom>
  </prolog>
  <learningContentbody></learningContentbody>
</learningContent>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *mapkeyref* (see page 0).

4.11.6.2 `<lomAggregationLevel>`

The `<lomAggregationLevel>` describes the functional size of the learning resource.

**Inheritance**

+ topic/data learningmeta-d/lomAggregationLevel

**Example**

See `<lom>`.

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and the attributes defined below.

@name

Name of the metadata being specified. The default value for this attribute is "lomAggregationLevel". This attribute is defined with the XML Data Type CDATA.

@value (REQUIRED)

The aggregation level for this content. Allowable values are 1, 2, 3, 4, or *Using the -dita-use-conref-target value* (see page 666).

@datatype
Datatype of the metadata being specified. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

Comment by robander, 17 February 2014
Read this while converting attribute table. Not certain what sort of values are expected here. Full text from 1.2 was "Datatype of the metadata being specified."

4.11.6.3 <lomContext>

The <lomContext> describes the typical learning environment where use of the learning object is intended to take place.

Inheritance
+ topic/data learning-d/lomContext

Example
See lcLom.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and the attributes defined below.

@name
Name of the metadata being specified. The default value for this attribute is "lomContext". This attribute is defined with the XML Data Type CDATA.

@value (REQUIRED)
The learning context for this content. Allowable values are "school", "highereducation", "training", "other", or Using the -dita-use-conref-target value (see page 666).

@datatype
Datatype of the metadata being specified. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

Comment by robander, 17 February 2014
Read this while converting attribute table. Not certain what sort of values are expected here. Full text from 1.2 was "Datatype of the metadata being specified."

4.11.6.4 <lomCoverage>

The <lomCoverage> provides learning metadata about the temporal or spatial characteristics of the content, such as historical context.

Inheritance
+ topic/data learning-d/lomCoverage

Example
See lcLom.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and the attributes defined below.

@name
Name of the metadata being specified. The default value for this attribute is "lomCoverage". This attribute is defined with the XML Data Type CDATA.

@value
The coverage description for this content. The default value for this attribute is the empty string "". This attribute is defined with the XML Data Type CDATA.

@datatype
Datatype of the metadata being specified. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

Comment by robander, 17 February 2014
Read this while converting attribute table. Not certain what sort of values are expected here. Full text from 1.2 was "Datatype of the metadata being specified."

4.11.6.5 <lomDifficulty>

The <lomDifficulty> provides learning metadata about how hard it is to work through the resource for the typical target audience.

Inheritance
+ topic/data learning-d/lomDifficulty

Example
See lcLom.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and the attributes defined below.

@name
Name of the metadata being specified. The default value for this attribute is "lomDifficulty". This attribute is defined with the XML Data Type CDATA.

@value (REQUIRED)
The difficulty for this content. Allowable values are "veryeasy", "easy", "medium", "difficult", "verydifficult", or Using the -dita-use-conref-target value (see page 666).

@datatype
The datatype for this data. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

Comment by robander, 17 February 2014
Read this while converting attribute table. Not certain what sort of values are expected here. Full text from 1.2 was "The datatype for this data."

4.11.6.6 <lomInstallationRemarks>

The <lomInstallationRemarks> provides learning metadata about how to install and needed resources.

Inheritance
+ topic/data learning-d/lomInstallationRemarks

Example
See lcLom.
**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and the attributes defined below.

@**name**

Name of the metadata being specified. The default value for this attribute is "lomInstallationRemarks". This attribute is defined with the XML Data Type CDATA.

@**value**

The installation remarks. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

@**datatype**

Datatype of the metadata being specified. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

**Comment by robander, 17 February 2014**

Read this while converting attribute table. Not certain what sort of values are expected here. Full text from 1.2 was "Datatype of the metadata being specified."

### 4.11.6.7 `<lomIntendedUserRole>`

The `<lomIntendedUserRole>` provides learning metadata about the normal user of the resource, most dominant first.

**Inheritance**

+ topic/data learning-d/lomIntendedUserRole

**Example**

See `lcLom`.

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and the attributes defined below.

@**name**

Name of the metadata being specified. The default value for this attribute is "lomIntendedUserRole". This attribute is defined with the XML Data Type CDATA.

@**value** *(REQUIRED)*

The intended user role for this content. Allowable values are "teacher", "author", "learner", "manager", or *Using the -dita-use-conref-target value* (see page 666).

@**datatype**

Datatype of the metadata being specified. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

**Comment by robander, 17 February 2014**

Read this while converting attribute table. Not certain what sort of values are expected here. Full text from 1.2 was "Datatype of the metadata being specified."
4.11.6.8 <lomInteractivityLevel>

The <lomInteractivityLevel> provides learning metadata about the level of interactivity between an end user and the learning object.

Inheritance

+ topic/data learning-d/lomInteractivityLevel

Example

See lcLom.

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and the attributes defined below.

@name

The name used to indicate this data. The default value for this attribute is "lomInteractivityLevel". This attribute is defined with the XML Data Type CDATA.

@value (REQUIRED)

The interactivity level for this content. Allowable values are "verylow", "low", "medium", "high", "veryhigh", or Using the -dita-use-conref-target value (see page 666).

@datatype

The datatype for this data. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

Comment by robander, 17 February 2014

Read this while converting attribute table. Not certain what sort of values are expected here. Full text from 1.2 was "The datatype for this data."

4.11.6.9 <lomInteractivityType>

The <lomInteractivityType> provides learning metadata about the the type of interactivity supported by the resource.

Inheritance

+ topic/data learningmeta-d/lomInteractivityType

Example

See lcLom.

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and the attributes defined below.

@name

The name used to indicate this data. The default value for this attribute is "lomInteractivityType". This attribute is defined with the XML Data Type CDATA.

@value (REQUIRED)

The interactivity type for this content. Allowable values are "active", "expositive", "mixed", "undefined", or Using the -dita-use-conref-target value (see page 666).
Comment by robander, 17 February 2014
Posted question to TC: DTD/XSD do not list a value of "undefined", not sure if it should remain.

@datatype
The datatype for this data. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

Comment by robander, 17 February 2014
Read this while converting attribute table. Not certain what sort of values are expected here. Full text from 1.2 was "The datatype for this data."

4.11.6.10 <lomLearningResourceType>

The <lomLearningResourceType> provides learning metadata about the specific kind of resource used, most dominant kind first.

Inheritance
+ topic/data learning-d/lomLearningResourceType

Example
See IcLom.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and the attributes defined below.

@name
The name used to indicate this data. The default value for this attribute is "lomLearningResourceType". This attribute is defined with the XML Data Type CDATA.

@value (REQUIRED)
The learning resource type for this content. Allowable values are exercise, simulation, questionnaire, diagram, figure, graph, index, slide, table, narrativetext, exam, experiment, problemstatement, selfassessmnet, lecture, or Using the -dita-use-conref-target value (see page 666).

Comment by robander
Changed description from "The interactivity type..." to "The learning resource type..." -- the previous value was clearly a copy/paste error from the lomInteractivityType element. Also, the DTD/XSD list a value of "lecture" that was not legal according to the spec - notified the TC to figure out if we should add it to the spec or add as "errata in the dtd/xsd". For now, listed as a valid value.

@datatype
The datatype for this data. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

Comment by robander, 17 February 2014
Read this while converting attribute table. Not certain what sort of values are expected here. Full text from 1.2 was "The datatype for this data."

4.11.6.11 <lomOtherPlatformRequirements>

The <lomOtherPlatformRequirements> provides learning metadata information about other software and hardware requirements.

Inheritance
+ topic/data learning-d/lomOtherPlatformRequirements
Example
See LcLom.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and the attributes defined below.

@name
Name of the metadata being specified. The default value for this attribute is "lomOtherPlatformRequirements". This attribute is defined with the XML Data Type CDATA.

@value
The other platform requirements for this content. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

@datatype
The datatype for this data. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

Comment by robander, 17 February 2014
Read this while converting attribute table. Not certain what sort of values are expected here. Full text from 1.2 was "The datatype for this data."

4.11.6.12 <lomSemanticDensity>

The <lomSemanticDensity> provides learning metadata about a subjective measure of the learning object's usefulness as compared to its size or duration.

Inheritance
+ topic/data learning-d/lomSemanticDensity

Example
See LcLom.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and the attributes defined below.

@name
The name used to indicate this data. The default value for this attribute is "lomSemanticDensity". This attribute is defined with the XML Data Type CDATA.

@value (REQUIRED)
The semantic density for this content. Allowable values are "verylow", "low", "medium", "high", "veryhigh", or Using the -dita-use-conref-target value (see page 666).

@datatype
The datatype for this data. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

Comment by robander, 17 February 2014
Read this while converting attribute table. Not certain what sort of values are expected here. Full text from 1.2 was "The datatype for this data."
4.11.6.13 <lomStructure>

The <lomStructure> provides learning metadata about the underlying organizational structure of the resource.

Inheritance
+ topic/data learning-d/lomStructure

Example
See lclm.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and the attributes defined below.

@name
The name used to indicate this data. The default value for this attribute is "lomStructure". This attribute is defined with the XML Data Type CDATA.

Comment by robander, 17 February 2014
Notified TC: XSD has a mispelling of "branched", DTD is missing several values. Leaving all listed for now.

@value (REQUIRED)
The learning structure for this content. Allowable values are "atomic", "branched", "collection", "hierarchical", "linear", "mixed", "networked", "parceled", or Using the -dita-use-conref-target value (see page 666).

@datatype
The datatype for this data. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

Comment by robander, 17 February 2014
Read this while converting attribute table. Not certain what sort of values are expected here. Full text from 1.2 was "The datatype for this data."

4.11.6.14 <lomTechRequirement>

The <lomTechRequirement> provides learning metadata about the operating system(s) under which the learning resource can run.

Inheritance
+ topic/data learning-d/lomTechRequirement

Example
See lclm.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and the attributes defined below.

@name
The name used to indicate this data. The default value for this attribute is "lomTechRequirement". This attribute is defined with the XML Data Type CDATA.

Comment by robander, 17 February 2014
Notified TC: XSD has a mispelling of "branched", DTD is missing several values. Leaving all listed for now.

@datatype
The datatype for this data. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

Comment by robander, 17 February 2014
Read this while converting attribute table. Not certain what sort of values are expected here. Full text from 1.2 was "The datatype for this data."
Queried TC: in 1.2, the default value was "Operating System | Browser" - but in the DTD/XSD, the default was the element name.

@value (REQUIRED)
The lomTechRequirements for this content. Allowable values are pc-dos, ms-windows, macos, unix, multi-os, none, any, netscapecommunicator, ms-internetexplorer, opera, amaya, orUsing the -dita-use-conref-target value (see page 666).

@datatype
The datatype for this data. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

Comment by robander, 17 February 2014
Read this while converting attribute table. Not certain what sort of values are expected here. Full text from 1.2 was "The datatype for this data."

4.11.6.15 <lomTypicalAgeRange>
The <lomTypicalAgeRange> provides learning metadata about the age of the typical intended user.

Inheritance
+ topic/data learning-d/lomTypicalAgeRange

Example
See lcLom.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and the attributes defined below.

@name
The name used to indicate this data. The default value for this attribute is "lomTypicalAgeRange". This attribute is defined with the XML Data Type CDATA.

@value (REQUIRED)
The age range intended for this content. This attribute is defined with the XML Data Type CDATA.

@datatype
The datatype for this data. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

Comment by robander, 17 February 2014
Read this while converting attribute table. Not certain what sort of values are expected here. Full text from 1.2 was "The datatype for this data."

4.11.6.16 <lomTypicalLearningTime>
The <lomTypicalLearningTime> provides learning metadata about the approximate or typical time it takes to work with the resource.

Inheritance
+ topic/data learning-d/lomTypicalLearningTime

Example
See lcLom.
Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and the attributes defined below.

@name

The name used to indicate this data. The default value for this attribute is "lomTypicalLearningTime". This attribute is defined with the XML Data Type CDATA.

@value

The typical learning time for this content. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

@datatype

The datatype for this data. The default value is the empty string "". This attribute is defined with the XML Data Type CDATA.

Comment by robander, 17 February 2014

Read this while converting attribute table. Not certain what sort of values are expected here. Full text from 1.2 was "The datatype for this data."

4.11.7 Other learning and training elements

This group includes those elements that are specifically related to learning, but do not fit in to one of the other major learning groups.

4.11.7.1 Common interactions elements

Elements in this group are designed specifically to describe learning and training interactions.

4.11.7.1.1 <lcAnswerContent>

The <lcAnswerContent> element was deprecated in DITA version 1.3. It was replaced with the <lcAnswerContent2> element. It was originally intended to provide the content for an answer option, which the learner can select as correct or incorrect, in a learning interaction.

Inheritance

+ topic/p learningInteractionBase-d/p learning-d/lcAnswerContent

Example

```
...
  <lcAnswerContent>True</lcAnswerContent>
...
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.7.1.2 <lcAnswerOption>

The <lcAnswerOption> element was deprecated in DITA version 1.3. It was replaced with the <lcAnswerOption2> element. It was originally intended to provide the content and feedback for a question option, and can indicate the correct option in a learning interaction.

Inheritance

+ topic/li learningInteractionBase-d/li learning-d/lcAnswerOption
Example

```xml
...<lcAnswerOption>
  <lcAnswerContent>True</lcAnswerContent>
  <lcCorrectResponse/>
</lcAnswerOption>
...```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.11.7.1.3 <lcAnswerOptionGroup>

The `<lcAnswerOptionGroup>` element was deprecated in DITA version 1.3. It was replaced with the `<lcAnswerOptionGroup2>` element. It was originally intended to provide a container for the options for a true-false, single-select, or multiple-select assessment interaction.

Inheritance

+ topic/ul learningInteractionBase-d/ul learning-d/lcAnswerOptionGroup

Example

```xml
...<lcAnswerOptionGroup>
  <lcAnswerOption>
    <lcAnswerContent>True</lcAnswerContent>
    <lcCorrectResponse/>
  </lcAnswerOption>
  <lcAnswerOption>
    <lcAnswerContent>False</lcAnswerContent>
  </lcAnswerOption>
</lcAnswerOptionGroup>
...```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.11.7.1.4 <lcArea>

A `<lcArea>` was deprecated in DITA version 1.3. It was replaced with the `<lcArea2>` element. It was originally intended to define an area of a hotspot image that contains a correct or incorrect choice in a hotspot assessment interaction.

Inheritance

+ topic/figgroup learningInteractionBase-d/figgroup learning-d/lcArea

Example

```xml
...<lcArea id="cardinals">
  <lcAreaShape>rect</lcAreaShape>
  <lcAreaCoords>0,50,50,100</lcAreaCoords>
  <lcFeedback>No.</lcFeedback>
</lcArea>
```
Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.11.7.1.5 `<lcAreaCoords>`

The `<lcAreaCoords>` element was deprecated in DITA version 1.3. It was replaced with the `<lcAreaCoords2>` element. It was originally intended to specify the coordinates of a linkable hotspot in a learning image.

This element contains text data representing coordinate data for learning images with linkable hotspots. Pixels are the recommended units for describing coordinates. The syntax of the coordinate data depends on the shape described by the coordinates, and is based on the image map definition in HTML. It uses the following data for the appropriate shapes:

**Shape**

- *Data format*
  - *rect* left-x, top-y, right-x, bottom-y
  - *circle* center-x, center-y, radius
  - *poly* x1, y1, x2, y2, ..., xN, yN. The first x and y coordinate pair and the last should be the same to close the polygon.

**Inheritance**

+ topic/ph learningInteractionBase-d/ph learning-d/lcAreaCoords

**Example**

```xml
<lcHotspotMap>
  <image href="hotlogos.gif">
    <alt>Baseball team logos</alt>
  </image>
  <lcArea id="giants">
    <lcAreaShape>rect</lcAreaShape>
    <lcAreaCoords>0,50,50,100</lcAreaCoords>
    <lcFeedback>No.</lcFeedback>
  </lcArea>
  <lcArea id="orioles">
    <lcAreaShape>rect</lcAreaShape>
    <lcAreaCoords>50,50,100,100</lcAreaCoords>
    <lcFeedback>No.</lcFeedback>
  </lcArea>
</lcHotspotMap>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) (with a narrowed definition of `@translate`, given below), *outputclass* (see page 0), and `@keyref`.

@translate

Indicates whether the content of the element should be translated or not. For this element the default value is "no". Setting to "yes" will override the default. The DITA architectural specification contains a list of each OASIS DITA element and its common processing default for the translate value; because this element uses an actual default, it will always be treated as `translate="no"` unless overridden as described. Available values are:
The <lcAreaShape> element was deprecated in DITA version 1.3. It was replaced with the <lcAreaShape2> element. It was originally intended to define the shape of a linkable hotspot in a learning image.

The <lcAreaShape> element supports these values:

- **rect**
  Define a rectangular region. If you leave the `<shape>` element blank, this is assumed.

- **circle**
  Define a circular region.

- **poly**
  Define a polygonal region.

- **default**
  Indicates the entire diagram.

### Inheritance

+ topic/keyword learningInteractionBase-d/keyword learning-d/lcAreaShape

### Example

```xml
<lcHotspotMap>
    <image href="hotlogos.gif">
        <alt>Baseball team logos</alt>
    </image>
    <lcArea id="giants">
        <lcAreaShape>rect</lcAreaShape>
        <lcAreaCoords>0,50,50,100</lcAreaCoords>
        <lcFeedback>No.</lcFeedback>
    </lcArea>
    <lcArea id="orioles">
        <lcAreaShape>rect</lcAreaShape>
        <lcAreaCoords>50,50,100,100</lcAreaCoords>
        <lcFeedback>No.</lcFeedback>
    </lcArea>
</lcHotspotMap>
```

### Attributes

The following attributes are available on this element: [Universal attribute group](see page 650) (with a narrowed definition of `@translate`, given below), [outputclass](see page 0), and [keyref].

- **@translate**
  Indicates whether the content of the element should be translated or not. For this element the default value is "no". Setting to "yes" will override the default. The DITA architectural specification contains a list of each OASIS DITA element and its common processing default for the translate value; because this element uses an actual default, it will always be treated as `translate="no"` unless overridden as described. Available values are:

  - **no**
    The content of this element is not translateable.
4.11.7.1.7 <lcAsset>

The <lcAsset> element was deprecated in DITA version 1.3. It was replaced with the <lcAsset2> element. It was originally intended to provide the images or other graphic assets to support the interaction.

**Inheritance**
+ topic/p learningInteractionBase-d/p learning-d/lcAsset

**Example**

```xml
.. 
<lcAsset>
  <image href="sunrise.gif">
    <alt>the sun rising on Earth</alt>
  </image>
</lcAsset>
```

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.11.7.1.8 <lcCorrectResponse>

The <lcCorrectResponse> element was deprecated in DITA version 1.3. It was replaced with the <lcCorrectResponse2> element. It was originally intended to indicate a correct response in an assessment interaction.

In an <lcHotspot> interaction, indicate the correct hotspot by setting the @value attribute of <lcCorrectResponse> to the id of the "correct" area in the image map.

**Inheritance**
+ topic/data learningInteractionBase-d/data learning-d/lcCorrectResponse

**Example**

```xml
.. 
<lcAnswerOption>
  <lcAnswerContent>True</lcAnswerContent>
  <lcCorrectResponse/>
</lcAnswerOption>
```

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and the attributes defined below.

@name

The name used to indicate this data. The default value for this attribute is "lcCorrectResponse".
@value (REQUIRED)

The value. In an <lcHotspot> interaction, set the value to the id corresponding to the correct area in the image map.

4.11.7.1.9 <lcFeedback>

The <lcFeedback> element was deprecated in DITA version 1.3. It was replaced with the <lcFeedback2> element. It was originally intended to provide information to the learner about a correct or incorrect response in an assessment interaction.

Inheritance

+ topic/p learningInteractionBase-d/p learning-d/lcFeedback

Example

```xml
...<lcFeedback>Yes, look to the East.</lcFeedback>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.7.1.10 <lcFeedbackCorrect>

The <lcFeedbackCorrect> element was deprecated in DITA version 1.3. It was replaced with the <lcFeedbackCorrect2> element. It was originally intended to provide feedback to the learner about a correct response in an assessment interaction.

Inheritance

+ topic/p learningInteractionBase-d/p learning-d/lcFeedbackCorrect

Example

```xml
...<lcFeedbackIncorrect>No, look to the East.</lcFeedbackIncorrect>
<lcFeedbackCorrect>Yes, look to the East.</lcFeedbackCorrect>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.7.1.11 <lcFeedbackIncorrect>

The <lcFeedbackIncorrect> element was deprecated in DITA version 1.3. It was replaced with the <lcFeedbackIncorrect2> element. It was originally intended to provide feedback about incorrect response in an assessment interaction.

Inheritance

+ topic/p learningInteractionBase-d/p learning-d/lcFeedbackIncorrect
Example

.. :
   <lcFeedbackIncorrect> No, look to the East.</lcFeedbackIncorrect>
   <lcFeedbackCorrect> Yes, look to the East.</lcFeedbackCorrect>
.. :

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.11.7.1.12 `<lcHotspotMap>`

The `<lcHotspotMap>` was deprecated in DITA version 1.3. It was replaced with the `<lcHotspotMap2>` element. It was originally intended to let you designate an action area or region over an image, allowing a click in that region to get scored as correct or incorrect in response to an interaction question.

Inheritance

+ topic/fig learningInteractionBase-d/figgroup learning-d/lcHotspotMap

Example

```dita
<!--Hotspot Interaction -->
<lchotspot id="hotspots">
  <title>Team logos hotspot</title>
  <lcQuestion>Which of the following is the logo for the Cleveland Indians?</lcQuestion>
  <lcHotspotMap>
    <image href="hotlogos.gif"/>
    <lcArea id="indians">
      <lcAreaShape>rect</lcAreaShape>
      <lcAreaCoords>0,0,50,50</lcAreaCoords>
      <lcCorrectResponse value="indians"/>
      <lcFeedback>Yes.</lcFeedback>
    </lcArea>
    <lcArea id="giants">
      <lcAreaShape>rect</lcAreaShape>
      <lcAreaCoords>50,0,100,50</lcAreaCoords>
      <lcFeedback>No.</lcFeedback>
    </lcArea>
    <lcArea id="cardinals">
      <lcAreaShape>rect</lcAreaShape>
      <lcAreaCoords>0,50,50,100</lcAreaCoords>
      <lcFeedback>No.</lcFeedback>
    </lcArea>
    <lcArea id="orioles">
      <lcAreaShape>rect</lcAreaShape>
      <lcAreaCoords>50,50,100,100</lcAreaCoords>
      <lcFeedback>No.</lcFeedback>
    </lcArea>
  </lcHotspotMap>
</lchotspot>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).
4.11.7.1.13 <lcItem>

The `<lcItem>` element was deprecated in DITA version 1.3. It was replaced with the `<lcItem2>` element. It was originally intended to provide the content for an item that matches the match item in a match table in an assessment interaction.

**Inheritance**

+ topic/stentry learningInteractionBase-d/stentry learning-d/lcItem

**Example**

```xml
<lcMatching id="matching">
  <title>Matching teams with cities</title>
  <lcQuestion>Match the team with the city.</lcQuestion>
  <lcMatchTable>
    <lcMatchingHeader>
      <lcItem>Team</lcItem>
      <lcMatchingItem>City</lcMatchingItem>
    </lcMatchingHeader>
    <lcMatchingPair>
      <lcItem>Boston</lcItem>
      <lcMatchingItem>Red Sox</lcMatchingItem>
    </lcMatchingPair>
    <lcMatchingPair>
      <lcItem>San Francisco</lcItem>
      <lcMatchingItem>Giants</lcMatchingItem>
    </lcMatchingPair>
    <lcMatchingPair>
      <lcItem>Chicago</lcItem>
      <lcMatchingItem>Cubs</lcMatchingItem>
    </lcMatchingPair>
    <lcMatchingPair>
      <lcItem>Toronto</lcItem>
      <lcMatchingItem>Blue Jays</lcMatchingItem>
    </lcMatchingPair>
  </lcMatchTable>
  <lcFeedbackCorrect>Good job.</lcFeedbackCorrect>
  <lcFeedbackIncorrect>Not quite.</lcFeedbackIncorrect>
</lcMatching>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.11.7.1.14 <lcMatchingHeader>

The `<lcMatchingHeader>` element was deprecated in DITA version 1.3. It was replaced with the `<lcMatchingHeader2>` element. It was originally intended to provide column headings for items to present in a matching table.

**Inheritance**

+ topic/sthead learningInteractionBase-d/sthead learning-d/lcMatchingHeader

**Example**

...

```xml
...<lcMatchingHeader>
  <lcItem>Team</lcItem>
  <lcMatchingItem>City</lcMatchingItem>
</lcMatchingHeader>
...```
4.11.7.1.15 <lcMatchingItem>

The <lcMatchingItem> element was deprecated in DITA version 1.3. It was replaced with the <lcMatchingItem2> element. It was originally intended to provide the content for the matching side of a matching pair of items in a match table interaction.

Example

```
...<lcMatchingPair>
  <lcItem>Boston</lcItem>
  <lcMatchingItem>Red Sox</lcMatchingItem>
</lcMatchingPair>
...
```

4.11.7.1.16 <lcMatchingItemFeedback>

The <lcMatchingItemFeedback> element was deprecated in DITA version 1.3. It was replaced with the <lcMatchingItemFeedback2> element. It was originally intended to provide feedback on the match as a whole or on correct and incorrect matches or both.

Example

```
...<lcMatchingPair>
  <lcItem>Toronto</lcItem>
  <lcMatchingItem>Blue Jays</lcMatchingItem>
  <lcMatchingItemFeedback>
    <lcFeedbackCorrect>The Blue Jays play in Toronto's SkyDome</lcFeedbackCorrect>
    <lcFeedbackIncorrect>That is not Toronto's home team.</lcFeedbackIncorrect>
  </lcMatchingItemFeedback>
</lcMatchingPair>
...```

Attributes

The following attributes are available on this element: 
Universal attribute group (see page 650) and outputclass (see page 0 ).
4.11.7.1.17 <lcMatchingPair>

The <lcMatchingPair> element was deprecated in DITA version 1.3. It was replaced with the <lcMatchingPair2> element. It was originally intended to provide a table row with the pair of items that comprise a correct match in a matching interaction and, optionally, feedback on the pair or correct and incorrect matches.

Inheritance
+ topic/strow learningInteractionBase-d/strow learning-d/lcMatchingPair

Example

...<lcMatchingPair>
  <lcItem>Boston</lcItem>
  <lcMatchingItem>Red Sox</lcMatchingItem>
  <lcMatchingItemFeedback>
    <lcFeedbackCorrect>The Red Sox play in Boston's Fenway Park</lcFeedbackCorrect>
    <lcFeedbackCorrect>That is not Boston's home team</lcFeedbackCorrect>
  </lcMatchingItemFeedback>
</lcMatchingPair>
...
Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.7.1.19 <lcOpenAnswer>
The <lcOpenAnswer> element was deprecated in DITA version 1.3. It was replaced with the <lcOpenAnswer2> element. It was originally intended to provide a suggested answer for an <lcOpenQuestion> interaction.

Inheritance
+ topic/p learningInteractionBase-d/p learning-d/lcOpenAnswer

Example

<lcOpenAnswer>Cow's can only jump over the moon in nursery rhymes.</lcOpenAnswer>

4.11.7.1.20 <lcQuestion>
The <lcQuestion> element was deprecated in DITA version 1.3. It was replaced with the <lcQuestion2> element. It was originally intended to ask the question in an interaction.

Inheritance
+ topic/p learningInteractionBase-d/lcQuestionBase learning-d/lcQuestion

Example

<lcQuestion>Which one of the following is a logo of a Major League Baseball team? (You may choose more than one.)</lcQuestion>

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).
4.11.7.1.21 <lcSequence>

The <lcSequence> element was deprecated in DITA version 1.3. It was replaced with the <lcSequence2> element. It was originally intended to provide the position of a sequence option in a sequence.

Inheritance
+ topic/data learningInteractionBase-d/data learning-d/lcSequence

Example

```xml
<lcSequencing id="sequencing">
  <title>Sequencing City Populations in the U.S.</title>
  <lcQuestion>Order the following U.S. cities according to population, from largest to smallest.</lcQuestion>
  <lcSequenceOptionGroup>
    <lcSequenceOption>
      <lcAnswerContent>Portland, Oregon</lcAnswerContent>
      <lcSequence name="lcSequence" value="2"/>
    </lcSequenceOption>
    <lcSequenceOption>
      <lcAnswerContent>Chicago, Illinois</lcAnswerContent>
      <lcSequence name="lcSequence" value="1"/>
    </lcSequenceOption>
    <lcSequenceOption>
      <lcAnswerContent>Portland, Maine</lcAnswerContent>
      <lcSequence name="lcSequence" value="4"/>
    </lcSequenceOption>
    <lcSequenceOption>
      <lcAnswerContent>Syracuse, New York</lcAnswerContent>
      <lcSequence name="lcSequence" value="3"/>
    </lcSequenceOption>
  </lcSequenceOptionGroup>
  <lcFeedbackIncorrect>No, try again, please.  </lcFeedbackIncorrect>
  <lcFeedbackCorrect>Very good.</lcFeedbackCorrect>
</lcSequencing>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and the attributes defined below.

@name

The name used to indicate this data. The default value for this attribute is "lcSequence".

@value (REQUIRED)

The position of an option in a sequence of options. The value MUST be a number.

@datatype (REQUIRED)

The datatype for this data. Use numbers for sequence.

Comment by robander, 18 February 2014

Read this while converting attribute table. There is something wrong here. The attribute table says "required", but I cannot tell from the description what is supposed to go in the attribute. The example above shows <lcSequence> elements that do not use htis attribute. I've sent email to the TC to clarify.

4.11.7.1.22 <lcSequenceOption>

The <lcSequenceOption> element was deprecated in DITA version 1.3. It was replaced with the <lcSequenceOption2> element. It was originally intended to provide the contents of an item in a sequence interaction.

Inheritance
+ topic/li learningInteractionBase-d/li learning-d/lcSequenceOption
Example

```xml
<lcSequencing id="sequencing">
  <title>Sequencing City Populations in the U.S.</title>
  <lcQuestion>Order the following U.S. cities according to population, from largest to smallest.</lcQuestion>
  <lcSequenceOptionGroup>
    <lcSequenceOption>
      <lcAnswerContent>Portland, Oregon</lcAnswerContent>
      <lcSequence name="lcSequence" value="2"/>
    </lcSequenceOption>
    <lcSequenceOption>
      <lcAnswerContent>Chicago, Illinois</lcAnswerContent>
      <lcSequence name="lcSequence" value="1"/>
    </lcSequenceOption>
    <lcSequenceOption>
      <lcAnswerContent>Portland, Maine</lcAnswerContent>
      <lcSequence name="lcSequence" value="4"/>
    </lcSequenceOption>
    <lcSequenceOption>
      <lcAnswerContent>Syracuse, New York</lcAnswerContent>
      <lcSequence name="lcSequence" value="3"/>
    </lcSequenceOption>
  </lcSequenceOptionGroup>
  <lcFeedbackIncorrect>No, try again, please.</lcFeedbackIncorrect>
  <lcFeedbackCorrect>Very good.</lcFeedbackCorrect>
</lcSequencing>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.7.1.23 <lcSequenceOptionGroup>

The <lcSequenceOptionGroup> element was deprecated in DITA version 1.3. It was replaced with the <lcSequenceOptionGroup2> element. It was originally intended to provide the options for an assessment sequence interaction.

Inheritance

+ topic/ol learningInteractionBase-d/ol learning-d/lcSequenceOptionGroup

Example

```xml
<lcSequencing id="sequencing">
  <title>Sequencing City Populations in the U.S.</title>
  <lcQuestion>Order the following U.S. cities according to population, from largest to smallest.</lcQuestion>
  <lcSequenceOptionGroup>
    <lcSequenceOption>
      <lcAnswerContent>Portland, Oregon</lcAnswerContent>
      <lcSequence name="lcSequence" value="2"/>
    </lcSequenceOption>
    <lcSequenceOption>
      <lcAnswerContent>Chicago, Illinois</lcAnswerContent>
      <lcSequence name="lcSequence" value="1"/>
    </lcSequenceOption>
    <lcSequenceOption>
      <lcAnswerContent>Portland, Maine</lcAnswerContent>
      <lcSequence name="lcSequence" value="4"/>
    </lcSequenceOption>
    <lcSequenceOption>
      <lcAnswerContent>Syracuse, New York</lcAnswerContent>
      <lcSequence name="lcSequence" value="3"/>
    </lcSequenceOption>
  </lcSequenceOptionGroup>
  <lcFeedbackIncorrect>No, try again, please.</lcFeedbackIncorrect>
</lcSequencing>
```
Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.11.7.2 Common topic elements

The learning and training content elements provide specialized content for learning topics.

4.11.7.2.1 `<lcAge>`

The `<lcAge>` provides the age range of the intended learner audience, for use by curriculum developers and course planners.

**Inheritance**
- `topic/p learningBase/p learningPlan/lcAge`

**Example**

```xml
...<lcAge>Adults age 30 - 45.</lcAge>
...
```

See *lcPlanAudience*.

Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.11.7.2.2 `<lcAssessment>`

The `<lcAssessment>` describes assessment plans.

**Inheritance**
- `topic/p learningBase/p learningPlan/lcAssessment`

**Example**

```xml
...<lcAssessment>The module includes a pass/fail assessment exam.</lcAssessment>
...
```

Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).
4.11.7.2.3 <lcAttitude>

The <lcAttitude> describes mental state that influences the choices of personal actions.

**Inheritance**
- topic/p learningBase/p learningPlan/lcAttitude

**Example**

```xml
...<lcAttitude>Learners must be willing to be open and flexible.</lcAttitude>...
```

**Attributes**

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.7.2.4 <lcAudience>

The <lcAudience> describes characteristics of the learners who take the instruction.

**Inheritance**
- topic/section learningBase/lcAudience

**Example**

```xml
...<lcAudience>
  <title>Audience</title>
  Food connoisseurs and chefs
</lcAudience>
...
```

**Attributes**

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and spectitle (see page 0).

4.11.7.2.5 <lcBackground>

The <lcBackground> provides the learners' professional background and the relevancy to the learning plan.

**Inheritance**
- topic/p learningBase/p learningPlan/lcBackground

**Example**

```xml
...<lcBackground>Experience with all phases of systems financial analysis and planning.</lcBackground>...
```
**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650) and *
outputclass* (see page 0).

**4.11.7.2.6 <lcChallenge>**

The `<lcChallenge>` refers to what it is that you want the student to practice. For example, if you're studying network diagrams, and challenge might be "see if you can put this network into its proper sequence" or "see if you understand this network flow".

**Inheritance**
- topic/section learningBase/lcChallenge

**Example**

```xml
<learningContent id="overview">  
  <title>Learning Content topic</title>  
  <learningContentbody>  
    <lcChallenge><title>Challenge</title>  
      <p>Describe the challenge.</p>  
    </lcChallenge>  
  </learningContentbody>  
</learningContent>
```

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650), *
outputclass* (see page 0), and *
spectitle* (see page 0).

**4.11.7.2.7 <lcCIN>**

The `<lcCIN>` provides a course identification number or other alternate identifier for the project title.

**Inheritance**
- topic/fig learningBase/fig learningPlan/lcCIN

**Example**

```xml
...  
  <lcCIN>  
    <title>CIN</title>  
    <p>A life without CIN is an unexciting life, indeed.</p>  
  </lcCIN>  
...  
```

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650), *
Display attribute group* (see page 660), *
outputclass* (see page 0), and *
spectitle* (see page 0).

**4.11.7.2.8 <lcClassroom>**

The `<lcClassroom>` describes the classroom environment.

**Inheritance**
- topic/fig learningBase/fig learningPlan/lcClassroom
Example

```
...<lcClassroom>
  <title>Classroom Setting</title>
  <p>Lecture hall with 500 seats and stage.</p>
</lcClassroom>
...```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Display attribute group* (see page 660), *outputclass* (see page 0), and *spectitle* (see page 0).

**4.11.7.2.9 <lcClient>**

The `<lcClient>` provides the person or organization sponsoring or requiring the learning event development.

**Inheritance**

- `topic/fig learningBase/fig learningPlan/lcClient`

Example

```
...<lcClient>
  <title>Client</title>
  <p>Los Angeles County K-5 elementary schools.</p>
</lcClient>
...```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Display attribute group* (see page 660), *outputclass* (see page 0), and *spectitle* (see page 0).

**4.11.7.2.10 <lcConstraints>**

The `<lcConstraints>` describes the organizational or technical aspects that may limit the organization's ability to effectively use the instruction to meet its goals.

**Inheritance**

- `topic/fig learningBase/section learningPlan/lcConstraints`

Example

```
...<lcConstraints>
  <title>Constraints</title>
  <p>Imagination.</p>
</lcConstraints>
...```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Display attribute group* (see page 660), *outputclass* (see page 0), and *spectitle* (see page 0).
4.11.7.2.11 \(<lcDelivDate>\)

The \(<lcDelivDate>\) provides the project delivery date.

Inheritance
- topic/fig learningBase/fig learningPlan/lcDelivDate

Example

```xml
...<lcDelivDate>20070630</lcDelivDate>
...```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Display attribute group (see page 660), outputclass (see page 0), and spectitle (see page 0).

4.11.7.2.12 \(<lcDelivery>\)

The \(<lcDelivery>\) describes the delivery method for this learning content.

Inheritance
- topic/p learningBase/p learningPlan/lcDelivery

Example

```xml
...<lcDelivery>Delivery shall be fast and to the point.</lcDelivery>
...```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.7.2.13 \(<lcDownloadTime>\)

The \(<lcDownloadTime>\) describes the maximum time allowed for download time in the client's delivery environment.

Inheritance
- topic/fig learningBase/fig learningPlan/lcDownloadTime

Example

```xml
...<lcDownloadTime>
<title>DownloadTime</title>
<p>On-board download in no more than 15 seconds at all times.</p>
</lcDownloadTime>
...```
Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Display attribute group (see page 660), outputclass (see page 0), and spectitle (see page 0).

4.11.7.2.14 <lcDuration>

The <lcDuration> provides an estimated duration for the learning activity.

Inheritance
- topic/section learningBase/lcDuration

Example

```xml
...<lcDuration><title>Duration</title>
    <lcTime name="lcTime" value="00:15"/>
</lcDuration>
...
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and spectitle (see page 0).

4.11.7.2.15 <lcEdLevel>

The <lcEdLevel> provides the range of learners' education level and the relevancy to the learning plan.

Inheritance
- topic/p learningBase/p learningPlan/lcEdLevel

Example

```xml
...<lcEdLevel>HS or equivalent education level is assumed.</lcEdLevel>
...
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.7.2.16 <lcFileSizeLimitations>

The <lcFileSizeLimitations> describes any file size limitation in the download environment.

Inheritance
- topic/fig learningBase/fig learningPlan/lcFileSizeLimitations

Example

```xml
...<lcFileSizeLimitations>
    <title>File Size Limitations</title>
    <p>Assume very large and complex files at all times.</p>
</lcFileSizeLimitations>
```
Attributes
The following attributes are available on this element: *Universal attribute group* (see page 650), *Display attribute group* (see page 660), *outputclass* (see page 0), and *spectitle* (see page 0).

4.11.7.2.17 `<lcGapAnalysis>`

The `<lcGapAnalysis>` compares existing learning objectives to current job task analysis.

**Inheritance**
- topic/section learningBase/section learningPlan/lcGapAnalysis

**Example: lcGapAnalysis**

```xml
...<lcGapAnalysis>
    <title>Gap Analysis</title>
    <lcGapItem>
        <title>Gap item</title>
        <lcPlanObjective>Learn the basics before you go further.</lcPlanObjective>
        <lcJtaItem>Current learning objectives provide guidance to developing small modules with terminal learning objectives.</lcJtaItem>
        <lcGapItemDelta>Current learners do not have the capability to approach learning development in a modular strategy that saves time and costs across the enterprise.</lcGapItemDelta>
    </lcGapItem>
</lcGapAnalysis>
...```

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *spectitle* (see page 0).

4.11.7.2.18 `<lcGapItem>`

The `<lcGapItem>` describes gaps between existing training objectives and related job-task-analysis content.

**Inheritance**
- topic/fig learningBase/fig learningPlan/lcGapItem

**Example**

```xml
...<lcGapItem>
    <title>Gap item</title>
    <lcPlanObjective>Learn the basics before you go further.</lcPlanObjective>
    <lcJtaItem>Current learning objectives provide guidance to developing small modules with terminal learning objectives.</lcJtaItem>
    <lcGapItemDelta>Current learners do not have the capability to approach learning development in a modular strategy that saves time and costs across the enterprise.</lcGapItemDelta>
</lcGapItem>
...```
Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Display attribute group (see page 660), outputclass (see page 0), and spectitle (see page 0).

4.11.7.2.19 <lcGapItemDelta>
The <lcGapItemDelta> describes the gap between the learning objective and the task analysis.

Inheritance
- topic/p learningBase/p learningPlan/lcGapItemDelta

Example: lcGapAnalysis

...<lcGapItemDelta>Current learners do not have the capability to approach learning development in a modular strategy that saves time and costs across the enterprise.</lcGapItemDelta>...

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.7.2.20 <lcGeneralDescription>
The <lcGeneralDescription> provides a space to develop a general description about the organization's training needs.

Inheritance
- topic/p learningBase/p learningPlan/lcGeneralDescription

Example

...<lcGeneralDescription>Generally speaking, this is what to expect.</lcGeneralDescription>...

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.7.2.21 <lcGoals>
The <lcGoals> provides the outcomes desired by the organization to be addressed by the training effort. These goals may require concurrent efforts outside of training such as technology acquisition, reorganization, and so forth.

Inheritance
- topic/p learningBase/p learningPlan/lcGoals
Example

```
... <lcGoals>Goals...</lcGoals>
...
```

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

**4.11.7.2.22 <lcGraphics>**

The `<lcGraphics>` describes standards and system requirements for displaying graphics and other related content types.

**Inheritance**
- topic/fig learningBase/fig learningPlan/lcGraphics

**Example**

```
... <lcGraphics><p>Expect many graphics.</p></lcGraphics>
...
```

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650), *Display attribute group* (see page 660), *outputclass* (see page 0), and *spectitle* (see page 0).

**4.11.7.2.23 <lcHandouts>**

The `<lcHandouts>` provides aspects of the course that are provided by the instructor in support of the course learning objectives.

**Inheritance**
- topic/fig learningBase/fig learningPlan/lcHandouts

**Example**

```
... <lcHandouts>
  <title>Handouts</title>
  <p>This course may have handouts.</p>
</lcHandouts>
...
```

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650), *Display attribute group* (see page 660), *outputclass* (see page 0), and *spectitle* (see page 0).

**4.11.7.2.24 <lcInstruction>**

The `<lcInstruction>` describes the specifics of a learning activity.

You have two options for including instructive content in a `<learningContent>` topic.
• When you want to include content directly in the body of the `<learningContent>` topic, use `<lcInstruction>`.

• When you want to use content that already exists in other DITA topics, or when you want to take advantage of additional topic types, such as concept, task, or reference, you can include that content as additional nested topic content in the `<learningContent>` topic, immediately after the `<learningContentbody>`.

**Inheritance**
- topic/section learningBase/lcInstruction

**Example**

```xml
<learningContent id="overview">
  <title>Learning Content topic</title>
  <learningContentbody>
    <lcInstruction>
      <title>Instruction</title>
      <p>Describe the instruction.</p>
    </lcInstruction>
  </learningContentbody>
</learningContent>
```

**Attributes**

The following attributes are available on this element: `Universal attribute group` (see page 650), `outputclass` (see page 0), and `spectitle` (see page 0).

4.11.7.2.25 `<lcInteraction>`

The `<lcInteraction>` is a wrapper element for all the interactions of the assessment. The interactions themselves are based on the `<lcInteractionBase>` type. A starter set of interaction types is defined in the learning domain.

**Inheritance**
- topic/section learningBase/lcInteraction

**Example**

```xml
<learningAssessment id="testAssess">
  <title>Certification Test</title>
  <shortdesc>Pass this test, and you are a certified genius.</shortdesc>
  <learningAssessmentbody>
    <lcIntro>Here's your test, folks. Good luck!</lcIntro>
    <lcInteraction>
      <lcSingleSelect id="asdf">
        <title>Multiple Choice - IEEE standards trivia</title>
        <lcQuestion>Which one of the listed standards committees is responsible for developing the token ring specification?</lcQuestion>
        <lcAnswerOptionGroup>
          <lcAnswerOption>
            <lcAnswerContent>IEEE 802.3</lcAnswerContent>
          </lcAnswerOption>
          <lcAnswerOption>
            <lcAnswerContent>IEEE 802.5</lcAnswerContent>
            <lcCorrectResponse/>
          </lcAnswerOption>
          <lcAnswerOption>
            <lcAnswerContent>IEEE 802.6</lcAnswerContent>
          </lcAnswerOption>
        </lcAnswerOptionGroup>
      </lcSingleSelect>
    </lcInteraction>
    <lcSummary>
      <title>Summary</title>
      <p>You are now certified.</p>
    </lcSummary>
  </learningAssessmentbody>
</learningAssessment>
```
Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *spectitle* (see page 0).

4.11.7.2.26 `<lcIntervention>`

The `<lcIntervention>` describes the approach and strategies to building the learning materials, based on the needs analysis.

**Inheritance**

- topic/section learningBase/section learningPlan/lcIntervention

**Example**

```xml
<learningPlan id="learningPlanTest">
  <title>Learning Plan</title>
  <shortdesc>It's always good to provide a plan.</shortdesc>
  <learningPlanbody>
    <lcIntervention>
      <title>Intervention</title>
      <lcInterventionItem>
        <lcLearnStrat>
          Lesson 5 consists of three SCOs, available from the LMS table of contents. The lesson addresses teaching points that support the module objectives. </lcLearnStrat>
        <lcPlanObjective>
          Learn the basics of SCORM. </lcPlanObjective>
        <lcAssessment>
          The module will have a final assessment with questions associated to the SCOs that comprise the module. </lcAssessment>
        <lcDelivery>
          The course may be presented by land, by sea, and by air. </lcDelivery>
      </lcInterventionItem>
    </lcIntervention>
  </learningPlanbody>
</learningPlan>
```

Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *spectitle* (see page 0).

4.11.7.2.27 `<lcInterventionItem>`

The `<lcInterventionItem>` describes how learning content is built, based on a systems approach to analyzing, designing, developing, implementing, and evaluating any instructional experience.

**Inheritance**

- topic/fig learningBase/fig learningPlan/lcInterventionItem

**Example**

```xml
<learningPlan id="learningPlanTest">
  <title>Learning Plan</title>
  <shortdesc>It's always good to provide a plan.</shortdesc>
  <learningPlanbody>
    <lcIntervention>
      <title>Intervention</title>
    </lcIntervention>
  </learningPlanbody>
</learningPlan>
```
Lesson 5 consists of three SCOs, available from the LMS table of contents. The lesson addresses teaching points that support the module objectives. The module will have a final assessment with questions associated to the SCOs that comprise the module.

The course may be presented by land, by sea, and by air.

### Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *Display attribute group* (see page 660), *outputclass* (see page 0), and *spectitle* (see page 0).

#### 4.11.7.2.28 <lcIntro>

The `<lcIntro>` provides a detailed introduction and description of the content to be delivered, in cases where the `<shortdesc>` is not adequate to fully describe the content. It may also include instructor notes.

#### Inheritance

- `topic/section learningBase/lcIntro`

#### Example

```xml
<learningOverview id="overview">
  <title>Learning Overview topic</title>
  <learningOverviewbody>
    <lcIntro><title>Introduction</title>
    <p>For more information, you would include it here.</p>
    <note>You can have notes, tables, all kinds of things like that here, if you desire.</note>
    </lcIntro>
  </learningOverviewbody>
</learningOverview>
```

### Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *spectitle* (see page 0).

#### 4.11.7.2.29 <lcJtaItem>

The `<lcJtaItem>` provides description of job task analysis (JTA) as related to a particular learning objective.

#### Inheritance

- `topic/p learningBase/p learningPlan/lcJtaItem`

#### Example: lcGapAnalysis

```xml
...  
  <title>Gap Analysis</title>
  <lcGapItem>
    <title>Gap item</title>
```
<lcPlanObjective>Learn the basics before you go further.</lcPlanObjective>

Current learning objectives provide guidance to developing small modules with
terminal learning objectives.

Current learners do not have the capability to approach learning
development in a modular strategy that saves time and costs across
the enterprise.

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0 ).

4.11.7.2.30 <lcKnowledge>

The <lcKnowledge> element describes the learner's current knowledge and the relevancy to the broader plan audience or a specific task in the plan.

Inheritance
- topic/p learningBase/p learningPlan/lcKnowledge

Example

```xml
<learningPlan id="learningPlanTest">
  <title>Learning Plan</title>
  <shortdesc>It's always good to provide a plan.</shortdesc>
  <learningPlanbody>
    <lcNeedsAnalysis>
      <title>Needs analysis</title>
      <lcTask>
        <title>Tasks</title>
        <lcTaskItem>Explain the importance of content reuse, and provide examples of reuse strategy options.</lcTaskItem>
        <lcKnowledge>Learners understand acquisition procedures, program management, instructional systems design.</lcKnowledge>
        <lcSkills>The audience is skilled in program management.</lcSkills>
        <lcAttitude>Learners must be willing to be open and flexible.</lcAttitude>
      </lcTask>
    </lcNeedsAnalysis>
  </learningPlanbody>
</learningPlan>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0 ).

4.11.7.2.31 <lcLearnStrat>

The <lcLearnStrat> describes the manner in which the learning content will be instructed. This should be a high level design that applies instructional-design theories and models.

Inheritance
- topic/p learningBase/p learningPlan/lcLearnStrat

Example

```xml
<learningPlan id="learningPlanTest">
  <title>Learning Plan</title>
  <shortdesc>It's always good to provide a plan.</shortdesc>
</learningPlan>
```
Lesson 5 consists of three SCOs, available from the LMS table of contents. The lesson addresses teaching points that support the module objectives.

The module will have a final assessment with questions associated to the SCOs that comprise the module.

The course may be presented by land, by sea, and by air.

---

### Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

#### 4.11.7.2.32 `<lcLMS>`

The `<lcLMS>` provides the LMS name and version number used in the learning event.

### Inheritance

- `topic/fig learningBase/fig learningPlan/lcLMS`

#### Example

```xml
...<lcLMS>
  <title>LMS</title>
  <p>LMS info, if needed, goes here.</p>
</lcLMS>
...```

### Attributes

The following attributes are available on this element: *Universal attribute group* (see page 650), *Display attribute group* (see page 660), *outputclass* (see page 0), and * spectitle* (see page 0).

#### 4.11.7.2.33 `<lcModDate>`

The `<lcModDate>` provides the project modification date.

### Inheritance

- `topic/fig learningBase/fig learningPlan/lcModDate`

#### Example

```xml
<lcModDate>
  <title>Modification Date</title>
  <p>20070315</p>
</lcModDate>
```
**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650), *Display attribute group* (see page 660), *outputclass* (see page 0), and *spectitle* (see page 0).

4.11.7.2.34 `<lcMotivation>`

The `<lcMotivation>` provides the reasons why the learners want and/or need to take the instruction.

*Inheritance*
- topic/p learningBase/p learningPlan/lcMotivation

**Example**

```
...<lcMotivation>Want to learn about and manage lifecycle costs of content development.</lcMotivation>
...```

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.11.7.2.35 `<lcNeeds>`

The `<lcNeeds>` provides the needs behind the outcomes described by the `<lcGoals>`.

*Inheritance*
- topic/p learningBase/p learningPlan/lcNeeds

**Example**

```
...<lcNeeds>Training efforts require a greater capability in tracking employee learning progress.</lcNeeds>
...```

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

4.11.7.2.36 `<lcNeedsAnalysis>`

The `<lcNeedsAnalysis>` describes the training requirement and identifies the need to develop or revise content. These include periodic training gap analyses, changes to operational or maintenance requirements, and changes to equipment or systems.

*Inheritance*
- topic/section learningBase/section learningPlan/lcNeedsAnalysis

**Example**

```
<learningPlan id="learningPlanTest">
<title>Learning Plan</title>
```
It's always good to provide a plan.

Organizations with new requirements can benefit from the use of this course. The goal for taking this course is to develop organizational self-sufficiency in planning for and developing conformant courseware. Training efforts require a greater capability in tracking employee learning progress. The organization will develop shared expertise in each book enabling an integrative lesson development environment. Some organizations may not have the technical ability to develop training. Overcome this constraint with guidance on selecting contractor support.

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0 ), and spectitle (see page 0 ).
4.11.7.2.37 <lcNextSteps>

The <lcNextSteps> suggests next steps to reinforce the knowledge learned.

Inheritance
- topic/section learningBase/lcNextSteps

Example

```xml
<learningSummary id="overview">
  <title>Learning Summary topic</title>
  <learningSummarybody>
    <lcNextSteps>
      <title>Next steps</title>
      <p>To provide information about next steps, you would include it here.</p>
      <note>You can have notes, tables, all kinds of things like that here, if you desire.</note>
    </lcNextSteps>
  </learningSummarybody>
</learningSummary>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and spectitle (see page 0).

4.11.7.2.38 <lcNoLMS>

Use <lcNoLMS> when you plan to deliver the content as a standalone package, without a learning management system (LMS).

Inheritance
- topic/fig learningBase/section learningPlan/lcNoLMS

Example

```xml
...<lcNoLMS>This content does not depend on delivery with a learning management system.</lcNoLMS>
 ...
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Display attribute group (see page 660), outputclass (see page 0), and spectitle (see page 0).

4.11.7.2.39 <lcObjective>

The <lcObjective> describes a single learning objective.

Inheritance
- topic/li learningBase/lcObjective

Example

```xml
<learningOverview id="overview">
  <title>Learning Overview topic</title>
  <learningOverviewbody>
    ...
</learningOverview>
```
<lcObjectives>
<title>Objectives</title>
<lcObjectiveStem>When you complete this lesson, you'll know how to do the following:</lcObjectiveStem>
<lcObjectivesGroup>
<lcObjective>Create a good learning overview topic.</lcObjective>
<lcObjective>Identify clear learning objectives.</lcObjective>
<lcObjective>Add good test items to assess knowledge gained.</lcObjective>
</lcObjectivesGroup>
</lcObjectives>
</learningOverviewbody>
</learningOverview>

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.7.2.40 <lcObjectives>

The <lcObjectives> lists or describes the learning goals.

Inheritance

- topic/section learningBase/lcObjectives

Example

<llearningOverview id="understanding_the_basics" xml:lang="en-us">
<title>Overview: Understanding the basics</title>
<shortdesc>Mail basics start from the inbox, viewing and opening messages you receive, and moving them to appropriate mail folders for easy access and retrieval.</shortdesc>
<learningOverviewbody>
<lcAudience>The intended audience includes new users of the company email system and anyone wanting a refresher on the basic features.</lcAudience>
<lcDuration>
<title>Expected duration</title>
<lcTime value="00:30">It should take you no more than 30 minutes to complete this module.</lcTime>
</lcDuration>
<lcObjectives>
<lcObjectivesStem>When you complete this lesson, you'll know how to perform the following mail basics:</lcObjectivesStem>
<lcObjectivesGroup>
<lcObjective>Viewing the inbox</lcObjective>
<lcObjective>Opening a message</lcObjective>
<lcObjective>Moving messages to a folder</lcObjective>
</lcObjectivesGroup>
</lcObjectives>
</learningOverviewbody>
</learningOverview>

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and spectitle (see page 0).

4.11.7.2.41 <lcObjectivesGroup>

The <lcObjectivesGroup> contains a list of one or more learning objectives.

Inheritance

- topic/ul learningBase/lcObjectivesGroup
Example

```
<learningOverview id="overview">
  <title>Learning Overview topic</title>
  <learningOverviewbody>
    <lcObjectives>
      <title>Objectives</title>
      <lcObjectivesStem>When you complete this lesson, you'll know how to do the following:</lcObjectivesStem>
      <lcObjectivesGroup>
        <lcObjective>Creating a good learning overview topic.</lcObjective>
        <lcObjective>Identifying clear learning objectives.</lcObjective>
        <lcObjective>Adding good test items to assess knowledge gained.</lcObjective>
      </lcObjectivesGroup>
    </lcObjectives>
  </learningOverviewbody>
</learningOverview>
```

### Attributes

The following attributes are available on this element: **Universal attribute group** (see page 650) and **outputclass** (see page 0).

4.11.7.2.42 `<lcObjectivesStem>`

The `<lcObjectivesStem>` provides a leading sentence to introduce a list of learning objectives.

**Inheritance**

- topic/ph learningBase/lcObjectivesStem

Example

```
<learningOverview id="overview">
  <title>Learning Overview topic</title>
  <learningOverviewbody>
    <lcObjectives>
      <title>Objectives</title>
      <lcObjectivesStem>When you complete this lesson, you'll know how to do the following:</lcObjectivesStem>
      <lcObjectivesGroup>
        <lcObjective>Creating a good learning overview topic.</lcObjective>
        <lcObjective>Identifying clear learning objectives.</lcObjective>
        <lcObjective>Adding good test items to assess knowledge gained.</lcObjective>
      </lcObjectivesGroup>
    </lcObjectives>
  </learningOverviewbody>
</learningOverview>
```

### Attributes

The following attributes are available on this element: **Universal attribute group** (see page 650) and **outputclass** (see page 0).

4.11.7.2.43 `<lcOJT>`

The `<lcOJT>` is "the on-the-job training" and describes aspects of the course taking place in the work environment.

**Inheritance**

- topic/fig learningBase/fig learningPlan/lcOJT
Example

```xml
<lcOJT>
  <title>The On-the-Job Training</title>
  <p>Describe the OJT.</p>
</lcOJT>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Display attribute group* (see page 660), *outputclass* (see page 0), and *spectitle* (see page 0).

4.11.7.2.44 `<lcOrganizational>`

The `<lcOrganizational>` describes an organization's learning requirements.

**Inheritance**

- topic/fig learningBase/fig learningPlan/lcOrganizational

**Example**

```xml
...
  <lcOrganizational>
    <title>Organizational</title>
    <lcGeneralDescription>all about the organization.</lcGeneralDescription>
    <lcGoals>the Goals</lcGoals>
    <lcNeeds>the Needs</lcNeeds>
    <lcValues>the Values</lcValues>
    <lcOrgConstraints>the Constraints</lcOrgConstraints>
  </lcOrganizational>
...
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *Display attribute group* (see page 660), *outputclass* (see page 0), and *spectitle* (see page 0).

4.11.7.2.45 `<lcOrgConstraints>`

The `<lcOrgConstraints>` provides organizational aspects that may limit the organization's ability to effectively use the instruction to meet its goals.

**Inheritance**

- topic/p learningBase/p learningPlan/lcOrgConstraints

**Example**

```xml
...
  <lcOrgConstraints>Some organizations may not have the technical ability to develop training. Overcome this constraint with guidance and planning.</lcOrgConstraints>
...
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).
4.11.7.2.46 <lcPlanAudience>

The <lcPlanAudience> describes characteristics of the learners who take the instruction.

Inheritance
- topic/fig learningBase/fig learningPlan/lcPlanAudience

Example

```xml
...<lcPlanAudience>
  <title>Audience</title>
  <lcGeneralDescription>Organizations with new training requirements can benefit from the use of this course.</lcGeneralDescription>
  <lcEdLevel>The target education levels include some college, college graduate, or post-graduate.</lcEdLevel>
  <lcAge>Adults age 30 - 45.</lcAge>
  <lcBackground>The target audience for this module includes acquisition personnel, program managers, project engineers, instructional designers, and business developers.</lcBackground>
  <lcKnowledge>Learners understand acquisition procedures, program management, instructional systems design.</lcKnowledge>
  <lcMotivation>Want to learn about and manage lifecycle costs of content development.</lcMotivation>
  <lcSpecChars>There are no known learning handicaps in the learning audience.</lcSpecChars>
...</lcPlanAudience>
...```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Display attribute group (see page 660), outputclass (see page 0), and spectitle (see page 0).

4.11.7.2.47 <lcPlanDescrip>

The <lcPlanDescrip> provides a plan description.

Inheritance
- topic/fig learningBase/fig learningPlan/lcPlanDescrip

Example

```xml
...<lcPlanDescrip>
  <title>Plan Description</title>
  <p>The goal of the Joint WG module is to provide learners with a broad overview of the Joint WG.</p>
...</lcPlanDescrip>
...```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Display attribute group (see page 660), outputclass (see page 0), and spectitle (see page 0).
The `<lcPlanObjective>` describes the objective to be addressed by a gap analysis or intervention.

### Example

```xml
<learningPlan id="learningPlanTest">
  <title>Learning Plan</title>
  <shortdesc>It's always good to provide a plan.</shortdesc>
  <learningPlanbody>
    <lcIntervention>
      <title>Intervention</title>
      <lcInterventionItem>
        <lcLearnStrat>
          Lesson 5 consists of three SCOs, available from the LMS table of contents. The lesson addresses teaching points that support the module objectives. </lcLearnStrat>
        <lcPlanObjective>Learn the basics of SCORM.</lcPlanObjective>
        <lcAssessment>The module will have a final assessment with questions associated to the SCOs that comprise the module.</lcAssessment>
        <lcDelivery>The course may be presented by land, by sea, and by air.</lcDelivery>
      </lcInterventionItem>
    </lcIntervention>
  </learningPlanbody>
</learningPlan>
```

### Attributes

The following attributes are available on this element: [Universal attribute group](#) and [outputclass](#).

The `<lcPlanPrereqs>` element provides the knowledge, skills, abilities, courses and other activities learners must have satisfied to take the instruction.

### Example

```xml
...<lcPlanPrereqs>
  <title>Prerequisites</title>
  <p>This course assumes you have mastery of JWG 101 (Fundamentals of Joint Workgroups) or the equivalent.</p>
</lcPlanPrereqs>
...```

### Attributes

The following attributes are available on this element: [Universal attribute group](#), [Display attribute group](#), [outputclass](#), and [spectitle](#).
4.11.7.2.50 <lcPlanResources>

The <lcPlanResources> describes resource needs.

Inheritance
- topic/p learningBase/p learningPlan/lcPlanResources

Example

```xml
<lcWorkEnv>
  <title>Work Environment</title>
  <lcWorkEnvDescription>All learners work in a typical office environment.</lcWorkEnvDescription>
  <lcPlanResources>You need lots of pencils.</lcPlanResources>
  <lcProcesses>Follow these processes:
    <ul>
      <li>Fill our the pencil request form.</li>
      <li>Sharpen the pencils as soon as they arrive, to keep them fresh.</li>
    </ul>
  </lcProcesses>
</lcWorkEnv>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.7.2.51 <lcPlanSubject>

The <lcPlanSubject> provides a complete description of the subject of the planned learning.

Inheritance
- topic/fig learningBase/fig learningPlan/lcPlanSubject

Example

```xml
...
  <lcPlanSubject>
    <title>Subject</title>
    <p>This course covers the goals of the Joint WG.</p>
  </lcPlanSubject>
...
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Display attribute group (see page 660), outputclass (see page 0), and spectitle (see page 0).

4.11.7.2.52 <lcPlanTitle>

The <lcPlanTitle> provides a title for a specific module, lesson, or section of the course that this plan describes.

Inheritance
- topic/fig learningBase/fig learningPlan/lcPlanTitle
Example

```xml
...  
<lcPlanTitle>  
<title>Plan Title</title>  
<p>Joint goals learning plan.</p>  
</lcPlanTitle>  
...  
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Display attribute group (see page 660), outputclass (see page 0), and spectitle (see page 0).

4.11.7.2.53 <lcPlayers>

The `<lcPlayers>` describes tools and plugins used for time-sequenced display at runtime.

Inheritance
- topic/fig learningBase/fig learningPlan/lcPlayers

Example

```xml
<learningPlan id="learningPlan">  
<title>Learning Plan</title>  
<shortdesc>It's always good to provide a plan.</shortdesc>  
<learningPlanbody>  
<lcTechnical>  
<title>lcTechnical</title>  
<lcPlayers>  
<title>Players</title>  
<p>The learning content requires no additional plugins or players.</p>  
</lcPlayers>  
</lcTechnical>  
</learningPlanbody>  
</learningPlan>  
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Display attribute group (see page 660), outputclass (see page 0), and spectitle (see page 0).

4.11.7.2.54 <lcPrereqs>

The `<lcPrereqs>` describes the knowledge, experience, or other prerequisites needed to complete the content.

Inheritance
- topic/section learningBase/lcPrereqs

Example

```xml
<learningOverview id="overview">  
<title>Learning Overview topic</title>  
<learningOverviewbody>  
<lcPrereq><title>Prereqs</title>  
<p>You have no need to know anything prior to taking this lesson. It informs all.</p>  
</lcPrereq>  
</learningOverviewbody>  
```

DITA Version 1.3 Specification
TBD
Copyright © OASIS Open 2005, 2014. All Rights Reserved.
Attributes
The following attributes are available on this element: Universal attribute group (see page 650), outputclass (see page 0), and spectitle (see page 0).

4.11.7.2.55 <lcProcesses>

The <lcProcesses> describes processes learners routinely follow.

Inheritance
- topic/p learningBase/p learningPlan/lcProcesses

Example

```xml
<lcWorkEnv>
  <title>Work Environment</title>
  <lcWorkEnvDescription>All learners work in a typical office environment.</lcWorkEnvDescription>
  <lcPlanResources>You need lots of pencils.</lcPlanResources>
  <lcProcesses>Follow these processes:
    <ul>
      <li>Fill our the pencil request form.</li>
      <li>Sharpen the pencils as soon as they arrive, to keep them fresh.</li>
    </ul>
  </lcProcesses>
</lcWorkEnv>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.7.2.56 <lcProject>

The <lcProject> provides learning content project plan description information.

Inheritance
- topic/section learningBase/section learningPlan/lcProject

Example

```xml
<learningPlan id="learningPlanTest">
  <title>Learning Plan</title>
  <shortdesc>It's always good to provide a plan.</shortdesc>
  <learningPlanbody>
    <lcProject>
      <title>lcProject</title>
      <lcClient>
        <title>Client</title>
        <p>Joint work group</p>
      </lcClient>
      <lcPlanTitle>
        <title>Plan Title</title>
        <p>Joint goals learning plan.</p>
      </lcPlanTitle>
      <lcCIN>
        <title>Joint WG</title>
      </lcCIN>
    </lcProject>
  </learningPlanbody>
</learningPlan>
```
<lcModDate>
  <title>Modification Date</title>
  <p>20070315</p>
</lcModDate>

<lcDelivDate>
  <title>Delivery Date</title>
  <p>20070630</p>
</lcDelivDate>

<lcPlanSubject>
  <title>Subject</title>
  <p>This course covers the goals of the Joint WG.</p>
</lcPlanSubject>

<lcPlanDescrip>
  <title>Plan Description</title>
  <p>The goal of the Joint WG module is to provide learners with a broad overview of the Joint WG.</p>
</lcPlanDescrip>

<lcPlanPrereqs>
  <title>Prerequisites</title>
  <p>This course assumes you have mastery of JWG 101 (Fundamentals of Joint Workgroups) or the equivalent.</p>
</lcPlanPrereqs>

<lcResolution>
  <p>Minimum resolution of 800 wide, 600 high.</p>
</lcResolution>

<lcResources>
  <p>Minimum resolution of 800 wide, 600 high.</p>
</lcResources>

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *spectitle* (see page 0).

4.11.7.2.57 <lcResolution>
The `<lcResolution>` describes the required computer screen resolution for the online instruction.

**Inheritance**
- topic/fig learningBase/fig learningPlan/lcResolution

**Example**

```xml
...<lcResolution><p>Minimum resolution of 800 wide, 600 high.</p></lcResolution>...
```

**Attributes**
The following attributes are available on this element: *Universal attribute group* (see page 650), *Display attribute group* (see page 660), *outputclass* (see page 0), and *spectitle* (see page 0).

4.11.7.2.58 <lcResources>
The `<lcResources>` provides a list of related resources and information about them, such as related articles or samples on the web.

**Inheritance**
- topic/section learningBase/lcResources
Example

<!-- Example: Learning Overview -->

```html
<learningOverview id="overview">
  <title>Learning Overview topic</title>
  <learningOverviewBody>
    <lcResources>
      <title>Resources</title>
      <p>Provide information about useful resources here.</p>
    </lcResources>
  </learningOverviewBody>
</learningOverview>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *spectitle* (see page 0).

### 4.11.7.2.59 lcReview

The `<lcReview>` provides a review of the main points.

**Inheritance**

- topic/section learningBase/lcReview

Example

```html
<learningSummary id="summary">
  <title>Learning Summary topic</title>
  <learningSummaryBody>
    <lcReview>
      <title>Review</title>
      <p>If you want to offer a review, include it here.</p>
      <note>You can have notes, tables, all kinds of things like that here, if you desire.</note>
    </lcReview>
  </learningSummaryBody>
</learningSummary>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *spectitle* (see page 0).

### 4.11.7.2.60 lcSecurity

The `<lcSecurity>` describes the security requirements in the delivered instruction.

**Inheritance**

- topic/fig learningBase/fig learningPlan/lcSecurity

Example

```html
...<lcSecurity>
  <p>Never leave equipment unattended during this course.</p>
</lcSecurity>...
```
Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Display attribute group (see page 660), outputclass (see page 0), and spectitle (see page 0).

4.11.7.2.61 <lcSkills>
The <lcSkills> describes the learners’ current skill set and the relevancy to the broader plan audience or a specific task in the plan.

Inheritance
- topic/p learningBase/p learningPlan/lcSkills

Example

```xml
... <lcSkills>...all about Skills needed...</lcSkills>
...
```

4.11.7.2.62 <lcSpecChars>
The <lcSpecChars> provides learner characteristics specific to the population that will influence the design, including learning disabilities, physical handicaps, and so forth.

Inheritance
- topic/p learningBase/p learningPlan/lcSpecChars

Example

```xml
... <lcSpecChars>There are no known learning handicaps in the learning audience.</lcSpecChars>
...
```
**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650) and *outputclass* (see page 0).

### 4.11.7.2.63 `<lcSummary>`

The `<lcSummary>` provides a textual summary that describes the main learning goals and lessons learned.

**Inheritance**

- topic/section learningBase/lcSummary

**Example**

```xml
<learningSummary id="overview">
  <title>Learning Summary topic</title>
  <learningSummarybody>
    <lcSummary><title>Summary</title>
      <p>If you need a summary section, you would include it here.</p>
      <note>You can have notes, tables, all kinds of things like that here, if you desire.</note>
    </lcSummary>
  </learningSummarybody>
</learningSummary>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *spectitle* (see page 0).

### 4.11.7.2.64 `<lcTask>`

The `<lcTask>` captures a work item to be performed, as part of the learning plan.

**Inheritance**

- topic/fig learningBase/fig learningPlan/lcTask

**Example**

```xml
<learningPlan id="learningPlanTest">
  <title>Learning Plan</title>
  <shortdesc>It's always good to provide a plan.</shortdesc>
  <learningPlanbody>
    <lcNeedsAnalysis>
      <title>Needs analysis</title>
      <lcTask>
        <title>Tasks</title>
        <lcTaskItem>Explain the importance of content reuse, and provide examples of reuse strategy options.</lcTaskItem>
        <lcKnowledge>Learners understand acquisition procedures, program management, instructional systems design.</lcKnowledge>
        <lcSkills>The audience is skilled in program management.</lcSkills>
        <lcAttitude>Learners must be willing to be open and flexible.</lcAttitude>
      </lcTask>
    </lcNeedsAnalysis>
  </learningPlanbody>
</learningPlan>
```
Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Display attribute group (see page 660), outputclass (see page 0), and spectitle (see page 0).

4.11.7.2.65 <lcTaskItem>
The <lcTaskItem> describes a discreet task to be taught.

Inheritance
- topic/p learningBase/p learningPlan/lcTaskItem

Example

```xml
<learningPlan id="learningPlanTest">
  <title>Learning Plan</title>
  <shortdesc>It's always good to provide a plan.</shortdesc>
  <learningPlanbody>
    <lcNeedsAnalysis>
      <title>Needs analysis</title>
      <lcTask>
        <title>Tasks</title>
        <lcTaskItem>Explain the importance of content reuse, and provide examples of reuse strategy options.</lcTaskItem>
        <lcKnowledge>Learners understand acquisition procedures, program management, instructional systems design.</lcKnowledge>
        <lcSkills>The audience is skilled in program management.</lcSkills>
        <lcAttitude>Learners must be willing to be open and flexible.</lcAttitude>
      </lcTask>
    </lcNeedsAnalysis>
  </learningPlanbody>
</learningPlan>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.7.2.66 <lcTechnical>
The <lcTechnical> describes the technical requirements to the learning content and how those requirements are supported by the instructional design.

Inheritance
- topic/section learningBase/section learningPlan/lcTechnical

Example: lcTechnical

```xml
<learningPlan id="learningPlan">
  <title>Learning Plan</title>
  <shortdesc>It's always good to provide a plan.</shortdesc>
  <learningPlanbody>
    <lcTechnical>
      <title>lcTechnical</title>
    </lcTechnical>
  </learningPlanbody>
</learningPlan>
```
**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and *spectitle* (see page 0).

### 4.11.7.2.67 `<lcTime>`

The `<lcTime>` specifies the time expected to complete an activity. You can specify the time both as text to display to the user and as an attribute value that follows a specific time format.

**Inheritance**

- `topic/data learningBase/lcTime`

**Example**

```xml
<learningOverview id="overview">
  <title>Learning Overview topic</title>
  <learningOverviewbody>
    <lcDuration>
      <lcTime name="lcTime" value="PT15M">15 minutes</lcTime>
    </lcDuration>
  </learningOverviewbody>
</learningOverview>
```

**Attributes**

The following attributes are available on this element: *Universal attribute group* (see page 650), *outputclass* (see page 0), and the attributes defined below.

- **@name**
  Name of the metadata being specified. The default value for this attribute is "lcTime". This attribute is defined with the XML Data Type CDATA.

- **@value** *(REQUIRED)*
  The time expected to complete the activity. There is no assumed time-specific data format or type for the time value. This attribute is defined with the XML Data Type CDATA.

- **@datatype**
  Provides a specific data type for the time value, if needed. The default value is string "TimeValue". This attribute is defined with the XML Data Type CDATA.

**Comment by robander, 18 February 2014**

Revised the description from 1.2 for clarity. Full text from 1.2 was "The datatype for this data", and the default value was listed as "There is no assumed time-specific data format or type. If you use a specific datatype for the time value, specify it here". It was also listed as #REQUIRED but there is a default value in the DTD/XSD so it is not required.

### 4.11.7.2.68 `<lcValues>`

The `<lcValues>` describes affective components of desired instructional outcomes.

**Inheritance**

- `topic/p learningBase/p learningPlan/lcValues`

**Example**

```xml
...<lcValues>The organization will develop shared expertise in each book enabling an integrative lesson
```
Attributes
The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.11.7.2.69 <lcViewers>

The <lcViewers> describes viewers used for time-sequenced display at runtime.

Inheritance
- topic/fig learningBase/fig learningPlan/lcViewers

Example

```xml
<learningPlan id="learningPlan">
  <title>Learning Plan</title>
  <shortdesc>It's always good to provide a plan.</shortdesc>
  <learningPlanbody>
    <lcTechnical>
      <title>lcTechnical</title>
      <lcViewers>
        <title>Viewers</title>
        <p>The module uses standard viewers, model 1.0 or 1.1.</p>
      </lcViewers>
    </lcTechnical>
  </learningPlanbody>
</learningPlan>
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Display attribute group (see page 660), outputclass (see page 0), and spectitle (see page 0).

4.11.7.2.70 <lcW3C>

The <lcW3C> provides requirements for use of world wide web consortium standards.

Inheritance
- topic/fig learningBase/fig learningPlan/lcW3C

Example

```
...<lcW3C><p>Whenever possible, stand up and do a W3C.</p></lcW3C>
...
```

Attributes
The following attributes are available on this element: Universal attribute group (see page 650), Display attribute group (see page 660), outputclass (see page 0), and spectitle (see page 0).
4.11.7.2.71 <lcWorkEnv>

The <lcWorkEnv> describes the working conditions and contexts in which the training will be applied.

Inheritance
- topic/fig learningBase/fig learningPlan/lcWorkEnv

Example

```xml
<lcWorkEnv>
    <title>Work Environment</title>
    <lcWorkEnvDescription>All learners work in a typical office environment.</lcWorkEnvDescription>
    <lcPlanResources>You need lots of pencils.</lcPlanResources>
    <ul>
        <li>Fill our the pencil request form.</li>
        <li>Sharpen the pencils as soon as they arrive, to keep them fresh.</li>
    </ul>
</lcWorkEnv>
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650), Display attribute group (see page 660), outputclass (see page 0), and spectitle (see page 0).

4.11.7.2.72 <lcWorkEnvDescription>

The <lcWorkEnvDescription> provides the general working environment in which the training will be applied.

Inheritance
- topic/p learningBase/p learningPlan/lcWorkEnvDescription

Example

```
...
    <lcWorkEnvDescription>All learners work in a typical office environment.</lcWorkEnvDescription>
...
```

Attributes

The following attributes are available on this element: Universal attribute group (see page 650) and outputclass (see page 0).

4.12 Attributes

This section collects commonly used attributes, with common definitions. If an element uses a different definition, or narrows the scope of, an otherwise common attribute, it will be called out in the topic that defines the element.

4.12.1 Universal attribute group

The universal attribute group defines a set of common attributes available on most DITA elements. The universal attribute group includes all attributes from the metadata, ID, localization, and debug attribute groups, plus the @class attribute.

Comment by robander, 14 April 2014

In Phase1 review, Jarno suggested making the following paragraph a non-normative note. This does seem a very good candidate for that, but holding off until we establish a consistent style for such notes.
Earlier versions of the DITA specification grouped some attributes under names taken directly from parameter entities in the standard DTD implementation of DITA. These groups used names such as “univ-atts”, which had no meaning outside the context of a DTD module. In DITA 1.3, these groups are restructured and renamed in the specification document for greater clarity, and no longer have a connection to the DTD. For example, this grouping of "Universal attributes" was previously referred to as “univ-atts”; the updated group includes several universal attributes that are not part of the univ-atts group in the DTD.

@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. The @class attribute supports specialization. Its predefined values allow DITA tools to work correctly with ranges of related content. In a generalized DITA document the @class attribute value in the generalized instance may differ from the default value for the @class attribute for the element as given in the DTD or schema. See Element type specialization hierarchy declaration (the @class attribute) 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.

4.12.1.1 ID attribute group

The ID attribute group includes attributes that enable the naming and referencing of elements in topics and maps.

@id

An anchor point. This ID is the target for references by @href and @conref attributes and for external applications that refer to DITA content. This attribute is defined with the XML data type NMTOKEN, except where noted for specific elements within the language reference. See ID attribute (see page 63) in the Architectural Specification for more details.

@conref

This attribute is used to reference an ID on content that can be reused. See The conref attribute (see page 666) for examples and details about the syntax. This attribute is defined with the XML Data Type CDATA.

@conrefend

The @conrefend attribute is used when reusing a range of elements through @conref. The syntax is the same as for the @conref attribute; see The conrefend attribute (see page 670) for examples. This attribute is defined with the XML Data Type CDATA.

@conaction

This attribute enables users to push content into a new location. Allowable values are mark, pushafter, pushbefore, pushreplace, and -dita-use-conref-target. See The conaction attribute (see page 667) for examples and details about the syntax.

@conkeyref

Allows the conref feature to operate using a key instead of a URI. This attribute is defined with the XML Data Type CDATA. See The conkeyref attribute (see page 674) for more details about the syntax and behaviors.

4.12.1.2 Metadata attribute group

The metadata attribute group includes common metadata attributes, several of which support conditional processing (filtering and flagging) or the creation of new attribute domain specializations.

@props

Root attribute from which new metadata attributes can be specialized. This is a property attribute which supports conditional processing for filtering or flagging. If no value is specified, but the attribute is specified on an ancestor within a map or within the related-links section, the value will cascade from the closest ancestor. The attribute takes a space-delimited set of values. However, when acting as a container for generalized attributes, the content model will be more complex; see Attribute generalization (see page 152) for more details. This attribute is defined with the XML Data Type CDATA.

@base
A generic attribute that has no specific purpose. It is intended to act as a base for specialized attributes that have a simple value syntax like the conditional processing attributes (one or more alphanumeric values separated by whitespace), but is not itself a filtering or flagging attribute. The attribute takes a space-delimited set of values. However, when acting as a container for generalized attributes, the content model will be more complex; see Attribute generalization (see page 152) for more details. This attribute is defined with the XML Data Type CDATA.

@platform
Indicates operating system and hardware. This is a property attribute which supports conditional processing for filtering or flagging. If no value is specified, but the attribute is specified on an ancestor within a map or within the related-links section, the value will cascade from the closest ancestor. The attribute takes a space-delimited set of values, with optional groups of space delimited properties. This attribute is defined with the XML Data Type CDATA.

@product
Contains the name of the product to which the element applies. This is a property attribute which supports conditional processing for filtering or flagging. If no value is specified, but the attribute is specified on an ancestor within a map or within the related-links section, the value will cascade from the closest ancestor. The attribute takes a space-delimited set of values, with optional groups of space delimited properties. This attribute is defined with the XML Data Type CDATA.

@audience
Indicates the intended audience for the element.

Comment by robander
Pedantic question during initial 1.3 editing: audience for the element? Or for the content of the element?

This is a property attribute which supports conditional processing for filtering or flagging. If no value is specified, but the attribute is specified on an ancestor within a map or within the related-links section, the value will cascade from the closest ancestor. The attribute takes a space-delimited set of values, with optional groups of space delimited properties. This attribute is defined with the XML Data Type CDATA.

@deliveryTarget
The intended delivery target of the content, for example "html", "pdf", or "epub". This attribute is a replacement for the now deprecated @print attribute.

The @deliveryTarget attribute is defined in the deliveryTargetAttDomain, which is integrated into all OASIS-provided document-type shells. If this domain is not integrated into a given document-type shell, the @deliveryTarget attribute will not be available.

@otherprops
This attribute can be used for any other properties that might be needed to describe an audience, or to provide selection criteria for the element. Alternatively, the @props attribute may be specialized to provide a new metadata attribute instead of using the general @otherprops attribute. This is a property attribute which supports conditional processing for filtering or flagging. If no value is specified, but the attribute is specified on an ancestor within a map or within the related-links section, the value will cascade from the closest ancestor. The attribute takes a space-delimited set of values, with optional groups of space delimited properties. This attribute is defined with the XML Data Type CDATA.

@importance
A range of values that describe an importance or priority attributed to an element. For example, in steps of a task, the attribute indicates whether a step is optional or required. This attribute is not used for DITAVAL-based filtering or flagging; applications may (but need not) use the importance value to highlight elements. Allowable values are obsolete, deprecated, optional, default, low, normal, high, recommended, required, urgent, and -dita-use-conref-target.

Comment by robander
Should move the allowable values into a <di>?

@rev
Indicates 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. It is not sufficient to be used for version control. If no value is specified, but the attribute is specified on an ancestor within a map or within the related-links section, the value will cascade from the closest ancestor. This attribute is defined with the XML Data Type CDATA.

@status

The modification status of the current element. Allowable values are new, changed, deleted, unchanged, and -dita-use-conref-target.

Comment by robander
Should move the allowable values into a <di>?

Related Links

3.2.3.2.1 Conditional processing attributes (see page 44)
   The metadata attributes specify properties of the content that can be used to determine how the content should be processed. Specialized metadata attributes can be defined to enable specific business-processing needs, such as semantic processing and data mining.

3.3.7 Conditional processing (profiling) (see page 88)
   Conditional processing, also known as profiling, is the filtering or flagging of information based on processing-time criteria.

3.4.3 Specialization (see page 141)
   The specialization feature of DITA allows for the creation of new element types and attributes that are explicitly and formally derived from existing types. The resulting specialization allows for the blind interchange of all conforming DITA content and a minimum level of common processing for all DITA content. It also allows specialization-aware processors to add specialization-specific processing to existing base processing.

3.4.3.11.1.5 Attribute domain module coding requirements (see page 164)
   An attribute domain vocabulary module declares a new attribute specialized from either the @props or @base attribute. An attribute domain module defines exactly one new attribute type.

4.12.1.3 Localization attribute group

The localization attribute group defines a set of common attributes related to translation and localization. These attributes are available on most DITA elements.

@translate
Indicates whether the content of the element should be translated or not. Allowable values are yes, no, and -dita-use-conref-target. See Element-by-element recommendations for translators (see page 693) for suggested processing defaults for each element.

@xml:lang
Specifies the language of the element content. The @xml:lang attribute and its values are described in the XML Recommendation at http://www.w3.org/TR/REC-xml/#sec-lang-tag. Allowable values are language tokens or the null string.

@dir
Specifies the directionality of text: left-to-right (ltr, the processing default) or right-to-left (rtl). The value lro indicates an override of normal bidi text presentation, forcing the element into left-to-right mode; rlo overrides normal rules to force right-to-left presentation. Allowable values are ltr, rtl, lro, rlo, and -dita-use-conref-target. See The dir attribute (see page 108) for more information.

Comment by robander
keeping this example, but wanted to call it out - examples removed as inappropriate from most other attribute group topics - remove this one?
The `@translate`, `@xml:lang`, and `@dir` attributes identify language-specific words or phrases for specific processing (or non-processing, in the case of `translate="no"`).

```xml
<p>The cordial response to the question is
<q translate="no" xml:lang="de-de" dir="ltr">nein.</q></p>
```

Related Links

3.3.10 Translation and localization (see page 107)

DITA has features that facilitate preparing content for translation and working with multilingual content, including the `<xml:lang>` attribute, the `@dir` attribute, and the `@translate` attribute. In addition, the `<index-sort-as>` element provides support for index sorting in languages in which the index sort order must be modified by the author or translator.

4.12.1.4 Debug attribute group

The debug attributes group includes a set of debugging attributes that are normally hidden from authoring view. These attributes are defined for every DITA element.

These attributes are intended to store debugging information during intermediate processing. One possible implementation is to use `@xtrf` (xml-trace-filename) to store the original source file name through intermediate processing steps, and use `@xtrc` (xml-trace-counter) to store an element counter for repositioning authoring tools at the originating element location. If values are assigned in this way in the first stage of a processing stream, values in these attributes can then be used for error recovery.

**Comment by robander**

Uncertain if following note is needed? If so may want to reword? Only adding because I've grouped former "global-atts" into "univ-atts" for the spec, though they are not grouped in the DTD.

**Note:** While this group of attributes is defined with the "Universal attribute group" within the specification, OASIS implementations of the DITA vocabulary do not currently group it with the common "univ-atts" group.

These attributes are normally hidden from authors and exposed only to processing tools or editor macros. Values used in these attributes may be implementation-dependent.

`@xtrf`

xml-trace-filename, the original filename. This attribute is defined with the XML Data Type CDATA.

`@xtrc`

xml-trace-counter, an element counter for repositioning editors at a known edit location. This attribute is defined with the XML Data Type CDATA.

4.12.2 Architectural attribute group

The architectural attributes group includes a set of attributes defined for document level elements such as `<topic>` and `<map>`. These attributes are intended to provide information about the DITA namespace, what level of DITA is in use, and what vocabulary modules are in use.

`@DITAArchVersion`

Designates the version of the architecture that is in use. The default value will increase with each release of DITA. This attribute is in the namespace "http://dita.oasis-open.org/architecture/2005/". This attribute is defined with the XML data type CDATA, but uses a default value of the current version of DITA. The current default is "1.3".

`@xmlns:ditaarch`

Declares the default DITA namespace. Although this is technically a namespace rather than an attribute, it is included here because it is specified as an attribute in the DTD grammar files distributed by OASIS. The value is fixed to "http://dita.oasis-open.org/architecture/2005/".

`@domains`
Indicates the specialized domains that are included in the DTD or Schema. This attribute is defined with the XML data type CDATA, and each new document type SHOULD specify a default. The value will differ depending on what domains are included in the current DTD or Schema; a sample value is "(topic ui-d) (topic hi-d) (topic pr-d) (topic sw-d) (topic ut-d) (topic indexing-d)".

### 4.12.3 Attributes common to many map elements

**Comment by robander**
Suggestions to rename this group? Original title "Common map attribute group" implies these are for maps, but they are for lots of elements in maps. Trying new title "Attributes common to many map elements".

This attribute group collects several attributes that are used on a variety of map elements. For a few elements, the group is modified slightly to remove an attribute such as @toc or @format; in those cases the element definition will clarify that the element does not use this full set. That is generally done in order to specify a default for one attribute, such as defaulting @format to "ditamap" on the <mapref> element.

**Comment by robander**
This was converted from "topicref-atts". I’ve removed type/scope/format from the group, as they are now the complete Link Relationship Group. Need to ensure any topic that referenced topicref-atts also references link-relationship-atts

---

@cascade

Controls how metadata attributes cascade within a map. There are two defined values that should be supported: "merge" and "nomerge". If no value is set, and no value cascades from an ancestor element, processors SHOULD assume a default of "merge". See Cascading of attributes and metadata in a DITA map (see page 48) for more information about how this attribute interacts with metadata attributes.

@collection-type

Collection types describe how links relate to each other. The processing default is "unordered", although no default is specified in the DTD or Schema. Allowable values are:

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

-dita-use-conref-target

  See Using the -dita-use-conref-target value (see page 666) for more information.

Usage of the @collection-type attribute on <reltable> and <relcolspec> is currently undefined and reserved for future use.

@processing-role

Describes the processing role of the referenced topic. The processing default is "normal". If no value is specified, but the attribute is specified on an ancestor within a map or within the related-links section, the value will cascade from the closest ancestor. Allowable values are:

- **normal**
  - Normal topic that is a readable part of the information.

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

-dita-use-conref-target

See Using the -dita-use-conref-target value (see page 666) for more information.

@locktitle

If the @locktitle attribute is set to "yes", the content of the <navtitle> element or @navtitle attribute is used for a navigation title, if it is present. If the @locktitle attribute is not present or set to "no", the content of the <navtitle> element or @navtitle attribute is ignored, and the title of the referenced topic is used as a navigation title.

Note: The @navtitle attribute is deprecated in favor of the <navtitle> element. When both a <navtitle> element and a @navtitle attribute are specified, the <navtitle> element should be used.

Allowable values for @locktitle are:

yes

The content of the <navtitle> element or @navtitle attribute is used for a navigation title.

no

The content of the <navtitle> element or @navtitle attribute is ignored. This is the processing default.

-dita-use-conref-target

See Using the -dita-use-conref-target value (see page 666) for more information.

@linking

Defines some specific linking characteristics of a topic's current location in the map. If the value is not specified locally, the value may cascade from another element in the map (for cascade rules, see Cascading of attributes and metadata in a DITA map (see page 48)). Allowable values are:

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 (see page 666) for more information.

@toc

Specifies whether a topic appears in the table of contents (TOC). If the value is not specified locally, the value may cascade from another element in the map (for cascade rules, see Cascading of attributes and metadata in a DITA map (see page 48)). Allowable values are:

yes

The topic should appear in a generated TOC.

no

The topic should not appear in a generated TOC.

-dita-use-conref-target

See Using the -dita-use-conref-target value (see page 666) for more information.

@print
Specifies whether the topic should be included in a print-specific rendition, such as PDF. The processing default is "yes". If the value is not specified locally, the value may cascade from another element in the map (for cascade rules, see Cascading of attributes and metadata in a DITA map (see page 48)).

Note: Beginning with DITA 1.3, the @print attribute is deprecated. It is replaced with a conditional processing attribute: @deliveryTarget. See @deliveryTarget for more details.

Allowable values are:

**yes**
- Include the topic in the print-oriented file.

**no**
- Do not include the topic in a print-oriented file.

**prinonly**
- Only include the topic when rendering the DITA content in a print-oriented context; the topic should not be included in other contexts, such as when rendering as HTML.

**-dita-use-conref-target**
- See Using the -dita-use-conref-target value (see page 666) for more information.

**@search**
- Describes whether the target is available for searching. If the value is not specified locally, the value may cascade from another element in the map (for cascade rules, see Cascading of attributes and metadata in a DITA map (see page 48)). Allowable values are:

**yes**
- Include the topic in the print-oriented file.

**no**
- Do not include the topic in a print-oriented file.

**-dita-use-conref-target**
- See Using the -dita-use-conref-target value (see page 666) for more information.

**@chunk**
- When a set of topics is transformed using a map, the @chunk attribute allows multi-topic documents to be broken into smaller files and multiple individual topics to be combined into larger combined documents. This attribute is defined with XML data type CDATA.

For a detailed description of the @chunk attribute and its usage, see Chunking (see page 102).

**@keyscope**
- Specifies that the element marks the boundaries of a key scope. See The keyscope attribute (see page 665) for details on how to use the @keyscope attribute.

Related Links
3.2.3.4 Cascading of attributes and metadata in a DITA map (see page 48)
- Certain map-level attributes and metadata elements cascade throughout a map, which facilitates attribute and metadata management. When attributes or metadata elements cascade, they apply to the elements that are children of the element where the attributes or metadata were specified. Cascading applies to a containment hierarchy, as opposed to a element-type hierarchy.

### 4.12.4 Complex-table attribute group

The Complex-table attribute group includes several attributes that are defined on complex table elements. Most of these attributes are part of the OASIS Exchange model; table elements generally use only a subset of the attributes defined in this group. These attributes are not available for the `<simpletable>` elements.

**@align**
- Describes the alignment of text in a table column. Allowable values are:

**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
Use the character specified on the @char attribute for alignment.

-dita-use-conref-target
See Using the -dita-use-conref-target value (see page 666) for more information.

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

@char
Specifies the character for aligning the table entry data.

Default source for <entry> elements starting in this column. If character alignment is specified, the value is the single alignment character source for any implied @char values for entry immediately in this column. A value of "" (the null string) means there is no aligning character. This attribute is defined with the XML Data Type CDATA.

For example, if align="char" and char="r" are specified, then text in the entry should align with the first occurrence of the letter "r" within the entry.

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

@charoff
Specifies the horizontal offset of alignment character when align="char". This attribute is defined with the XML Data Type NMTOKEN.

Default source for <entry> elements starting in this column. For character alignment on an entry in the column, horizontal character offset is the percent of the current column width to the left of the (left edge of the) alignment character.

This value should be number, greater than 0 and less than or equal to 100.

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

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

@colsep
Column separator. A value of 0 indicates no separators; 1 indicates separators. This attribute is defined with the XML Data Type NMTOKEN.

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

@rowsep
Row separator. A value of 0 indicates no separators; 1 indicates separators. This attribute is defined with the XML Data Type NMTOKEN.

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

@rowheader
Indicates whether the entries in the respective column SHOULD be considered row headers. Allowable values are:

firstcol
Indicates that entries in the first column of the table are functionally row headers (analogous to the way that a <thead> element provides column headers). Applies when @rowheader is used on the <table> element.
headers
Indicates that entries of a column described using the `<colspec>` element are functionally row headers (for cases with more than one column of row headers). Applies when `@rowheader` is used on the `<colspec>` element.

`norowheader`
Indicates that entries in the first column have no special significance with respect to column headers. Applies when `@rowheader` is used on the `<table>` element.

`-dita-use-conref-target`
See *Using the `-dita-use-conref-target` value* (see page 666) for more information.

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

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

`@valign`
Indicates the vertical alignment of text in a table entry (cell). Allowable values are:

- **top**
  Align the text to the top of the table entry (cell).
- **bottom**
  Align the text to the bottom of the table entry (cell).
- **middle**
  Align the text to the middle of the table entry (cell).

`-dita-use-conref-target`
See *Using the `-dita-use-conref-target` value* (see page 666) for more information.

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

### 4.12.5 Data element attributes group

The data element attributes group includes attributes that are defined for the `<data>` element, and are reused on most or all specializations of the `<data>` element.

`@name`
Defines a unique name for the object. This attribute is defined with the XML Data Type CDATA.

`@datatype`
Describes 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. This attribute is defined with the XML Data Type CDATA.

`@value`
Specifies a value associated with the current property or element. This attribute is defined with the XML Data Type CDATA.

### 4.12.6 Date attributes group

The date attributes group includes attributes that take date values, and are defined on metadata elements that work with date information.

`@expiry`
The date when the information should be retired or refreshed, entered as YYYY-MM-DD, where YYYY is the year, MM is the month from 01 to 12, and DD is the day from 01-31. This attribute is defined with the XML Data Type CDATA.
It's CDATA in DTD, but I think we restrict in Schema to match this definition?

@golive
The publication or general availability (GA) date, entered as YYYY-MM-DD, where YYYY is the year, MM is the month from 01 to 12, and DD is the day from 01-31. This attribute is defined with the XML Data Type CDATA.

Comment by robander
It's CDATA in DTD, but I think we restrict in Schema to match this definition?

4.12.7 Display attribute group

The display attribute group includes attributes whose values may be used for affecting the display of many elements.

@expanse
Determines the horizontal placement of the element. Allowable values are:

- **page**
  Places the element on the left page margin for left-to-right presentation, or right page margin for right-to-left presentation.

- **column**
  Aligns the element with the current column margin

- **textline**
  Aligns the element 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.

- **spread**
  Indicates that, if possible, the object should be rendered across a multi-page spread. If the rendition target does not have anything corresponding to spreads then spread has the same meaning as "page".

@dita-use-conref-target
See *Using the -dita-use-conref-target value* (see page 666) for more information.

In DITA tables, in place of the `@expanse` attribute used by other DITA elements, the `@pgwide` attribute is used in order to conform to the OASIS Exchange Table Model. The `@pgwide` attribute has a similar semantic (1=page width; 0=resize to galley or column).

Some DITA processors or output formats may not be able to support all values.

@frame
Specifies which portion of a border should surround the element. Allowable values are:

- **top**
  Draw a line before the element

- **bottom**
  Draw a line after the element

- **topbot**
  Draw a line both before and after the element

- **all**
  Draw a box around the element

- **sides**
  Draw a line at each side of the element

- **none**
  Don't draw any lines around this element

@dita-use-conref-target
See Using the -dita-use-conref-target value (see page 66) for more information.

Some DITA processors or output formats may not be able to support all values.

@scale
Specifies a percentage, selected from an enumerated list, that is used to resize fonts in relation to the normal text size. This attribute is primarily useful for print-oriented display.

The @scale attribute provides an acknowledged style-based property directly on DITA elements. For the <table> and <fig> elements, the intent of the property is to allow authors to adjust font sizes on the content of the containing element, primarily for print accommodation. An <image> in these contexts is to be scaled only by its own direct scale property. If not specifically scaled, such an <image> is unchanged by the scale property of its parent <table> or <fig>.

Allowable values are 50, 60, 70, 80, 90, 100, 110, 120, 140, 160, 180, 200, and -dita-use-conref-target. Some DITA processors or output formats may not be able to support all values.

4.12.8 Link relationship attribute group

The link relationship attribute group includes attributes whose values can be used for representing navigational relationships. These attributes occur only on elements that represent relationships among DITA elements or between DITA elements and non-DITA resources.

Comment by robander
Moved @role and @otherrole out of the list of "relational-atts" -- instead, these should link directly from link, linkpool, linklist, and related-links in to the "theroleattribute.dita" topic. Need to make sure topics are updated accordingly.

@href
Provides a reference to a resource. See The href attribute (see page 663) for detailed information on supported values and processing implications. This attribute is defined with the XML Data Type CDATA.

@format
The @format attribute identifies the format of the resource being referenced. See The format attribute (see page 676) for details on supported values. This attribute is defined with the XML Data Type CDATA.

@scope
The @scope attribute identifies the closeness of the relationship between the current document and the target resource. Allowable values are local, peer, external, and -dita-use-conref-target; see The scope attribute (see page 677) for more information on these values.

@type
Describes the target of a reference. See The type attribute (see page 674) for detailed information on supported values and processing implications. This attribute is defined with the XML Data Type CDATA.

4.12.9 Other common attributes

These common attributes are used across a wide variety of elements, with the common definition included below. These attributes are not defined as a group, and many elements only specify one or two from this list.

@keyref
@keyref provides a redirectable reference based on a key defined within a map. See The keyref attribute (see page 665) for information on using this attribute. This attribute is defined with the XML Data Type CDATA.

@outputclass
Names 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. This attribute is defined with the XML Data Type CDATA.
Indicates close vertical spacing between list items. Expanded spacing is the processing default. The output result of compact spacing depends on the processor or browser. Allowable values are:

- **yes**
  - Indicates compact spacing.
- **no**
  - Indicates expanded spacing.

### -dita-use-conref-target

See *Using the -dita-use-conref-target value* (see page 666) for more information.

@xml:space

This attribute is provided on `<pre>`, `<lines>`, and on elements specialized from those. It ensures that parsers in editors and transforms respect the white space, including line-end characters, that is part of the data in those elements. It is intended to be part of the default properties of these elements, and not for authors to change or delete. When defined, it has a fixed value of "preserve".

@anchorref

Identifies a location within another map file where this map will be anchored at runtime. Resolution of the map is deferred until the final step in the delivery of any rendered content. For example, `anchorref="map1.ditamap/a1"` inside `map1.ditamap` when `map1.ditamap` is rendered for delivery. This attribute is defined with the XML Data Type CDATA.

@mapkeyref

Identifies the map, if any, from which the contained links or metadata are derived. This value may be automatically generated by a process that creates the links or metadata based on map context, as a way to identify which map the material came from. If the `<linklist>`, `<linkpool>`, or metadata is manually created by an author, there is no need to use this attribute. Note that this attribute is not related to the @keyref attribute, and is not used for key based processing. This attribute is defined with the XML Data Type CDATA.

---

**Comment by robander**

This description seemed odd, entirely based on having a temporary processing state - revised slightly but still not ideal.

### 4.12.10 Simpletable attribute group

The simpletable attribute group includes several attributes that are defined on the `<simpletable>` element and `<simpletable>` specializations. These attributes are not defined for the OASIS exchange table (`<table>`).

@keycol

Defines the column that can contains headings for each row. No value indicates no key column. When present, the numerical value causes the specified column to be treated as a vertical header. This attribute is defined with the XML Data Type NMTOKEN.

@relcolwidth

A relative value to specify the width of each column in relationship to the width of the other columns. The values are totaled and make a percent. For example:

- `relcolwidth="1* 2* 3*"`
  - causes widths of 16.7%, 33.3%, and 50%.
- `relcolwidth="90* 150*"`
  - causes width of 37.5% and 62.5%. This attribute is defined with the XML Data Type CDATA.
The @refcols attribute is currently undefined, and is reserved for future use. This attribute is defined with the XML Data Type NMTOKENS.

4.12.11 Specialization attributes group

These attributes are designed to be used by specializers, and are not intended for direct use by authors.

@specentry
The specialized entry attribute allows architects of specialized types to define a fixed or default header title for a specialized <stentry> element. Not intended for direct use by authors. This attribute is defined with the XML Data Type CDATA.

@spectitle
The specialized title attribute allows architects of specialized types to define a fixed or default title for a specialized element. Not intended for direct use by authors. This attribute is defined with the XML Data Type CDATA.

4.12.12 Topicref element attributes group

The topicref element attributes group includes attributes that are defined for the <topicref> element, and are reused on most or all specializations of the <topicref> element.

@copy-to
Use the @copy-to attribute on the <topicref> element to provide a different resource name for a particular instance of a resource referenced by the <topicref> (for example, to separate out the different versions of the topic, rather than combining them on output). If applicable, the @copy-to value may include path information. The links and navigation associated with that instance will point to a copy of the topic with the file name you specified. Applications MAY support @copy-to for references to local non-DITA resources. The @copy-to attribute is not supported for references to resources where the effective value for @scope is "peer" or "external". This attribute is defined with the XML Data Type CDATA.

Use the <linktext> and <shortdesc> in the <topicref>'s <topicmeta> to provide a unique name and short description for the new copy.

@navtitle
Specifies the title of the topic as it will appear in the navigation or tables of contents that are generated from the map. Beginning with DITA 1.2, the preferred way to specify the navigation title in a map is with the <navtitle> element, available inside the <topicmeta> element. This attribute is defined with the XML Data Type CDATA.

@query (DEPRECATED)
This attribute is deprecated. It may be removed in the future. This attribute is defined with the XML Data Type CDATA.

4.12.13 Complex attribute definitions

Several DITA attributes require more explanation than can fit in a single table cell. Those attributes are collected here.

4.12.13.1 The @href attribute

The @href attribute is used to reference another DITA topic or map, a specific element inside a DITA topic or map, an external Web page, or another non-DITA resource.

Comment by robander , April 2 2014
Need to add info here about the "/#./elem-id" syntax having dependency on conref.

The value of a DITA @href attribute must be a valid URI reference [RFC 3986]. It is an error if the value is not a valid URI reference. An implementation MAY generate an error message; it MAY recover from this error
condition by attempting to convert the value to a valid URI reference. Note that the path separator character in a URI is the forward slash ("/"); the backward slash character ("\") is not permitted unescaped within URIs.

When an @href attribute references a DITA resource, an @href value that consists of a URI without a fragment identifier resolves to the document element in the referenced document. For the purposes of rendering, such as when a <topicref> reference to a DITA document is used to render the content as HTML, this means that all topics (and topic specializations) in the target document are included in the reference. For the purpose of linking, the reference resolves to the first (or only) topic (or topic specialization) in the document.

An @href value that consists of a URI with a fragment identifier must have a DITA local identifier as the portion after the hash. A DITA local identifier consists of topicID/elementID for a subelement of a topic, and of elementID for topics, maps, and subelements of a map. If the topic referenced by a DITA local identifier is for the same topic, then topicID can be replaced by a period; see Handling xrefs and conrefs within a conref (see page 86) for more information on how this syntax relates to conref resolution.

Note that certain characters -- including but not limited to the hash sign ("#"), question mark ("?"), back slash ("\"), and space -- are not permitted unescaped within URIs. Such characters must be percent-encoded. Also note that the ampersand ("&") and less than ("<") characters are not permitted in XML attribute values; they must be represented by appropriate character or entity references. Some tools might perform this encoding automatically, while other tools might require that users either avoid the special characters or manually insert the encoding.

Example: Common syntax for the @href attribute

The following table includes some examples of common @href syntax. Note that these examples represent only a few common scenarios and are not all inclusive.

<table>
<thead>
<tr>
<th>Target</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>The first topic in a DITA document</td>
<td>href=&quot;file.dita&quot;</td>
</tr>
<tr>
<td>A specific topic in a DITA document</td>
<td>href=&quot;file.dita#topicid&quot;</td>
</tr>
<tr>
<td>A non-topic element inside a DITA topic</td>
<td>href=&quot;#topicid/elementid&quot;</td>
</tr>
<tr>
<td>A non-topic element inside the same DITA topic as the reference</td>
<td>href=&quot;#./elementid&quot;</td>
</tr>
<tr>
<td>An element in a DITA map</td>
<td>href=&quot;myMap.ditamap#map-branch&quot;</td>
</tr>
<tr>
<td>An image</td>
<td>href=&quot;exampleImage.jpg&quot;</td>
</tr>
<tr>
<td>An external resource</td>
<td>href=&quot;<a href="http://www.example.org">http://www.example.org</a>&quot;</td>
</tr>
</tbody>
</table>

where:

- **topicid** is the value of the @id attribute on the DITA topic.
- **elementid** is the value of the @id attribute on the (non-topic) DITA element.
- **map-branch** is the value of the @id attribute on the DITA map element.

Related Links

3.3.6.1 Handling xrefs and conrefs within a conref (see page 86)
4.12.13.2 The @keys attribute

A @keys attribute consists of one or more space-separated keys. Map authors define keys using a <topicref> or <topicref> specialization that contains the @keys attribute. Each key definition introduces an identifier for a resource referenced from a map. Keys resolve to the resources given as the @href value on the key definition <topicref> element, to content contained within the key definition <topicref> element, or both.

The @keys attribute uses the following syntax:

- The value of the @keys attribute is one or more space-separated key names.
- Key names consist of characters that are legal in a URI. The case of key names is significant.
- The following characters are prohibited in key names: "{", "}"", "[", "]", "/", ",", and white space characters.

A key may not resolve to sub-topic elements, although a keyref attribute may do so by combining a key with a sub-topic element id.

Related Links

3.3.5 Indirect key-based addressing (see page 66)

DITA allows references to be made indirectly by referencing a name for a link target, instead of referencing the location of that target. This name is called a key. The DITA key-reference mechanism provides a layer of abstraction so that resource locations can be defined globally at the DITA map level instead of locally in each topic.

4.12.13.3 The @keyref attribute

The @keyref attribute provides an indirect, late-bound reference to topics, to collections of topics (ditabase), to maps, to referenceable portions of maps, to non-DITA documents, to external URIs, or to XML content contained within a key definition topic reference. When the DITA content is processed, the key references are resolved using key definitions from DITA maps.

For elements that may only refer to topics or non-DITA resources, the value of the @keyref attribute is a key name. For elements that may refer to elements within maps or topics, the value of the @keyref attribute is a key name, a slash (/), and the ID of the target element, where the key name must be bound to either the map or topic that contains the target element.

Related Links

3.3.5 Indirect key-based addressing (see page 66)

DITA allows references to be made indirectly by referencing a name for a link target, instead of referencing the location of that target. This name is called a key. The DITA key-reference mechanism provides a layer of abstraction so that resource locations can be defined globally at the DITA map level instead of locally in each topic.

4.12.13.4 The @keyscope attribute

The @keyscope attribute consists of one or more space-separated key scope names. Map authors define the boundaries for key scopes by specifying the @keyscope attribute on <map> elements, <topicref> elements, or elements that are specializations of <map> or <topicref>. Such elements, their contents, and any locally-scoped content referenced from within the element, are considered to be part of the scope. Keys defined within a scope are only directly referenceable from within the same scope. They can be referenced from the parent scope using the scope's name, followed by a period, followed by the key name.

Within a given root map, two distinct key scopes with the same name have no relationship with each other aside from that implied by their relative locations in the key scope hierarchy. They do not, for example, share key definitions. There is no such thing as a non-contiguous key scope. The only processing impact of a key scope's names is in defining the prefixes used when contributing qualified key names to the parent scope. For example, consider the following map segment:

```xml
<map>
  <topicgroup keyscope="xyz" id="scope1">
    <keydef keys="a" id="def1"/>
  </topicgroup>
</map>
```
This map creates two distinct scopes that happen to use the same name (xyz). This results in the following:

- Each `<topicgroup>` sets a scope of "xyz" and includes a key "a". From outside of those two scopes, references to keyref="xyz.a" (key "a" within the scope "xyz") will always resolve to the first instance of that value, which is in the first `<topicgroup>`.
- Within the first `<topicgroup>`, content uses keyref="a" will resolve to the key in that branch (defined on the element with id="def1").
- Within the second `<topicgroup>`, content uses keyref="a" will resolve to the key in that branch (defined on the element with id="def2").

### 4.12.13.5 The @conref attribute

The @conref attribute is used to reference content that can be reused. It allows reuse of DITA elements, including topic or map level elements.

The value of the @conref attribute must be a URI reference to a DITA element. See [URI-based (direct) addressing](see page 64) for details on specifying URI references to DITA elements. As with other DITA references, a @conref attribute that references a resource without an ID is treated as a reference to the first topic or map in the document.

**Note:** When using the @conref attribute on an element, the content of that element is ignored. For example, if a phrase is marked up like this:

```xml
<ph conref="#topic/ph">Something</ph>
```

the word "Something" will be replaced by the content of the referenced `<ph>` element.

**Related Links**

3.3.6 Content inclusion (conref) (see page 84)

The DITA `@conref`, `@conkeyref`, `@conrefend`, and related attributes provide a mechanism for reuse of content fragments within DITA topics or maps.

### 4.12.13.5.1 Using the -dita-use-conref-target value

The value -dita-use-conref-target is available on enumerated attributes and may also be specified on other attributes. When an element uses @conref to pull in content, for any of its attributes assigned a value of ",-dita-use-conref-target", the resulting value for those attributes should also be pulled in from the referenced element.

Ordinarily, when an element uses @conref, any other attributes specified locally will be preserved when the reference is resolved. This causes problems when attributes are required, because required attributes must be specified regardless of whether the @conref attribute is present. The purpose of the -dita-use-conref-target value is to allow the author to specify a value for a required attribute while still allowing the conref resolution process to use the matching attribute from the referenced element. The value has the same result when the attribute is not required.

This example shows a map where the `<topichead>` element uses conref. It specifies the @navtitle attribute as well as the @toc attribute. In the resolved element, @navtitle is not preserved because it uses -dita-use-conref-target; the @toc attribute overrides the @toc attribute on the referenced element using normal conref resolution rules.

```xml
<map><title>Conref demonstration</title>
  <topichead id="heading"
    navtitle="This is a heading"
</topichead>...
```

---

DITA Version 1.3 Specification

Copyright © OASIS Open 2005, 2014. All Rights Reserved.
The DITA `@conref`, `@conkeyref`, `@conrefend`, and related attributes provide a mechanism for reuse of content fragments within DITA topics or maps.

### 4.12.13.6 The `@conaction` attribute

The `@conaction` attribute allows users to push content from one topic into another. It causes the `@conref` attribute to work in reverse, so that the content is pushed from the current topic into another, rather than pulled from another topic into the current one.

**Note:** In the descriptions below, the word *target* always refers to the element referenced by a `@conref` attribute.

There are three possible functions using the `@conaction` attribute: replacing an element, pushing content before an element, and pushing content after an element. The `@conaction` attribute always declares the desired function while the `@conref` attribute provides the target of the reference using the standard `@conref` syntax.

In each case, an element pushed using `@conref` must be of the same type as, or more specialized than, its target. If the pushed element is more specialized than the target, then it should be generalized when the `@conref` is resolved. This ensures that the content will be valid in the target topic.

- It is valid to push using `@conref` when the two elements involved are of the same type. For example, a `<step>` element may use the conref push feature with another `<step>` as the target of the `@conref`.
- The target element may be more general than the source. For example, it is legal to push a `<step>` element to replace a general list item (`<li>`); the `<step>` element should be generalized back to a list item during the process.
- It is not possible to push a more general element into a specialized context. For example, it is not legal to push a list item (`<li>`) in order to replace a `<step>`, because the list item allows many items that are not valid in the specialized context.
Replacing content in another topic

When the @conaction attribute is set to "pushreplace", the source element will replace the target specified on the @conref attribute. The pushed content remains in the source topic where it was originally authored.

For example, assume that a task in example.dita has the id "example", and contains a <step> with the id "b":

```xml
<task id="example" xml:lang="en">
  <title>Example topic</title>
  <taskbody>
    <steps>
      <step id="a"><cmd>A</cmd></step>
      <step id="b"><cmd>B</cmd></step>
      <step id="c"><cmd>C</cmd></step>
    </steps>
  </taskbody>
</task>
```

In order to replace the step with id="b", another topic must combine a @conaction value of "pushreplace" with a @conref attribute that references this <step>:

```xml
<task id="other" xml:lang="en">
  ...
  <step conaction="pushreplace"
    conref="example.dita#example/b">
    <cmd>Updated B</cmd>
  </step>
  ...
</task>
```

The result will be an updated version of example.dita which contains the pushed <step>:

```xml
<task id="example" xml:lang="en">
  <title>Example topic</title>
  <taskbody>
    <steps>
      <step id="a"><cmd>A</cmd></step>
      <step id="b"><cmd>Updated B</cmd></step>
      <step id="c"><cmd>C</cmd></step>
    </steps>
  </taskbody>
</task>
```

When resolving a conref push action, attributes are resolved using the same precedence as for normal @conref, with one exception. Attributes on the element with the @conref attribute (in this case, the source doing the push) will take priority over those on the referenced element. The exception is that if the source element does not specify an ID, the ID on the referenced element remains; if the source element does specify an ID then that replaces the ID on the referenced element.

Note: It is an error for two source topics to replace the same element. Applications MAY warn users if more than one element attempts to replace a single target.

Pushing content before or after another element

Setting the @conaction attribute to "pushbefore" allows an element to be pushed before the element referenced by the @conref Attribute. Likewise, setting the @conaction attribute to "pushafter" allows an element to be pushed after the element referenced by the @conref Attribute. Multiple sources may push content before or after the same target; the order in which that content is pushed is undefined.

When an element is pushed before or after a target, the resulting document will have at least two of that element. Because this is not always valid, a document attempting to push content before or after a target must take an extra step to ensure that the result will be valid. The extra step makes use of the conaction="mark" value.

When pushing before, the @conref attribute itself looks just as it did when replacing, but the @conaction attribute is set to "mark" because it is marking the target element. This element remains empty; its purpose is to
ensure that it is legal to have more than one of the current element. Immediately before the element which marks the target, you will place the content that you actually want to push. This element will set the @conaction attribute to “pushbefore”.

When pushing after, the procedure is the same, except that the order of the elements is reversed. The element with conaction="pushafter" comes immediately after the element which marks the target.

Attributes on the element which is pushed (the one with conaction="pushbefore") must be retained on the target, apart from the @conaction attribute itself. If this causes the result document to end up with duplicate IDs, an application may (but need not) recover by dropping the duplicate ID, modifying it to ensure uniqueness, or warning the user.

Note: The following restrictions apply when pushing content before or after an element:

• The elements that use conaction="mark" and conaction="pushbefore" must be the same type as each other and must appear in sequence. This restriction prevents a topic from trying to push a <body> element before or after another <body> element, because it is not valid to have two body elements in sequence.

• The container elements of the source and target must match, or the container of the source element may be a specialization of the target's container. This is also to ensure validity of the target; for example, while it is possible to include multiple titles in a <section>, it is not possible to do so in a figure. Comparing the parents prevents a second <section> title from being pushed before a figure title (the resulting figure would not be valid DITA). This restriction only applies to the pushbefore or pushafter actions, not to the pushreplace action.

When content is pushed from one topic to another, it is still rendered in the original context. Processors may delete the empty element that has the conaction="mark" attribute. In order to push content from a topic without actually rendering that topic on its own, the topic should be referenced from the map with the @processing-role attribute set to "resource-only".

Example: pushing an element before the target

The following example pushes a <step> before "b" in the example.dita file shown above.

```dita
<step conaction="pushbefore"><cmd>Do this before B</cmd></step>
<step conaction="mark" conref="example.dita#example/b">
<cmd/>
</step>
```

The result contains the pushed <step> element before "b".

```dita
<task id="example" xml:lang="en">
<title>Example topic</title>
<taskbody>
<steps>
  <step id="a"><cmd>A</cmd></step>
  <step id="b"><cmd>Do this before B</cmd></step>
  <step id="c"><cmd>B</cmd></step>
  <step id="d"><cmd>C</cmd></step>
</steps>
</taskbody>
</task>
```

Example: pushing an element after the target

Pushing an element after a target is exactly the same as pushing before, except that the order of the "mark" element and the pushed element are reversed.

```dita
<step conaction="mark" conref="example.dita#example/b">
<cmd/>
</step>
<step conaction="pushafter"><cmd>Do this AFTER B</cmd></step>
```
In this case the resulting document has the pushed content after `<step> b`:

```xml
<task id="example" xml:lang="en">
  <title>Example topic</title>
  <taskbody>
    <steps>
      <step id="a">A</step>
      <step id="b">B</step>
      <step>Do this AFTER B</step>
      <step id="c">C</step>
    </steps>
  </taskbody>
</task>
```

**Combining @conaction with @conkeyref OR @conrefend**

The @conkeyref attribute may be used as an indirect way to specify a @conref target. If the @conkeyref attribute is specified on an element that also uses the @conaction attribute, the @conkeyref attribute is used to determine the target of the conref push (as it would normally be used to determine the target of @conref).

The conref push function does not provide the ability to push a range of elements, so it is an error to specify the @conrefend attribute together with the @conaction attribute. If the two are specified together an application may (but need not) recover by warning the user and ignoring the @conrefend attribute.

**4.12.13.7 The @conrefend attribute**

The @conrefend attribute is used when referencing a range of elements with the conref mechanism. The @conref or @conkeyref attribute points to the first element in the range, while @conrefend points to the last element in the range. Although the start and end elements must be of the same type as the referencing element (or specialized from that element), the intermediary, contiguous nodes in the middle of the range do not have to be the same.

**Using @conref together with @conrefend**

Several items must be taken into account when using or implementing @conrefend.

- Processors will resolve the range by pulling in the start target and following sibling DOM nodes across to and including the end target.
- The start and end elements of a range must be of the same type as the referencing element, or they must be generalizable to the referencing element. For example, @conref and @conrefend may point from `<li>` to other `<li>` elements, or to specializations of `<li>` such as `<step>`.
- As with @conref, if the @conrefend points to a more specialized version of the referencing element, applications should generalize the target when resolving.
- It is not valid to use @conrefend to point to a more general version of an element (such as using `<step>` to reference an `<li>` element).
- Other nodes (such as elements or text) between the start and end of a range do not have to match the referencing element.
- The start and end elements in a range must share the same parent.
- The parent of the referencing element must be the same as the parent of the target range, OR the parent of the target range may be a specialized version of the reference’s parent. For example, it is possible to pull a range from `<conbody>` into `<body>`, because `<conbody>` is specialized from `<body>`. It is not possible to pull a range from `<body>` into `<conbody>`, because the result may not be valid in `<conbody>`.
- With single conref, an @id attribute from the target will not be preserved on the resolved content. With a range, any @id on both the start and the end will not be preserved. @id attributes on intermediate or child nodes should be preserved, although if this results in duplicate @id values, an application may or may not recover by changing the @id or by warning the user.
- With a single conref, attributes specified locally may be used to override attributes on the referenced content. With a conref range, the same is true, with the following clarifications:
• When an @id attribute is specified on the referencing element, it will only be preserved on the first element of the resolved range.
• When other attributes are specified, they will only apply to referenced elements of the same type. For example, if `<step>` is used to pull in a range of sequential `<step>` elements, locally specified attributes apply to all steps in the range. If `<ol>` is used to pull in a series of (`<ol>`, `<p>`, `<ol>`), locally specified attributes apply only to the `<ol>` elements in that range.

Example: reusing a set of list items

```xml
<topic id="x">
  ...
  <body>
    <ol>
      <li id="apple">A</li>
      <li id="bear">B</li>
      <li id="cat">C</li>
      <li id="dog">D</li>
      <li id="eel">E</li>
    </ol>
  </body>
</topic>
```

**Figure 108: List example: Source topic.dita with ids**

```xml
<topic id="y">
  ...
  <body>
    <ol>
      <li>My own first item</li>
      <li conref="topic.dita#x/bear" conrefend="topic.dita#x/dog"/>
      <li>And a different final item</li>
    </ol>
  </body>
</topic>
```

**Figure 109: List example: Reusing topic with conrefs**

```xml
<topic id="y">
  ...
  <body>
    <ol>
      <li>My own first item</li>
      <li>B</li>
      <li id="cat">C</li>
      <li>D</li>
      <li>And a different final item</li>
    </ol>
  </body>
</topic>
```

**Figure 110: List example: Processed result of reusing topic**

Example: Reusing a set of blocks

```xml
<topic id="x">
  ...
  <body>
    <p id="p1">First para</p>
    <ol id="mylist">
      <li id="apple">A</li>
      <li id="bear">B</li>
      <li id="cat">C</li>
      <li id="dog">D</li>
      <li id="eel">E</li>
    </ol>
    <p id="p2">Second para</p>
  </body>
</topic>
```
Figure 111: Block level example: Source `topic.dita` with ids

```xml
<topic id="y">
  ...
  <body>
    <p conref="topic.dita#x\pl conrefend="topic.dita#x\p2"/>
  </body>
</topic>
```

Figure 112: Block level example: Reusing topic with conrefs

```xml
<topic id="y">
  ...
  <body>
    <p>First para</p>
    <ol id="mylist">
      <li id="apple">A</li>
      <li id="bear">B</li>
      <li id="cat">C</li>
      <li id="dog">D</li>
      <li id="eel">E</li>
    </ol>
    <p>Second para</p>
  </body>
</topic>
```

Figure 113: Block level example: Processed result of reusing topic

Using `@conrefend` together with `@conkeyref`

When the `@conkeyref` attribute is used in place of `@conref`, a key is used to address the target of the reference. The `@conrefend` attribute, which indicates the end of a `@conref` range, may not use a key. Instead the the map or topic element addressed by the key name component of the `@conkeyref` is used in place of whatever map or topic element is addressed by the `@conrefend` attribute.

For example, if the value of the `@conkeyref` attribute is "config/step1" and the value of the `@conrefend` is `defaultconfig.dita#config/laststep`, the conref range will end with the step that has id="laststep" in whatever topic is addressed by the key name "config". If the key name "config" is not defined, and the `@conref` attribute itself is not present for fallback, the `@conrefend` attribute is ignored.

Example: Combining `@conrefend` with `@conkeyref`

In this example the key "xmp" is defined as the first topic in the file `examples.dita`.

```xml
<map>
  <!-- ... -->
  <keydef keys="xmp" href="examples.dita"/>
  <!-- ... -->
</map>

examples.dita:
<topic id="examples">
  <title>These are examples</title>
  <body>
    <ul>
      <li id="first">A first example</li>
      <li>Another trivial example</li>
      <li id="last">Final example</li>
    </ul>
  </body>
</topic>
```
To reuse these list items by using the key, the `@conkeyref` attribute combines the key itself with the sub-topic id (first) to define the start of the range. The `@conrefend` attribute defines a default high-level object along with the sub-topic id (last) that ends the range:

```xml
<li conkeyref="xmp/first"
    conrefend="default.dita#default/last"/>
```

The `@conkeyref` attribute uses a key to reference the first topic in `examples.dita`, so the range begins with the object `examples.dita#examples/first`. The high-level object in the `@conrefend` attribute (`default.dita#default`) is replaced with the object represented by the key (the first topic in `examples.dita`), resulting in a range that ends with the object `examples.dita#examples/last`.

**Figure 114: Defining and referencing a key with `@conkeyref`**

When `@conref`, `@conkeyref`, and `@conrefend` are all specified, the key value takes priority.

```xml
<li conkeyref="thisconfig/start"
    conref="standardconfig.dita#config/start"
    conrefend="standardconfig.dita#config/end"/>
```

- If the key "thisconfig" is defined as `mySpecialConfig.dita#myconfig`, then the range will go from the list item with id="start" to the list item with id="end" in the topic `mySpecialConfig.dita#myconfig`.
- If the key "thisconfig" is defined as `myConfig.dita`, then the range will go from the list item with id="start" to the list item with id="end" within the first topic in `myConfig.dita`.
- If the key "thisconfig" is not defined, then the unchanged `@conref` and `@conrefend` attributes are used as fallback. In that case, the range will go from the list item with id="start" to the list item with id="end" within the topic `standardconfig.dita#config`.

**Figure 115: Combining `@conref`, `@conkeyref`, and `@conrefend`**

**Error conditions**

When encountering an error condition, an implementation may but need not issue an error message.

<table>
<thead>
<tr>
<th>Condition or Issue</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <code>@conref</code> attribute cannot be resolved in the target document (the target element may have been removed or its id has changed).</td>
<td>The <code>@conref</code> is ignored.</td>
</tr>
<tr>
<td>The <code>@conrefend</code> attribute cannot be resolved in the target document (the target element may have been removed or its id has changed).</td>
<td>Range cannot be resolved, optional recovery processes the result as a simple conref.</td>
</tr>
<tr>
<td>Start and end elements are not siblings in the target document.</td>
<td>If the start element exists, optional recovery processes the result as a simple conref.</td>
</tr>
<tr>
<td>End element occurs before the start element in the target document.</td>
<td>If the start element exists, optional recovery processes the result as a simple conref.</td>
</tr>
<tr>
<td>An element has a <code>@conrefend</code> attribute but is missing the <code>@conref</code> attribute.</td>
<td>No result.</td>
</tr>
</tbody>
</table>
4.12.13.8 The @conkeyref attribute

The @conkeyref attribute provides an indirect content reference to topic elements, map elements, or elements within maps or topics. When the DITA content is processed, the key references are resolved using key definitions from DITA maps. For content references from map elements to map elements or topic elements to topic elements, the value of the @conkeyref attribute is a key name, where the key must be bound to a map element (for references from map elements) or a topic element (for references from topic elements). For all other elements, the value of the @conkeyref attribute is a key name, an optional slash ("/"), and the ID of the target element, where the key name must be bound to the map or topic that contains the topic element.

When the key name specified by the @conkeyref attribute is not defined and the element also specifies a @conref attribute, the @conref attribute is used to determine the content reference relationship. If no @conref attribute is specified there is no content reference relationship. Processors SHOULD issue a warning when a @conkeyref reference cannot be resolved and there is no @conref attribute to use as a fallback. Processors MAY issue a warning when a @conkeyref cannot be resolved to an element and a specified @conref is used as a fallback.

The @conrefend attribute, which defines the end of a conref range, may not include a key. Instead the map or topic element addressed by the key name component of the @conkeyref is used in place of whatever map or topic element is addressed by the @conrefend attribute. See Using conrefend together with conkeyref (see page 672) for more information and for examples of this behavior.

4.12.13.9 The @type attribute

The @type attribute is used on linking elements to describe the target of a cross-reference. It also is used on the <note> element to describe the note type, as well as on several other elements for varying purposes. The descriptions for the @type attribute on linking elements and on <note> are included in this section; for other elements, such as <audience>, <copyright>, and <object>, the description can be found with the topic for the specific element.

Using @type on a linking element

The @type attribute describes the target of a cross-reference and might generate cross-reference text based on that description. Only the <xref> element can link to content below the topic level: other types of linking can target whole topics, but not parts of topics. Typically <xref> should also be limited to topic-level targets, unless the output is primarily print-oriented. Web-based referencing works best at the level of whole topics, rather than anchor locations within topics.

If not explicitly specified on an element, the @type attribute value cascades from the closest ancestor element. If there is no explicit value for the @type attribute on any ancestor, a default value of "topic" is used. During output processing for references to DITA topics (format="dita"), it is an error if the actual type of a DITA topic and the explicit, inherited, or default value for the @type attribute are not the same as or a specialization of the @type attribute value. In this case, an implementation MAY give an error message, and MAY recover from this error condition by using the @type attribute value. During output processing for references to non-DITA objects (that is, either scope is "external" or format is neither "dita" nor "ditamap") or other cases where the type of the referenced item cannot be determined from the item itself, the explicit, inherited, or default value for the @type attribute is used without any validation. When a referencing element is first added to or updated in a document, DITA aware editors MAY set the @type attribute value based on the actual type of a referenced DITA topic.

If the @type attribute is specified when referencing DITA content, it should match one of the values in the referenced element's @class attribute. The @type value can be an unqualified local name (for example, "fig") or a qualified name exactly as specified in the @class attribute (for example, "mymodule/mytype"). Processors might ignore qualified names or consider only the local name.
For example, if the value is set to type="topic", the link could be to a generic topic, or any specialization of topic, including concept, task, and reference. Applications MAY issue a warning when the specified or inherited @type attribute value does not match the target (or a specialization ancestor of the target).

Some possible values for use on the <xref> element and its specializations include:

fig
- Indicates a link to a figure.

table
- Indicates a link to a table.

li
- Indicates a link to an ordered list item.

fn
- Indicates a link to a footnote.

section
- Indicates a link to a section.

Other values that may be used on any linking element include:

concept, task, reference, topic
- Cross-reference to a topic type.

(no value)
- The processor should retrieve the actual type from the target if available. If the type cannot be determined, the default should be treated as "topic".

-dita-use-conref-target
- See Using the -dita-use-conref-target value (see page 666) for more information.

Other values can be used to indicate other types of topics or elements as targets. Processing is only required to support the above list or specializations of types in that list. Supporting additional types as targets may require the creation of processing overrides.

Using @type in a <note> element

In a <note> element, this defines the type of note. For example, if the note is a tip, the word Tip may be used to draw the reader’s attention to it. The values danger, warning, and notice have meanings that are based on ANSI Z535 and ISO 3864 regulations.

If @type is set to "other", the value of the @othertype attribute may be used. If you use @othertype, many processors will require additional information on how to process the value. Allowable values for the @type attribute are:

note
- This is just a note.

attention
- Please pay extra attention to this note.

caution
- Care is required when proceeding.

danger
- Important! Be aware of this before doing anything else. When used with the <hazardstatement> element, this indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.

fastpath
- This note will speed you on your way.

important
- This note is important.
notice
Indicates a potential situation which, if not avoided, may result in an undesirable result or state.

remember
Don't forget to do what this note says.

restriction
You can't do what this note says.

tip
This is a fine little tip.

warning
Indicates a potentially hazardous situation. When used with the <hazardstatement> element, this indicates a situation which, if not avoided, could result in death or serious injury.

@trouble
Provides information about how to remedy a trouble situation.

other
This is something other than a normal note.

@dita-use-conref-target
See Using the -dita-use-conref-target value (see page 666) for more information.

4.12.13.10 The @format attribute

The @format attribute identifies the format of the resource that is referenced. If no value is specified, but the attribute is specified on an ancestor within a map or within the related-links section, the value will cascade from the closest ancestor.

The following values for @format have special processing implications:

dita
The destination uses DITA topic markup or markup specialized from a DITA topic. Unless otherwise specified, when @format is set to "dita", the value for the @type attribute will be treated as "topic".

ditamap
The linked-to resource is a DITA map. It represents the referenced hierarchy at the current point in the referencing map. References to other maps can occur at any point in a map, but because relationship tables are only valid as children of a map, referenced relationship tables are treated as children of the referencing map.

Note: If a <topicref> element that references a map contains child <topicref> elements, the processing behavior regarding the child <topicref> elements is undefined.

(no value)
The processing default is used. The processing default for the @format attribute is determined by inspecting the value of the @href attribute. If the @href attribute specifies a file extension, the processing default for the @format attribute is that extension, after conversion to lower-case and with no leading period. The only exception to this is if the extension is "xml", in which case the default format is "dita".

Comment by Bob Thomas , 8 March 2014
This description needs to specify that a ".html" extension MUST imply format value "html".

If there is no extension, but the @href value is an absolute URI whose scheme is "http" or "https", then the processing default is "html". In all other cases where no extension is available, the processing default is "dita".

For other formats, using the file extension without the "." character typically represents the format. For example, the following values are all possible values for @format:

html
The format of the linked-to resource is HTML or XHTML.

pdf
The format of the linked-to resource is PDF.

txt
The format of the linked-to resource is a text file.

4.12.13.11 The @scope attribute

The @scope attribute identifies the closeness of the relationship between the current document and the target resource.

• Set @scope to "local" when the resource is part of the current set of content.
• Set @scope to "peer" when the resource is part of the current set of content but is not accessible at build time, or for maps to be treated as root maps for the purpose of creating map-to-map key references (peer maps). An implementation might choose to open such resources in the same browser window to distinguish them from those with @scope set to "external".
• Set @scope to "external" when the resource is not part of the current information set and should open in a new browser window.
• See Using the -dita-use-conref-target value (see page 666) for more information on -dita-use-conref-target.

If no value is specified, but the attribute is specified on an ancestor within a map or within the related-links section, the value will cascade from the closest ancestor. The processing default is determined by the value of the @href attribute. In most cases, the processing default is "local" whenever the absolute URI in the @href attribute begins with one of the following schemes:

• http
• https
• ftp
• mailto

Processors may consider additional URI schemes as external by default, but must not consider all absolute URIs as external by default. Processors MUST always consider relative URIs as local by default.

Comment by Kristen Eberlein, 7 March 2014
Do we need to use normative RFC-2119 terminology in the above paragraph?

4.12.13.12 The @role and @otherrole attributes

The @role attribute defines the role the target 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 structure could be used to sort and classify links at display time.

Comment by robander
Moved that text out of conref file and inline; was reused in the attribute table for "role", but that usage should no longer occur.

Supported values for @role

Allowable values for the @role attribute are:

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

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

sibling
indicates a link between two children of the same parent 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.

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

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

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

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

**sample**
Deprecated.

**external**
Deprecated--use the scope="external" attribute to indicate external links.

**other**
Indicates any other kind of relationship or role. Enter that role as the value for the `@otherrole` attribute.

**-dita-use-conref-target**
See *Using the -dita-use-conref-target value* (see page 666) for more information.

The `@otherrole` attribute is available to specify an alternate role that is not available in the list above, and should be used in conjunction with `role="other"`. 
5 Conformance

This section outlines the requirements that documents, document types, vocabulary and constraint modules, and processors MUST meet in order to be considered DITA conforming. This section also defines conformance-related terminology and categories.

Conformance to the DITA specification allows documents and document types that are used with different processors or different versions of a processor to produce the same or similar results with little or no reimplementation or modification.

Key words

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, OPTIONAL in the DITA specification are to be interpreted as described in IETF RFC 2119: Key words for use in RFCs to Indicate Requirement Levels.

The use of these key words and other conformance requirements increase the level of interoperability that is available between DITA conforming implementations. Their use is not meant to impose particular methods on implementers where the method is not required for interoperability.

The key words informative and non-normative identify content that is not normative. The DITA specifications include examples and other suggestions that are informative rather than normative. While informative information is often very helpful, it is never a binding part of the DITA specification even when the example or other information is about a feature that is required. Unless it is clearly stated otherwise, examples and the appendices are always informative rather than normative.

Conformance statement

Documents, document types, document type shells, vocabulary and constraint modules, and processors that implement the requirements given in the DITA specification are considered conforming.

Conformance of DITA implementations

A DITA implementation can consist of any combination of processing components that claim DITA awareness, custom vocabulary and constraint modules, and custom document-types shells.

For example, a DITA implementation might be a DITA-based authoring and publishing system within an enterprise; a task-specific product, such a DITA-aware XML editor or component management system;or a set of specialization and constraint modules that are packaged for integration with larger implementations.

Conforming DITA implementations MUST include a conformance statement that gives the version of the DITA specification that is supported. The conformance statement must include one of the following:

- List of the DITA features that are supported by the implementation in accordance with the requirements of the DITA specification
- Statement that the implementation includes all features except for a specific list of features that are not supported in accordance with the requirements of the DITA specification

Implementations that include some (but not all) DITA features are considered conforming as long as all REQUIRED features relevant to the implementation are included and all of the features that are included follow the requirements given in the DITA specification.

An implementation that does not include a particular optional feature MUST be prepared to interoperate with other implementations that do include the feature, though perhaps with reduced functionality. An implementation that does include a particular optional feature MUST be prepared to interoperate with other implementations that do not include the feature.
Organizations and individuals are free to impose additional constraints on their own use of DITA that go beyond the requirements imposed by the DITA specification, including enforcement of the constraints by their local processors, as long as the result continues to meet the requirements given in the DITA specification. For example, a user community can impose rules on how files must be named or organized even if those rules go beyond the requirements given in the DITA specification.

Processors that are not DITA-aware (as defined here) are not considered conforming, but such processors might still be useful when working with DITA.

**Conformance of documents**

A *conforming DITA document* is a document that meets all of the following criteria:

- It is a well-formed XML document.
- Its elements either are DITA elements or non-DITA elements that are contained within `<foreign>` or `<unknown>` elements.
- Its content conforms to all DITA requirements for element content and attribute values.
- If the document has a document type declaration or an associated XSD, the referenced document type or XSD is a conforming DITA document-type shell.

The use of non-DITA-conforming document type declarations or schemas for conforming DITA documents **MUST NOT** affect the ability of processors to process those documents. However, the use of non-conforming document types or schemas might impede interchange or interoperation of those documents with tools that expect or require the use of conforming DITA document types or schemas.

**Conformance of document types and modules**

A *conforming document type* is a document type that consists only of conforming DITA vocabulary and constraint modules.

A *conforming document-type shell* is a document type shell that represents a conforming DITA document type and conforms to the requirements for DITA document-type shells.

A *conforming vocabulary module* is a vocabulary module that conforms to the requirements for its module type.

A *conforming constraint module* is a constraint module that conforms to the requirements for its module type.

**Conformance of processors**

The conformance of processors can only be determined for *DITA aware processors*. We define three types of DITA aware processors:

- A *DITA aware processor* is a processor that can handle documents conforming to *at least one* conforming DITA document type, as specified by the processor, but need not support any features not required by that document type.

  **Comment by Kristen Eberlein, 16 November 2013**

  Is the clause "as specified by the processor needed"? What does it add?

- A *specialization aware processor* is a DITA aware processor that can handle any document specialized from some set of supported vocabulary modules and with, possibly, the required use of specific constraint modules.

  **Comment by Kristen Eberlein, 16 November 2013**

  Why "some set of supported vocabulary modules"? Can we change this to "a set"? What do we mean by "and with, possibly, the required use of specific constraint modules"?

- A *fully aware processor* is a DITA aware processor that supports all base vocabulary modules without constraint, which implies support for all non-vocabulary-specific DITA features, such as content references and key references.

  **Comment by Kristen Eberlein, 16 November 2013**

  Is the use of the word "constraint" overloaded and ambiguous here?
For example, a general-purpose processor that can process XML documents to produce a particular output using user-created configurations or scripts is not itself DITA-aware. However, that same processor packaged with a DITA-specific configuration or script would be a DITA aware processor. A script or configuration for this processor that only operates on tag names as defined in specific DITA vocabulary modules would make the tool DITA aware but not specialization aware. A script or configuration that operated on DITA @class attribute values would be both DITA aware and specialization aware.

A **conforming DITA processor** is a DITA aware processor that implements all required processing relevant to that processor for the vocabulary modules that it claims to support. A DITA-aware processor **MUST** support at least one map or topic type, whether defined by the DITA standard or defined as a custom vocabulary module.

A **conforming specialization-aware processor** is a DITA aware processor that applies relevant processing to all DITA elements based on their @class and @domains attribute values.

**Note**: In general, specialization aware processors will be able to reliably process all conforming DITA documents, providing at least some default behavior for all DITA elements, while DITA aware processors might only be able to reliably process documents that use the vocabulary modules that those processors support.

DITA aware processors support the following functional areas:

**Production of output**

The production of final form output from DITA documents. Examples of processors that perform this function are publishing systems and tools, such as the DITA Open Toolkit.

**Editing, managing, and storage**

The editing, managing, and storage of DITA documents. Examples of processors that perform this function are authoring tools and component content management systems (CCMs).

A given processor might provide some or all of this function. For example, a DITA aware editor that includes the ability to generate print versions of DITA documents represents both a final-form processor and an editing processor. Likewise, a content or component management system might tightly integrate final-form DITA processors. Each processor type may have different conformance requirements, even though the processors are part of a single product or package.

For processors that produce final form output, all features that are relevant to the type of processing that the processor performs **MUST** be implemented, with the exception of features that are vocabulary-specific. In particular, such processors **MUST** implement address resolution and content reference resolution. Such processors **SHOULD** implement filtering.

For example, a specialization aware processor that produces final-form output need not provide special presentation results for glossary entry topics, but it must implement resolution of key-based references to glossary entry topics from `<keyword>` or `<term>` elements, because address resolution is both required and not vocabulary specific.

Processors that store, manage, or edit DITA documents might choose to not implement specific features that would be required for final-form processing. However, such processors **MUST** enable the creation or storage of DITA documents that use all DITA features, even if the processor is not aware of the DITA semantics for those features.

For example, a DITA aware editor need not provide specific support for creating or resolving content references, but it must allow, using normal XML editing methods, the creation and editing of content references. A content management system that supports map types that allow relationship tables but does not directly support relationship table processing must be able to store and manage conforming map documents that include relationship tables.
Appendix A Acknowledgments

(Non-normative) Many individuals participated in the creation of this specification and are gratefully acknowledged.

Robert Anderson, IBM
Deb Bissantz, LSI
Doug Burgess, SAP
Tom Cihak, Freescale Semiconductor, Inc.
Don Day, Individual Member
Stanley Doherty, Individual Member
Kristen James Eberlein, Individual Member
JoAnn Hackos, Comtech Services, Inc.
Richard Hamilton, Individual Member
Nancy Harrison, Individual Member
David Helfinstine, PTC
Scott Hudson, Schneider Electric Industries
John Hunt, IBM
Eliot Kimber, Individual Member
Tom Magliery, JustSystems
Mark Myers, SAP
Chris Nitchie, Oberon Technologies
Seth Park, Freescale Semiconductor, Inc.
Mark Poston, Mekon
Michael Priestley, IBM
Tony Self, Individual Member
Eric Sirois, IBM
Dawn Stevens, Comtech Services, Inc.
Amber Swope, Individual Member
Bob Thomas, Individual Member
Appendix B Non-normative information

This section contains non-normative information, including topics about new features in DITA 1.2 and migrating from DITA 1.1 to DITA 1.2.

B.1 Changes from previous versions

(Non-normative) The following topics outline the changes from earlier versions of DITA to the current version.

B.1.1 Changes from DITA 1.2 to DITA 1.3

(Non-normative) DITA 1.3 adds a number of new features and refinements to DITA.

Comment by Kristen Eberlein, 26 March 2014
Need to remove proposal numbers from text before release.

Enhancements to keys

- [13004]: Scoped keys
- [13041] Added facility for key-based, cross-deliverable referencing

New XML grammars

- [13112] The DITA standard now supports RELAX NG.

Enhancements to conditional processing (filtering)

- [13059a] Enabled information architects to associate DITAVAL conditions with specific topic collections in a map. Prior to DITA 1.3, a set of DITAVAL conditions could only be associated with a root map.
- [13056] Expanded syntax for filtering attributes. The four original-property attributes (@audience, @platform, @product, and @otherprops) now accept grouped values, using the same syntax for generalized attribute values within @props and @base.

Enhancements to addressing

- [13001] New syntax for addressing an element within the same DITA topic

Enhancements to specialization

- [13121] Enabled parts of structural specializations to be reused by other structural specializations without requiring one to be specialized from the other. For example, allowed the <steps> element from <task> to be reused in place of an <ol> within a <troubleshootingsteps> section of the troubleshooting document type.

Increased support for producing troubleshooting information

- [13098] A new value, "trouble", for the @type attribute of the <note> element
- [13086] A new element in the content model for <step>,<steptroubleshooting>, designed to contain information that might assist users when a step does not produce the expected result or complete successfully
- [13096] A new section in the task document type,<tasktroubleshooting>, designed to contain information that might assist users when a task does not produce the expected result or complete successfully
- [13097] A new troubleshooting document type

Increased support for producing user assistance

- [13008, 13060] Added the following new attributes to <resourceid>:
  - @appid
  - @ux-context-string
  - @ux-source-priority
• @ux-windowref

[13061] Added a new element, <ux-windowref>, designed to specify a window or viewport in which an online help topic or Web page is displayed

New domains

• [13035] Markup and XML mention domains, designed to support the description and documentation of XML document types and XML applications
• [13102] Release management domain, designed to enable content authors to log comments and metadata in DITA topics and maps when changes are made
• [13111] Eqauation and MathML domains, designed to enable use of MathML markup within DITA documents, as well as use-by-reference of MathML markup that is held in separate, non-DITA documents
• [13119] SVG domain, designed to enable use of SVG markup within DITA documents, as well as use-by-reference of SVG markup that is held in separate, non-DITA documents

Enhancements to the learning and training specialization

• [13089] Added the learning object map: <learningObjectMap>
• [13089] Added the learning group map: <learningGroupMap>
• [13106] New base domain and specialized domain for question and answer interactions

New base elements and attributes

• [13023] <div> element. This element enables arbitrary grouping of content within a topic. It is primarily intended as a way to group content for content referencing or as a basis for specialization.
• [13103] Deprecated the @print attribute and replaced it with a new conditional-processing attribute: @deliveryTarget. Users can set controlled values for the @deliveryTarget attribute by using a subject scheme map.
• [13078] Added @orient to <table> and @rotate to <entry>. These new attributes enable authors to control formatting more precisely for print-based output formats.
• [62] Added @scope and @headers attributes to the <entry> element in tables. These new attributes are designed to improve accessibility for very complex tables that cannot be simplified.
• [13059] Added a new attribute, @cascade, to better control how metadata attributes cascade within a DITA map

Modifications to the base content model

• [13027] Enabled <draft-comment> in more places
• [13029] Enabled <text> to be used in more places
• [13044] Extended content model of <data> to allow %basic.ph; rather than %words.cnt;
• [13090] Extended the content model of the @style attribute used on <prop> and <revprop> elements in DITAVAL files.
• [13092] Enabled <ph> within <indexterm>. This makes it possible to use specializations of the <ph> element, such as <sup> and <sub>, within index term entries.
• [13104] Enabled @keyref for <object> and <param>
• [13105] Added <data> to the content model for lists
• [13107] New defaults for @format and @scope attributes
• [13114] Added @rev attribute to <title>
• [13116] Added <cite> to the content model of <title> and similar elements
• [13117] Added <xref> to the content model for <shortdesc>

Refinements to base domains

• [13010] Added element for sorting, <sort-as>, to the utilities domain
• [13031] Added element for strikeouts, <line-through>, to the highlighting domain
B.1.2 Changes from DITA 1.1 to DITA 1.2

(Non-normative) DITA 1.2 adds a number of new features to DITA, including indirect addressing using map-defined keys; the ability to define content-model constraints for DITA document types; specializations for learning content and the machine industry; and taxonomies, ontologies, and controlled vocabularies. Other refinements include extended markup for glossaries and terminology.

New features

The following features are new in DITA 1.2:

- Keys and key references. See Key-based addressing.
- Constraint modules. Constraint modules allow base content models to be further restricted without the need for specialization. For example, a constraint module can make optional elements required or disallow optional elements in a specific content model. See Constraint domains.
- Topic and map specializations for learning and training information, including interactive assessments.
- New elements for use with glossary entry topics for more complete description of terms, definition of acronyms, and so on.
- New map specialization for defining controlled vocabularies and taxonomies. See subjectScheme.
- New machine-industry task specialization.

New element types

The following base element types are new in DITA 1.2:

- `<text>`
  - Allowed in most contexts where text is allowed but neither `<ph>` nor `<keyword>` are allowed. Enables reuse of text in almost any context.
- `<bodydiv>`
  - Allows creation of untitled containers within topic bodies. Intended primarily for specialization.
- `<sectiondiv>`
  - Allows creation of untitled containers within sections. Intended primarily for specialization.
- `<keydef>`
  - Topicref specialization for defining keys. Sets the default value for the processing-@role attribute to "resource-only".
- `<mapref>`
  - Topicref specialization for referring to DITA maps. Sets the default value for the @format attribute to "ditamap".
- `<topicset>`
  - Used to define sets of topicrefs that represent an atomic unit of reusable navigation structure. Requires the @id attribute be specified.
- `<topicsetref>`
  - References a `<topicset>` element. Enables preservation of the identity of the referenced topic set.
- `<anchor>`
  - Defines a point within a map to which `<topicref>` elements can be bound using the `<anchorref>` element.
- `<anchorref>`
  - "Pushes" one or more `<topicref>` elements onto an anchor point defined by an `<anchor>` element. Similar to a conref push but allows the relationship to be managed dynamically by the renderer.

Refinements to maps

- Map elements can use the `<title>` element in place of the @title attribute.
- Relationship table elements can have `<title>` as an optional first child.
• Topicref elements can use the `<navtitle>` element in place of the `@navtitle` attribute.
• Maps and topicrefs can now contain the same metadata elements as topic prologs.
• New `<topicref>` attribute named `@processing-role`. Indicates whether or not a topic reference contributes to the navigation structure of the containing map.

**Refinements to content references**

• Content references can now point to ranges of elements. For example, a single content reference from a `<step>` element can include a sequence of `<step>` elements.
• Content references can "push" elements into a target context, allowing unilateral augmentation of topics from other topics. For example, given a base topic with generic content, a using map could include both the generic topic and a separate topic that uses conref push to add map-specific content to the generic topic.
• Content reference resolution can be deferred so that it is done later in a rendering process or completely deferred so that it can be done by a separate delivery mechanism, for example, Eclipse information centers.

**Refinements to topic elements**

• The base task topic type has a more relaxed content model. This enables creation of a wider variety of specialized tasks, including task specializations that do not have formal markup for individual steps. The strict-task document type integrates a constraint module that imposes the same constrained content model as defined in the DITA 1.1 task topic type.
• A number of content elements allow the new `@keyref` attribute, including the `<ph>`, `<keyword>`, and `<term>` elements. When using the `@keyref` attribute, these elements can get their effective content from the key-defining `<topicref>` element and can also be treated as navigation links to the resource pointed to by the key-defining `<topicref>` element, if any. For example, a `<term>` element can use `@keyref` to link to the glossary entry topics for the term.
• The `<image>` element takes the new `@scalefit` attribute, which indicates whether or not the image should be scaled to fit the presentation context.
• The `<draft-comment>` element is now allowed in most contexts.
• The `<figgroup>` element now allows `<data>` as a subelement.

**Refinements to specialization**

• Structural and domain vocabulary modules can now both be listed in the `@domains` attribute. Structural modules can depend on and specialize elements from domains. For example, a structural domain for reference topics for a specific programming language could depend on the Programming domain (pr-d) and specialize elements from that domain.
• Information Architects can indicate whether the use of a given vocabulary module requires strict or weak checking of content reference constraints.
• The implementation patterns for vocabulary modules have been refined. In particular, each element type now defines a separate parameter entity for its content model and attribute list, allowing per-element configuration of content models and attribute lists through constraint modules.

**Other refinements**

• The `<dita>` element now has the `@DITAArchVersion` attribute.
• A number of processing details have been clarified where they were underspecified in DITA 1.1.
• Most attributes that had enumerated values in DITA 1.1 are now unenumerated, allowing specializations to define different enumerations if they choose.

**B.1.3 Changes from DITA 1.0 to DITA 1.1**

(Non-normative) The DITA 1.1 specification was designed to be backwards-compatible with applications that conform to the DITA 1.0 specification.

The following major changes to the architecture provided added functionality for DITA 1.1:

• A `<bookmap>` specialization for encoding book-specific information in a DITA map
• A `<glossentry>` specialization for glossary entries
• Indexing specializations for see, see-also, page ranges, and sort order
• Improvements to graphic scaling capability
• Improved short description flexibility through a new `<abstract>` element
• Specialization support for new global attributes, such as conditional processing attributes
• Support for integration of existing content structures through the `<foreign>` element
• Support for new kinds of information and structures through the `<data>` and `<unknown>` elements
• Formalization of conditional processing profiles

B.2 Migrating from DITA 1.1 to 1.2

(Non-normative) DITA 1.2 is compatible with prior versions of the DITA specification in that all valid DITA 1.0 and 1.1 documents are valid DITA 1.2 documents. However, some changes to existing document type shells and specializations might be needed in order to maintain the same behavior under DITA 1.2 or to take full advantage of new DITA 1.2 features.

Migration of task specializations

DITA 1.2 relaxes the content model for the `<taskbody>` element in the base `<task>` topic type. Specializations of `<task>` that use the `<taskbody>` element unchanged or use the content-model parameter entities for task body will use the more general DITA 1.2 task model when used with the DITA 1.2 task module.

If a document should use the strict task model instead of the new general task model:

• Documents that use the task topic type included with the OASIS DITA 1.2 specification will not need to migrate. This includes documents that use task inside of the ditabase document type included with the DITA 1.2 specification.
• Documents that use the standard task module inside of a new document type shell, or in a local ditabase document type, will need to migrate in order to preserve the module.
• Documents that use a specialization of task with a specialized element to replace `<taskbody>` will not need to migrate. Documents that specialize task but reuse the `<taskbody>` element will need to migrate.

For documents that do not use the standard OASIS DITA doctypes for which the strict task model from DITA 1.0 and 1.1 is desired, the strict task model can be restored by using the strict Taskbody Constraint module, by creating and integrating a constraint module similar to the strict Taskbody Constraint, or by adjusting the specialization's content models to impose the appropriate constraints.

To integrate the strict Taskbody Constraint module in a DTD, add the following content immediately before the section that imports the "topic.mod" file. The `strictTaskbodyConstraint.mod` file must be defined in any catalog used to locate DTD modules. See the `task.dtd` file in the OASIS DITA 1.2 DTD package for a sample:

```xml
<!ENTITY % strictTaskbody-c-def
  PUBLIC "-//OASIS//ELEMENTS DITA 1.2 Strict Taskbody Constraint//EN"
  "strictTaskbodyConstraint.mod">
%strictTaskbody-c-def;
```

To do the same thing for a schema, the `strictTaskbodyConstraintMod.xsd` file must be defined in the catalog used to resolve the schema using the following URN. The `<xs:include>` element should be placed immediately before the `<xs:include>` for `topicMod.xsd`:

```xml
<xs:include schemaLocation="urn:oasis:names:tc:dita:xsd:strictTaskbodyConstraintMod.xsd:1.2" />
```

Documents which use the task document type from OASIS and wish to make use of the general `<taskbody>` model must also be migrated. In this case, the documents must be updated to reference the General Task document type instead of the traditional Task document type. For DTD-based documents, this means that the document type declaration must change as follows within each task document:

```xml
DITA 1.1:
<!DOCTYPE task PUBLIC "-//OASIS//DTD DITA Task//EN"
```
For Schema based documents, the xsi:noNamespaceSchemaLocation attribute must change on each task element:

DITA 1.1:
<task xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:oasis:names:tc:dita:xsd:task.xsd"
DITA 1.2:
<task xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

Use of glossary document type shell and module files
The files named glossary.dtd, glossary.mod, and glossary.ent in DITA 1.1 have been renamed to glossentry.dtd, glossentry.mod, and glossentry.ent in order to follow the file naming requirements for vocabulary modules. The XML Schema files glossary.xsd, glossaryGrp.xsd, and glossaryMod.xsd have been renamed to glossentry.xsd, glossentryGrp.xsd, and glossentryMod.xsd. The DITA 1.2 entity resolution catalogs map the public identifiers for the glossary.* files to the corresponding glossentry.* files and URNs for the glossary*.xsd files to the corresponding glossentry*.xsd files, meaning that existing documents and shell document types that refer to the glossary.* files will continue to work with DITA 1.2.

However, shell document types that refer to the glossary.* files should be updated to refer to the corresponding glossentry.* files as it is convenient to do so.

Migration of specializations to reflect new implementation patterns
DITA 1.2 defines new implementation patterns for specializations. It is not necessary to update existing specialization modules to reflect these new patterns. However, specializations that want to enable the use of constraint modules must be updated to provide per-element-type content model and attribute list parameter entities.

Existing specializations may benefit from the new DITA 1.2 ability to have structural modules that depend on and specialize elements from domains.

Update of domains attributes for specializations
In DITA 1.1 the @domains attribute only listed the domains on which a given map or topic type depended. In DITA 1.2 the @domains attribute should also list structural modules and constraint modules, providing a complete list of the vocabulary modules that a given map or topic type depends on in the context of a given DITA document type.

Existing structural specializations should be updated to include the structural module names in their domains attribute configuration entities. Having a complete declaration of all structural and domains modules in the @domains attribute allows processors to determine the set of required modules for a given topic or map document without also examining all the @class attributes of all the elements in the document.

XML Schema declaration of domains attribute
For XML schema shell document types, the way the @domains attribute is declared has changed. In DITA 1.1 the @domains attribute was defined using an <xs:redefine> element. For DITA 1.2 the attribute is defined in document type shells and referenced from modules, like so:

```
<xs:attributeGroup name="domains-att">
  <xs:attribute name="domains" type="xs:string"
    default="(topic ui-d)
      (topic hi-d)
      (topic sw-d)"
```
XML Schema document type shells must be updated to use the new declaration mechanism in order to continue to be valid XSD schemas.

### B.3 DITA MIME type

(Non-normative) It is common for Web-based services to establish default actions for content based on the MIME type value sent in HTTP headers. For example, the "text/html" MIME type is what normally causes browsers to interpret the content of a web page as presentation-oriented markup, versus "text/plain" which would cause the markup to be displayed literally. A DITA MIME type enables applications to recognize content as DITA to enable special services such as semantically-informed search indexing or on-the-fly rendering in the browser.

The OASIS DITA Technical Committee has requested the registration of "application/dita+xml" with the IANA organization as the formally recognized DITA MIME type. This process is in progress, however the details are public so that implementors can make early use of the proposed value.

More information about the proposal is documented in this note of transmittal to OASIS for registration of the DITA MIME type: [http://lists.oasis-open.org/archives/dita/200806/msg00008.html](http://lists.oasis-open.org/archives/dita/200806/msg00008.html)

### B.4 Considerations for generalizing `<foreign>` elements

(Non-normative) Generalization of a DITA element employs the value of its `@class` attribute. The content of a `<foreign>` element, or of a specialization of the `<foreign>` element, cannot be generalized because there are no DITA elements in it, and therefore no `@class` attributes. The non-DITA contents can only be processed by appropriate non-DITA means.

Only the specialization of the `<foreign>` element can be generalized with normal rules.

Its contents would be exported to a separate file and replaced in-line with an `<object>` element. The `@data` attribute of the `<object>` element would point to the generated file, and the `@type` attribute of the `<object>` element would be set to the value "DITA-foreign".

If an `<object>` element is present within the `<foreign>` element (or specialization thereof) during generalization, it would not be included with the content that is exported to the separate file. This original object element is used to specify alternate content in publishing systems that cannot display the foreign content. It would not be modified except as the ordinary rules of generalization require it.

In the exported file, exported content would be enclosed within a root `<foreign>` element in order to accommodate the possibility that it may contain several main elements apart from the alternate content.

For easy recognition, the name of the exported file would start with "dita-generalized-" , and it is recommended that the file name also contain the topic ID, specialization type, and element ID or generated identifier.

#### Example: Simple object generalization

For example, a DITA document could contain a specialization of `<foreign>` for MathML. It could look like this:

```xml
<mathml class="+ topic/foreign mathml/mathml ">
  <math xmlns="http://www.w3.org/1998/Math/MathML"
    <mi>x</mi><mo>+</mo><mn>3</mn></math>
</mathml>
```
The `<mathml>` container is a DITA element, so it should be generalized using normal rules. The `<math>` element, which is not a DITA element, will be exported to another file. The `<object>` element will remain:

```
<foreign class="+ topic/foreign mathml/mathml ">
  <object data="dita-generalized-topicid_mathml1.xml" type="DITA-foreign"/>
  <object><desc>X plus three</desc></object>
</foreign>
```

Contents of `dita-generalized-topicid_mathml1.xml`:

```
<foreign class="+ topic/foreign mathml/mathml ">
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <mi>x</mi><mo>+</mo><mn>3</mn>
  </math>
</foreign>
```

Example: Multiple object generalization

An object might also contain multiple object elements:

```
<mathml class="+ topic/foreign mathml/mathml ">
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <mi>x</mi><mo>+</mo><mn>3</mn>
  </math>
  <object><desc>X plus three</desc></object>
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <mi>y</mi><mo>-</mo><mn>2</mn>
  </math>
</mathml>
```

The `<mathml>` container, which is a normal DITA element, should be generalized using normal rules. A file should be generated for each set of elements bounded by the container and any existing object elements. In this case, two files will be generated, and two new object elements added to the source.

The modified source:

```
<foreign class="+ topic/foreign mathml/mathml ">
  <object data="dita-generalized-topicid_mathml1.xml" type="DITA-foreign"/>
  <object><desc>X plus three</desc></object>
  <object data="dita-generalized-topicid_mathml2.xml" type="DITA-foreign"/>
</foreign>
```

The contents of `dita-generalized-topicid_mathml1.xml`, the first exported file:

```
<foreign class="+ topic/foreign mathml/mathml ">
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <mi>x</mi><mo>+</mo><mn>3</mn>
  </math>
</foreign>
```

The contents of `dita-generalized-topicid_mathml2.xml`, the second exported file:

```
<foreign class="+ topic/foreign mathml/mathml ">
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <mi>y</mi><mo>-</mo><mn>2</mn>
  </math>
</foreign>
```

**B.5 Summary of addressing elements**

This topic summarizes DITA elements that can be used to link to or address other items. The table describes how and why each element uses the addressing mechanism, rather than defining the element itself.

**Comment by robander, 30 July 2014**

This is non-normative information - it is just an informative list that contains no new information. It repeats information that is defined everywhere (mostly restating or partially rewording information from short descriptions...
in the language reference). It often describes what elements "might" do, when we would be better of referring to
the language ref for a more complete description of what it "does" do.

I've moved this from the "DITA Addressing" section of the base architectural spec, where it sat oddly after the
key definition section. I'm a little nervous about continuing to maintain this going forward but for the moment I've
just shifted it to the "non-normative" section.

Table 19: DITA addressing elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Description and notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;topicref&gt;</td>
<td>Establishes a relationship between the containing map and another map, DITA topic, or non-DITA resource when the @href or @keyref attribute is specified. When @processing-role is set to &quot;resource-only&quot;, it establishes a dependency on the target resource but does not contribute to the navigation tree. Might establish additional relationships between the referenced resource and other resources in the navigation hierarchy as determined by the values of the @collection-type attribute. By default, these additional relationships are bi-directional. The directionality of additional relationships can be controlled using the @linking attribute.</td>
</tr>
<tr>
<td>&lt;reltable&gt;</td>
<td>Establishes relations of a specific type (as defined by the relationship table) among topicref-linked resources where each row in the table establishes a single set of relationships among the topicref-linked resources in each cell of the row. Relationships defined in relationship tables are outside of any navigation structure defined by the map.</td>
</tr>
<tr>
<td>&lt;navref&gt;</td>
<td>Establishes a map-to-map relationship where the integration of the referenced map's navigation structure is deferred. The referenced map is processed independently from the referencing map and does not contribute to the key space of the referencing map.</td>
</tr>
<tr>
<td>&lt;link&gt;</td>
<td>Establishes a link from its containing topic to another resource. Any &lt;link&gt; element within a topic can be functionally replaced by the equivalent link defined in a relationship table. Likewise, topic-to-topic links defined by relationship tables can be replaced by the equivalent set of &lt;link&gt; elements in the topics involved.</td>
</tr>
<tr>
<td>&lt;xref&gt;</td>
<td>Establishes a navigation link from a topic abstract or body to another DITA element or non-DITA resource.</td>
</tr>
<tr>
<td>&lt;image&gt;</td>
<td>Links to an image that should be displayed at the point of reference.</td>
</tr>
<tr>
<td>Element</td>
<td>Description and notes</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>&lt;object&gt;</td>
<td>Links to a media object that should be displayed at the point of reference.</td>
</tr>
<tr>
<td>&lt;longdescref&gt;</td>
<td>Links to a long description for an image or object. Can be used in place of the @longdescref attribute on the parent image or object element.</td>
</tr>
<tr>
<td>&lt;longquoteref&gt;</td>
<td>Links to the source of a long quotation. Used in place of the @href or @keyref attribute on the &lt;lq&gt; element. Enables use of all the normal link-controlling attributes.</td>
</tr>
<tr>
<td>&lt;data-about&gt;</td>
<td>Establishes an explicit relationship between one or more &lt;data&gt; elements and the DITA element or non-DITA resource to which the data applies.</td>
</tr>
<tr>
<td>Elements that take @keyref but not @href</td>
<td>Establishes navigation links to the referenced DITA element or non-DITA resource when @keyref is specified and the key is bound to a topic, map, or non-DITA resource. If the linking element has empty content and the key definition has a matching subelement in its &lt;topicmeta&gt;, establishes a use-by-reference relationship to the matching element in the key definition. Includes &lt;ph&gt;, &lt;cite&gt;, &lt;keyword&gt;, and &lt;term&gt;.</td>
</tr>
<tr>
<td>&lt;imagemap&gt; (utilities domain)</td>
<td>Enables linking from defined areas that are overlaid on a graphic. Modeled on the HTML image map.</td>
</tr>
<tr>
<td>&lt;author&gt;</td>
<td>Might link to a resource that represents the author, for example, a biographical topic or image.</td>
</tr>
<tr>
<td>&lt;data&gt;</td>
<td>Might link to a resource that represents the metadata value.</td>
</tr>
<tr>
<td>&lt;fragref&gt; (programming domain)</td>
<td>Links to a syntax definition fragment.</td>
</tr>
<tr>
<td>&lt;lq&gt;</td>
<td>Might link to the source of the quotation.</td>
</tr>
<tr>
<td>&lt;publisher&gt;</td>
<td>Might link to a resource that represents the publisher, for example, the publisher Web site or a descriptive topic.</td>
</tr>
<tr>
<td>&lt;source&gt;</td>
<td>Might link to a description of the source for the topic.</td>
</tr>
<tr>
<td>&lt;synnoteref&gt; (programming domain)</td>
<td>Might link to a syntax note.</td>
</tr>
<tr>
<td>&lt;fn&gt;</td>
<td>Establishes a relationship between the content within which the footnote appears and the note itself, such that the footnote is an annotation of the content.</td>
</tr>
<tr>
<td>&lt;mathmlref&gt; (MathML domain)</td>
<td>References a MathML &lt;math&gt; element that is stored in a non-DITA document.</td>
</tr>
<tr>
<td>&lt;svgref&gt; (SVG domain)</td>
<td>References an SVG &lt;svg&gt; element that stored in a non-DITA document.</td>
</tr>
</tbody>
</table>
B.6 Element-by-element recommendations for translators

(Non-normative) This topic contains a list of all OASIS DITA elements. It includes recommendations on how to present the element type to translators, 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.

Since the distinction between block and inline elements is ultimately controlled by the container of the element and the processing associated with it, the same element may be a block in one context and an inline element in another. Specializing document types may vary this behavior according to the needs of the document type being created, and the distinctions given below are provided only as a guide to known behavior with the base DITA document types.

Notes on the tables below

- For specializations, the second column gives the ancestor element, and the third column gives a quick yes/no guide to indicate whether all behavior is inherited. If something is not inherited, the change will appear in bold.
- For any specialization not listed below, the suggested default is to fall back to the closest listed ancestor.
- The block/inline presentation column indicates whether the element is formatted as a single block.
- The block/inline translation column indicates whether the element represents a complete translatable segment. For example, the element `<cmd>` is presented inline with other elements, but represents a complete translation segment.
- Items marked as block*** are blocks on their own, but may appear in the middle of a segment. They should not break the flow of the current segment. These are considered "subflow" elements for translation. We recommend that, when possible, these elements should only be placed at sentence boundaries to aid in translation.
- For all elements, the `@translate` attribute will override the suggested default translation setting. So, a translation setting of "yes" or "no" in the table below does not guarantee that an element will always, or never, be translated.
- If an element has translatable attributes, they are listed in the last column. Note that the `@spectitle` and `@specentry` attributes are described with a footnote.
- 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 may 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, and is not part of a larger segment; in that location, `<keyword>` can be considered a "subflow" element.

**topic elements**

---

1. This element is considered a "subflow" element for translation. If it is located in the middle of a translation segment, it should not be translated as part of that segment. For example, `<indexterm>`, `<fn>`, and `<draft-comment>` may divide a sentence in two, but should be treated as blocks, and should not interrupt the sentence.
2. The `@spectitle` and `@specentry` attributes can contain translatable text. The direct use of fixed-in-the-DTD text by tools is discouraged, in favor of using the value as a lookup string to find the translation outside of the file, using accepted localization methods for generated text.
3. The block vs. inline designation for the `<foreign>` element is likely to change for some specializations.
4. The `<desc>`, `<object>`, and `<image>` elements inside `<foreign>` should still be translatable; they provide an alternative display if the foreign content cannot be processed.
5. The use of the `@alt` attribute is deprecated in favor of the `<alt>` element.
<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;abstract&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;alt&gt; Footnote.</td>
<td>N/A</td>
<td>block***Footnote.</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;audience&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;author&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;body&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;bodydiv&gt; (new in DITA 1.2)</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;boolean&gt;</td>
<td>N/A</td>
<td>inline</td>
<td>inline</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>&lt;brand&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;category&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;cite&gt;</td>
<td>N/A</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;colspec&gt;</td>
<td>N/A</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>&lt;component&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;copyrholder&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;copyright&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;copyryear&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;created&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;critdates&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;data&gt;</td>
<td>N/A</td>
<td>N/A (metadata)</td>
<td>block</td>
<td>no (likely to change for some specializations)</td>
<td></td>
</tr>
<tr>
<td>&lt;data-about&gt;</td>
<td>N/A</td>
<td>N/A (metadata)</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;dd&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;ddhd&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;desc&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;dl&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;dlentry&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;dlhead&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;draft-comment&gt;</td>
<td>N/A</td>
<td>block***Footnote.</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;dt&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;dthd&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

6 The desc, object, and image elements inside `<foreign>` should still be translatable; they provide an alternative display if the foreign content cannot be processed.
<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;entry&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;example&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote.</td>
</tr>
<tr>
<td>&lt;featnum&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;fig&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote.</td>
</tr>
<tr>
<td>&lt;figgroup&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;fn&gt;</td>
<td>N/A</td>
<td>block**Footnote.</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;foreign&gt; 7</td>
<td>N/A</td>
<td>blockFootnote.</td>
<td>block</td>
<td>yes</td>
<td>noFootnote.</td>
</tr>
<tr>
<td>&lt;image&gt;</td>
<td>N/A</td>
<td>block when @placement= break, otherwise inline</td>
<td>block when @placement= break, otherwise inline</td>
<td>yes</td>
<td>@altFootnote.</td>
</tr>
<tr>
<td>&lt;index-base&gt;</td>
<td>N/A</td>
<td>block**Footnote.</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;index-sort-as&gt; (new in DITA 1.2)</td>
<td>N/A</td>
<td>block***xref</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;indexterm&gt;</td>
<td>N/A</td>
<td>block**Footnote.</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;indextermref&gt;</td>
<td>N/A</td>
<td>inline</td>
<td>n/a</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>&lt;itemgroup&gt;</td>
<td>N/A</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;keyword&gt;</td>
<td>N/A</td>
<td>inline</td>
<td>inline (except when within <code>&lt;keywords&gt;</code> – see note above the table)</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;keywords&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;li&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lines&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote.</td>
</tr>
<tr>
<td>&lt;link&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;linkinfo&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote.</td>
</tr>
<tr>
<td>&lt;linklist&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote.</td>
</tr>
<tr>
<td>&lt;linkpool&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;linktext&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lq&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@reftitle</td>
</tr>
<tr>
<td>&lt;metadata&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

7 The block vs. inline designation for the `<foreign>` element is likely to change for some specializations.
<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;navtitle&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;no-topic-nesting&gt;</td>
<td>N/A</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>&lt;note&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@othertype,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>@spectitleFootnote.</td>
</tr>
<tr>
<td>&lt;object&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@standby</td>
</tr>
<tr>
<td>&lt;ol&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote.</td>
</tr>
<tr>
<td>&lt;othermeta&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td>@content</td>
</tr>
<tr>
<td>&lt;p&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;param&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>&lt;permissions&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;ph&gt;</td>
<td>N/A</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;platform&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;pre&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote.</td>
</tr>
<tr>
<td>&lt;prodinfo&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;prodname&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;prognum&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;prolog&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;publisher&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;q&gt;</td>
<td>N/A</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;related-links&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;required-cleanup&gt;</td>
<td>N/A</td>
<td>block**Footnote.</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;resourceid&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;revised&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;row&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;searchtitle&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;section&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;sectiondiv&gt; (new in DITA 1.2)</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;series&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;shortdesc&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Element name</td>
<td>Specialized from</td>
<td>Block/Inline (presentation)</td>
<td>Block/Inline (translation)</td>
<td>Translatable content?</td>
<td>Translatable attributes?</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
<td>-----------------------------</td>
<td>---------------------------</td>
<td>-----------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>&lt;simpletable&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle Footnote.</td>
</tr>
<tr>
<td>&lt;sl&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle Footnote.</td>
</tr>
<tr>
<td>&lt;sli&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;source&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;state&gt;</td>
<td>N/A</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td>@value</td>
</tr>
<tr>
<td>&lt;stentry&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@specentry Footnote.</td>
</tr>
<tr>
<td>&lt;sthead&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;strow&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;table&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;tbody&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;term&gt;</td>
<td>N/A</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;text&gt; <em>(new in DITA 1.2)</em></td>
<td>N/A</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;tgroup&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;thead&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;title&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;titlealts&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;tm&gt;</td>
<td>N/A</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;topic&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;ul&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle Footnote.</td>
</tr>
<tr>
<td>&lt;unknown&gt; <em>(new in DITA 1.1)</em></td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;vrmlist&gt;</td>
<td>N/A</td>
<td>block (metadata)</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;xref&gt;</td>
<td>N/A</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

**map elements**

As of DITA 1.2, the `@navtitle` attribute is deprecated, for translation purposes, in favor of the new `<navtitle>` element. The `<navtitle>` element is also available in topics, and is included in the topic table above.
<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;linktext&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;map&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@title 8</td>
</tr>
<tr>
<td>&lt;navref&gt;</td>
<td>N/A</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>&lt;relcell&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;relcolspec&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;relheader&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;relrow&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;reltable&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;searchtitle&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;shortdesc&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;topicmeta&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;topicref&gt;</td>
<td>N/A</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
</tbody>
</table>

**bookmap elements (new in DITA 1.1)**

The bookmap specialization contains many phrase-based elements inside the bookmeta. These are metadata, and should not be translated.

As of DITA 1.2, the @navtitle attribute is deprecated, for translation purposes, in favor of the new <navtitle> element.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everythin from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;abbrevlist&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;amendments&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;appendix&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;approved&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;backmatter&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;bibliolist&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;bookabstract&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;bookchangehistory&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;bookevent&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;bookeventtype&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;bookid&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

8  (deprecated as of DITA 1.2 for translation purposes)
<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;booklibrary&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;booklist&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;booklists&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;bookmap&gt;</td>
<td>&lt;map&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removed title attribute</td>
</tr>
<tr>
<td>&lt;bookmeta&gt;</td>
<td>&lt;topicmeta&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;booknumber&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;bookowner&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;bookpartno&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;bookrestriction&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;bookrights&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;booktitle&gt;</td>
<td>&lt;title&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;booktitlealt&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;chapter&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;colophon&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;completed&gt;</td>
<td>&lt;ph&gt;</td>
<td>no</td>
<td>inline</td>
<td>inline</td>
<td>no</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;copyrfirst&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;copyrlast&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;day&gt;</td>
<td>&lt;ph&gt;</td>
<td>no</td>
<td>inline</td>
<td>inline</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;dedication&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;draftintro&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;edited&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;edition&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;figurelist&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;frontmatter&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;glossarylist&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;indexlist&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;isbn&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;mainbooktitle&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;maintainer&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;month&gt;</td>
<td>&lt;ph&gt;</td>
<td>no</td>
<td>inline</td>
<td>inline</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Element name</td>
<td>Specialized from</td>
<td>Inherits everything from ancestor?</td>
<td>Block/Inline (presentation)</td>
<td>Block/Inline (translation)</td>
<td>Translatable content?</td>
<td>Translatable attributes?</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------</td>
<td>-------------------------------------</td>
<td>----------------------------</td>
<td>---------------------------</td>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>&lt;notices&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;organization&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;part&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;person&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;preface&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;printlocation&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;published&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;publisherinformation&gt;</td>
<td>&lt;publisher&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;publishertype&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;reviewed&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;revisionid&gt;</td>
<td>&lt;ph&gt;</td>
<td>no</td>
<td>inline</td>
<td>inline</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;started&gt;</td>
<td>&lt;ph&gt;</td>
<td>no</td>
<td>inline</td>
<td>inline</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;summary&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;tablelist&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;tested&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;toc&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;trademarklist&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;volume&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;year&gt;</td>
<td>&lt;ph&gt;</td>
<td>no</td>
<td>inline</td>
<td>inline</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

concept elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;conbody&gt;</td>
<td>&lt;body&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;conbodydiv&gt; (new in DITA 1.2)</td>
<td>&lt;bodydiv&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;concept&gt;</td>
<td>&lt;topic&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>
### Glossentry Elements (New in DITA 1.1)

<table>
<thead>
<tr>
<th>Element Name</th>
<th>Specialized From</th>
<th>Inherits Everything from Ancestor?</th>
<th>Block/Inline (Presentation)</th>
<th>Block/Inline (Translation)</th>
<th>Translatable Content?</th>
<th>Translatable Attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;glossAbbreviation&gt;</code></td>
<td><code>&lt;title&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td><em>(new in DITA 1.2)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;glossAcronym&gt;</code></td>
<td><code>&lt;title&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td><em>(new in DITA 1.2)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;glossAlt&gt;</code></td>
<td><code>&lt;section&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes @spectitle</td>
</tr>
<tr>
<td><em>(new in DITA 1.2)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;glossAlternateFor&gt;</code></td>
<td><code>&lt;xref&gt;</code></td>
<td>yes</td>
<td>inline</td>
<td>n/a (empty element)</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td><em>(new in DITA 1.2)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;glossbody&gt;</code></td>
<td><code>&lt;body&gt;</code>, <code>&lt;conbody&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td><em>(new in DITA 1.2)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;glossdef&gt;</code></td>
<td><code>&lt;abstract&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td><code>&lt;glossentry&gt;</code></td>
<td><code>&lt;topic&gt;</code>, <code>&lt;concept&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td><code>&lt;glossPartOfSpeech&gt;</code></td>
<td><code>&lt;data&gt;</code></td>
<td>yes</td>
<td>N/A (metadata)</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td><em>(new in DITA 1.2)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;glossProperty&gt;</code></td>
<td><code>&lt;data&gt;</code></td>
<td>yes</td>
<td>N/A (metadata)</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td><em>(new in DITA 1.2)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;glossScopeNote&gt;</code></td>
<td><code>&lt;note&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@othertype</td>
</tr>
<tr>
<td><em>(new in DITA 1.2)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;glossShortForm&gt;</code></td>
<td><code>&lt;title&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td><em>(new in DITA 1.2)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;glossStatus&gt;</code></td>
<td><code>&lt;data&gt;</code></td>
<td>yes</td>
<td>N/A (metadata)</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td><em>(new in DITA 1.2)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;glossSurfaceForm&gt;</code></td>
<td><code>&lt;p&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td><em>(new in DITA 1.2)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;glossSymbol&gt;</code></td>
<td><code>&lt;image&gt;</code></td>
<td>yes</td>
<td>block when @placement= break, otherwise inline</td>
<td>block when @placement= break, otherwise inline</td>
<td>yes</td>
<td>removes @alt</td>
</tr>
<tr>
<td><em>(new in DITA 1.2)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;glossSynonym&gt;</code></td>
<td><code>&lt;title&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td><em>(new in DITA 1.2)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;glossTerm&gt;</code></td>
<td><code>&lt;title&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td><em>(new in DITA 1.2)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>&lt;glossUsage&gt;</code></td>
<td><code>&lt;note&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@othertype (removes @spectitle)</td>
</tr>
<tr>
<td><em>(new in DITA 1.2)</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### reference elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;propdesc&gt;</td>
<td>&lt;stentry&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@specentryFootnote</td>
</tr>
<tr>
<td>&lt;propdeschd&gt;</td>
<td>&lt;stentry&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@specentryFootnote</td>
</tr>
<tr>
<td>&lt;properties&gt;</td>
<td>&lt;simpletable&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;property&gt;</td>
<td>&lt;strow&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;prophead&gt;</td>
<td>&lt;sthead&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@specentryFootnote</td>
</tr>
<tr>
<td>&lt;proptype&gt;</td>
<td>&lt;stentry&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@specentryFootnote</td>
</tr>
<tr>
<td>&lt;proptypehd&gt;</td>
<td>&lt;stentry&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@specentryFootnote</td>
</tr>
<tr>
<td>&lt;propvalue&gt;</td>
<td>&lt;stentry&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@specentryFootnote</td>
</tr>
<tr>
<td>&lt;propvaluehd&gt;</td>
<td>&lt;stentry&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@specentryFootnote</td>
</tr>
<tr>
<td>&lt;refbody&gt;</td>
<td>&lt;body&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@specentryFootnote</td>
</tr>
<tr>
<td>&lt;refbodydiv&gt;</td>
<td>(new in DITA 1.2)</td>
<td>&lt;bodydiv&gt;</td>
<td>yes</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;reference&gt;</td>
<td>&lt;topic&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;refsyn&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

### task elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;chdesc&gt;</td>
<td>&lt;stentry&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@specentryFootnote</td>
</tr>
<tr>
<td>&lt;chdeschd&gt;</td>
<td>&lt;stentry&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@specentryFootnote</td>
</tr>
<tr>
<td>&lt;chhead&gt;</td>
<td>&lt;sthead&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@specentryFootnote</td>
</tr>
<tr>
<td>&lt;choice&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;choices&gt;</td>
<td>&lt;ul&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes spectitle</td>
</tr>
</tbody>
</table>

---

**DITA Version 1.3 Specification**

TBD

Copyright © OASIS Open 2005, 2014. All Rights Reserved.

Page 702 of 747
<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;choicetable&gt;</td>
<td>&lt;simpletable&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote.</td>
</tr>
<tr>
<td>&lt;choption&gt;</td>
<td>&lt;stentry&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@specentryFootnote.</td>
</tr>
<tr>
<td>&lt;choptionhd&gt;</td>
<td>&lt;stentry&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@specentryFootnote.</td>
</tr>
<tr>
<td>&lt;chrow&gt;</td>
<td>&lt;strow&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@specentryFootnote.</td>
</tr>
<tr>
<td>&lt;cmd&gt;</td>
<td>&lt;ph&gt;</td>
<td>NO</td>
<td>inline</td>
<td>block</td>
<td>yes</td>
<td>removes @spectitle</td>
</tr>
<tr>
<td>&lt;context&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes @spectitle</td>
</tr>
<tr>
<td>&lt;info&gt;</td>
<td>&lt;itemgroup&gt;</td>
<td>NO</td>
<td>inline</td>
<td>block</td>
<td>yes</td>
<td>removes @spectitle</td>
</tr>
<tr>
<td>&lt;postreq&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes @spectitle</td>
</tr>
<tr>
<td>&lt;prereq&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes @spectitle</td>
</tr>
<tr>
<td>&lt;result&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes @spectitle</td>
</tr>
<tr>
<td>&lt;step&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes @spectitle</td>
</tr>
<tr>
<td>&lt;stepresult&gt;</td>
<td>&lt;itemgroup&gt;</td>
<td>NO</td>
<td>inline</td>
<td>block</td>
<td>yes</td>
<td>removes @spectitle</td>
</tr>
<tr>
<td>&lt;steps&gt;</td>
<td>&lt;ol&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes @spectitle</td>
</tr>
<tr>
<td>&lt;steps-informal&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes @spectitle</td>
</tr>
<tr>
<td>&lt;steps-unordered&gt;</td>
<td>&lt;ul&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes @spectitle</td>
</tr>
<tr>
<td>&lt;stepsection&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;stepxmp&gt;</td>
<td>&lt;itemgroup&gt;</td>
<td>NO</td>
<td>inline</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;substep&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes @spectitle</td>
</tr>
<tr>
<td>&lt;substeps&gt;</td>
<td>&lt;ol&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes @spectitle</td>
</tr>
<tr>
<td>&lt;task&gt;</td>
<td>&lt;topic&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes @spectitle</td>
</tr>
<tr>
<td>&lt;taskbody&gt;</td>
<td>&lt;body&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;tutorialinfo&gt;</td>
<td>&lt;itemgroup&gt;</td>
<td>NO</td>
<td>inline</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>
### hi-d elements (highlight domain)

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;b&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;i&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;sub&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;sup&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;tt&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;u&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

### indexing-d elements (indexing domain) (new in DITA 1.1)

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;index-see&gt;</td>
<td>&lt;index-base&gt;</td>
<td>yes</td>
<td>block<strong>Footnote.</strong></td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;index-see-also&gt;</td>
<td>&lt;index-base&gt;</td>
<td>yes</td>
<td>block<strong>Footnote.</strong></td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;index-sort-as&gt;</td>
<td>&lt;index-base&gt;</td>
<td>yes</td>
<td>block<strong>Footnote.</strong></td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

### pr-d elements (programming domain)

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;apiname&gt;</td>
<td>&lt;keyword&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;codeblock&gt;</td>
<td>&lt;pre&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;codeph&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td>@spectitleFootnote.</td>
</tr>
<tr>
<td>&lt;coderef&gt;</td>
<td>&lt;xref&gt;</td>
<td>yes</td>
<td>inline</td>
<td>N/A (empty element)</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>&lt;delim&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;fragment&gt;</td>
<td>&lt;figgroup&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;fragref&gt;</td>
<td>&lt;xref&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;groupchoice&gt;</td>
<td>&lt;figgroup&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;groupcomp&gt;</td>
<td>&lt;figgroup&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Element name</td>
<td>Specialized from</td>
<td>Inherits everything from ancestor?</td>
<td>Block/Inline (presentation)</td>
<td>Block/Inline (translation)</td>
<td>Translatable content?</td>
<td>Translatable attributes?</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
<td>-----------------------------------</td>
<td>-----------------------------</td>
<td>-----------------------------</td>
<td>----------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>&lt;groupseq&gt;</td>
<td>&lt;figgroup&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;kwd&gt;</td>
<td>&lt;keyword&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;oper&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;option&gt;</td>
<td>&lt;keyword&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;parml&gt;</td>
<td>&lt;dl&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote.</td>
</tr>
<tr>
<td>&lt;parmname&gt;</td>
<td>&lt;keyword&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;pd&gt;</td>
<td>&lt;dd&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;plentry&gt;</td>
<td>&lt;dlentry&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;pt&gt;</td>
<td>&lt;dt&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;repsep&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;sep&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;synblk&gt;</td>
<td>&lt;figgroup&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;synnote&gt;</td>
<td>&lt;fn&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;synnoteref&gt;</td>
<td>&lt;xref&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;synph&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;syntaxdiagram&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes @spectitle</td>
</tr>
<tr>
<td>&lt;var&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

**sw-d elements (software domain)**

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;cmdname&gt;</td>
<td>&lt;keyword&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;filepath&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;msgblock&gt;</td>
<td>&lt;pre&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote.</td>
</tr>
<tr>
<td>&lt;msgnum&gt;</td>
<td>&lt;keyword&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;msgph&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;systemoutput&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;userinput&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;varname&gt;</td>
<td>&lt;keyword&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>
### ui-d elements (UI domain)

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;menuscascade&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td>&lt;screen&gt;</td>
<td>&lt;pre&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td>&lt;shortcut&gt;</td>
<td>&lt;keyword&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;uicontrol&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;wintitle&gt;</td>
<td>&lt;keyword&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

### ut-d elements (Utilities domain)

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;area&gt;</td>
<td>&lt;figgroup&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;coords&gt;</td>
<td>&lt;ph&gt;</td>
<td>NO</td>
<td>inline</td>
<td>inline</td>
<td>no</td>
<td>@spectitle</td>
</tr>
<tr>
<td>&lt;imagemap&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>(can contain translatable alternate text) @spectitle</td>
</tr>
<tr>
<td>&lt;shape&gt;</td>
<td>&lt;keyword&gt;</td>
<td>NO</td>
<td>inline</td>
<td>inline</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

### mapgroup-d elements (mapgroup domain)

As of DITA 1.2, the @navtitle attribute is deprecated, for translation purposes, in favor of the new <navtitle> element.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;anchorref&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>(new in DITA 1.2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;keydef&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>(new in DITA 1.2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;mapref&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>(new in DITA 1.2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;topicgroup&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;topichread&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;topicset&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>(new in DITA 1.2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Element name</td>
<td>Specialized from</td>
<td>Inherits everything from ancestor?</td>
<td>Block/Inline (presentation)</td>
<td>Block/Inline (translation)</td>
<td>Translatable content?</td>
<td>Translatable attributes?</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
<td>-------------------------------------</td>
<td>-----------------------------</td>
<td>---------------------------</td>
<td>------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>&lt;topicsetref&gt; (new in DITA 1.2)</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
</tbody>
</table>

**xnal-d elements (XNAL domain) (new in DITA 1.1)**

The XNAL information is all metadata, so it generally does not need to be translated. Exceptions may be needed when selections from this metadata are used for display purposes. The standard behavior may need to change based on local business rules. For example, in some cases it may be appropriate to translate the honorific, country, or organizationname elements.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;addressdetails&gt;</td>
<td>&lt;ph&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;administrativearea&gt;</td>
<td>&lt;ph&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;authorinformation&gt;</td>
<td>&lt;author&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;contactnumber&gt;</td>
<td>&lt;data&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;contactnumbers&gt;</td>
<td>&lt;data&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;country&gt;</td>
<td>&lt;ph&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;emailaddress&gt;</td>
<td>&lt;data&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;emailaddresses&gt;</td>
<td>&lt;data&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;firstname&gt;</td>
<td>&lt;data&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;generationidentifier&gt;</td>
<td>&lt;data&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;honorific&gt;</td>
<td>&lt;data&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;lastname&gt;</td>
<td>&lt;data&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;locality&gt;</td>
<td>&lt;ph&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;localityname&gt;</td>
<td>&lt;ph&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;middlename&gt;</td>
<td>&lt;data&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;namedetails&gt;</td>
<td>&lt;data&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;organizationinfo&gt;</td>
<td>&lt;data&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;organizationname&gt;</td>
<td>&lt;data&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;organizationnamedetails&gt;</td>
<td>&lt;ph&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;otherinfo&gt;</td>
<td>&lt;data&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>&lt;personinfo&gt;</td>
<td>&lt;data&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Element name</td>
<td>Specialized from</td>
<td>Inherits everything from ancestor?</td>
<td>Block/Inline (presentation)</td>
<td>Block/Inline (translation)</td>
<td>Translatable content?</td>
<td>Translatable attributes?</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------</td>
<td>-------------------------------------</td>
<td>-----------------------------</td>
<td>---------------------------</td>
<td>-----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>&lt;personname&gt;</td>
<td>&lt;data&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;postalcode&gt;</td>
<td>&lt;ph&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;thoroughfare&gt;</td>
<td>&lt;ph&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;url&gt;</td>
<td>&lt;data&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;urls&gt;</td>
<td>&lt;data&gt;</td>
<td>no</td>
<td>block</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

**Delayed Conref Resolution domain elements (new in DITA 1.2)**

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;anchorid&gt;</td>
<td>&lt;keyword&gt;</td>
<td>no</td>
<td>N/A (metadata)</td>
<td>n/a</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;anchorkey&gt;</td>
<td>&lt;keyword&gt;</td>
<td>no</td>
<td>N/A (metadata)</td>
<td>n/a</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;exportanchors&gt;</td>
<td>&lt;keywords&gt;</td>
<td>no</td>
<td>N/A (metadata)</td>
<td>n/a</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

**Hazard Statement Domain (new in DITA 1.2)**

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;consequence&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;hazardstatement&gt;</td>
<td>&lt;note&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;hazardsymbol&gt;</td>
<td>&lt;image&gt;</td>
<td>yes</td>
<td>block when @placement=break, otherwise inline</td>
<td>block when @placement=break, otherwise inline</td>
<td>yes</td>
<td>removes @alt</td>
</tr>
<tr>
<td>&lt;howtoavoid&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;messagepanel&gt;</td>
<td>&lt;ul&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;typeofhazard&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>
### glossgroup elements (new in DITA 1.2)

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;glossgroup&gt;</td>
<td>&lt;topic&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

### glossref domain elements (new in DITA 1.2)

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;glossref&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
</tbody>
</table>

### abbreviated-form domain elements (new in DITA 1.2)

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;abbreviated-form&gt;</td>
<td>&lt;term&gt;</td>
<td>yes</td>
<td>inline</td>
<td>N/A (empty element)</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

### task requirements domain (new in DITA 1.2)

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;closereqs&gt;</td>
<td>&lt;section&gt;, &lt;postreq&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;esttime&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;noconds&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;nosafety&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;nospares&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A (metadata)</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;nosupeq&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A (metadata)</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;nosupply&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A (metadata)</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;perscat&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;perskill&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;personnel&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;prelreqs&gt;</td>
<td>&lt;section&gt;, &lt;prereq&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;reqcond&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Element name</td>
<td>Specialized from</td>
<td>Inherits everything from ancestor?</td>
<td>Block/Inline (presentation)</td>
<td>Block/Inline (translation)</td>
<td>Translatable content?</td>
<td>Translatable attributes?</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
<td>------------------------------------</td>
<td>-----------------------------</td>
<td>---------------------------</td>
<td>-----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>&lt;reqconds&gt;</td>
<td>&lt;ol&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;reqcontp&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;reqpers&gt;</td>
<td>&lt;ol&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;safecond&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;safety&gt;</td>
<td>&lt;ol&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;spare&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;spares&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote.</td>
</tr>
<tr>
<td>&lt;sparesli&gt;</td>
<td>&lt;ul&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote.</td>
</tr>
<tr>
<td>&lt;supeqli&gt;</td>
<td>&lt;ul&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote.</td>
</tr>
<tr>
<td>&lt;supeqol&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;supplies&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;supply&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;supplyli&gt;</td>
<td>&lt;ul&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote.</td>
</tr>
</tbody>
</table>

**classification domain elements (new in DITA 1.2)**

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;subjectCell&gt;</td>
<td>&lt;relcell&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;subjectref&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;topicapply&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;topicCell&gt;</td>
<td>&lt;relcell&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;topicsubject&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;topicSubjectHeader&gt;</td>
<td>&lt;relrow&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;topicSubjectRow&gt;</td>
<td>&lt;relrow&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;topicSubjectTable&gt;</td>
<td>&lt;reltable&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>
### subject scheme elements (new in DITA 1.2)

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;attributedef&gt;</code></td>
<td><code>&lt;data&gt;</code></td>
<td>yes</td>
<td>N/A (metadata)</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td><code>&lt;defaultSubject&gt;</code></td>
<td><code>&lt;topicref&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td><code>%navtitle</code></td>
</tr>
<tr>
<td><code>&lt;elementdef&gt;</code></td>
<td><code>&lt;data&gt;</code></td>
<td>yes</td>
<td>N/A (metadata)</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td><code>&lt;enumerationdef&gt;</code></td>
<td><code>&lt;topicref&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td><code>%navtitle</code></td>
</tr>
<tr>
<td><code>&lt;hasInstance&gt;</code></td>
<td><code>&lt;topicref&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td><code>%navtitle</code></td>
</tr>
<tr>
<td><code>&lt;hasKind&gt;</code></td>
<td><code>&lt;topicref&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td><code>%navtitle</code></td>
</tr>
<tr>
<td><code>&lt;hasNarrower&gt;</code></td>
<td><code>&lt;topicref&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td><code>%navtitle</code></td>
</tr>
<tr>
<td><code>&lt;hasPart&gt;</code></td>
<td><code>&lt;topicref&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td><code>%navtitle</code></td>
</tr>
<tr>
<td><code>&lt;hasRelated&gt;</code></td>
<td><code>&lt;topicref&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td><code>%navtitle</code></td>
</tr>
<tr>
<td><code>&lt;relatedSubjects&gt;</code></td>
<td><code>&lt;topicref&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td><code>%navtitle</code></td>
</tr>
<tr>
<td><code>&lt;subjectdef&gt;</code></td>
<td><code>&lt;topicref&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td><code>%navtitle</code></td>
</tr>
<tr>
<td><code>&lt;subjectHead&gt;</code></td>
<td><code>&lt;topicref&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td><code>%navtitle</code></td>
</tr>
<tr>
<td><code>&lt;subjectHeadMeta&gt;</code></td>
<td><code>&lt;topicmeta&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td><code>&lt;schemeref&gt;</code></td>
<td><code>&lt;topicref&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td><code>%navtitle</code></td>
</tr>
<tr>
<td><code>&lt;subjectRel&gt;</code></td>
<td><code>&lt;relrow&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td><code>&lt;subjectRelHeader&gt;</code></td>
<td><code>&lt;relrow&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td><code>&lt;subjectRelTable&gt;</code></td>
<td><code>&lt;reltable&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td><code>&lt;subjectRole&gt;</code></td>
<td><code>&lt;relcell&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td><code>&lt;subjectScheme&gt;</code></td>
<td><code>&lt;map&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>%title removes</td>
</tr>
</tbody>
</table>

### learningAssessment elements (new in DITA 1.2)

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;learningAssessment&gt;</code></td>
<td><code>&lt;topic&gt;</code>, <code>&lt;learningBase&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td><code>&lt;learningAssessmentbody&gt;</code></td>
<td><code>&lt;body&gt;</code>, <code>&lt;learningBasebody&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>
### learning base elements (new in DITA 1.2)

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;lcAudience&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td>&lt;lcChallenge&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>Footnote.</td>
</tr>
<tr>
<td>&lt;lcDuration&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>Footnote.</td>
</tr>
<tr>
<td>&lt;lcInstruction&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>Footnote.</td>
</tr>
<tr>
<td>&lt;lcInteraction&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>Footnote.</td>
</tr>
<tr>
<td>&lt;lcInteractionBase&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>Footnote.</td>
</tr>
<tr>
<td>&lt;lcIntro&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>Footnote.</td>
</tr>
<tr>
<td>&lt;lcNextSteps&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>Footnote.</td>
</tr>
<tr>
<td>&lt;lcObjective&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>Footnote.</td>
</tr>
<tr>
<td>&lt;lcObjectives&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>Footnote.</td>
</tr>
<tr>
<td>&lt;lcObjectivesGroup&gt;</td>
<td>&lt;ul&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>Footnote.</td>
</tr>
<tr>
<td>&lt;lcObjectivesStem&gt;</td>
<td>&lt;ph&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcPrereqs&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>Footnote.</td>
</tr>
<tr>
<td>&lt;lcResources&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>Footnote.</td>
</tr>
<tr>
<td>&lt;lcReview&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>Footnote.</td>
</tr>
<tr>
<td>&lt;lcSummary&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>Footnote.</td>
</tr>
<tr>
<td>&lt;lcTime&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>inline</td>
<td>inline</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;learningBase&gt;</td>
<td>&lt;topic&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;learningBasebody&gt;</td>
<td>&lt;body&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>
### learningContent elements (new in DITA 1.2)

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;learningContent&gt;</td>
<td>&lt;topic&gt;, &lt;learningBase&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;learningContentbody&gt;</td>
<td>&lt;body&gt;, &lt;learningBasebody&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

### learningOverview elements (new in DITA 1.2)

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;learningOverview&gt;</td>
<td>&lt;topic&gt;, &lt;learningBase&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;learningOverviewbody&gt;</td>
<td>&lt;body&gt;, &lt;learningBasebody&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

### learningSummary elements (new in DITA 1.2)

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;learningSummary&gt;</td>
<td>&lt;topic&gt;, &lt;learningBase&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;learningSummarybody&gt;</td>
<td>&lt;body&gt;, &lt;learningBasebody&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

### learningPlan elements (new in DITA 1.2)

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;lcAge&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcAssessment&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcAttitude&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcBackground&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcCIN&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle Footnote.</td>
</tr>
<tr>
<td>&lt;lcClassroom&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle Footnote.</td>
</tr>
<tr>
<td>Element name</td>
<td>Specialized from</td>
<td>Inherits everything from ancestor?</td>
<td>Block/Inline (presentation)</td>
<td>Block/Inline (translation)</td>
<td>Translatable content?</td>
<td>Translatable attributes?</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------</td>
<td>------------------------------------</td>
<td>-----------------------------</td>
<td>---------------------------</td>
<td>-----------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>&lt;lcClient&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcConstraints&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcDelivDate&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcDelivery&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcDownloadTime&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcEdLevel&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcFileSizeLimitations&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcGapAnalysis&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcGapItem&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcGapItemDelta&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcGeneralDescription&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcGoals&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcGraphics&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcHandouts&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcIntervention&gt;</td>
<td>&lt;section&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcInterventionItem&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcJtaItem&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcKnowledge&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcLearnStrat&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcLMS&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcModDate&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcMotivation&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>&lt;lcNeeds&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitleFootnote</td>
</tr>
<tr>
<td>Element name</td>
<td>Specialized from</td>
<td>Inherits everything from ancestor?</td>
<td>Block/Inline (presentation)</td>
<td>Block/Inline (translation)</td>
<td>Translatable content?</td>
<td>Translatable attributes?</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------</td>
<td>-------------------------------------</td>
<td>-----------------------------</td>
<td>-----------------------------</td>
<td>-----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td><code>&lt;lcNeedsAnalysis&gt;</code></td>
<td><code>&lt;section&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcNoLMS&gt;</code></td>
<td><code>&lt;fig&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcOJT&gt;</code></td>
<td><code>&lt;fig&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcOrganizational&gt;</code></td>
<td><code>&lt;fig&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcOrgConstraints&gt;</code></td>
<td><code>&lt;p&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcPlan Audience&gt;</code></td>
<td><code>&lt;fig&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcPlanDescrip&gt;</code></td>
<td><code>&lt;fig&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcPlanObjective&gt;</code></td>
<td><code>&lt;p&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcPlanPrereqs&gt;</code></td>
<td><code>&lt;fig&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcPlanResources&gt;</code></td>
<td><code>&lt;p&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcPlanSubject&gt;</code></td>
<td><code>&lt;fig&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcPlanTitle&gt;</code></td>
<td><code>&lt;fig&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcPlayers&gt;</code></td>
<td><code>&lt;fig&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcProcesses&gt;</code></td>
<td><code>&lt;p&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcProject&gt;</code></td>
<td><code>&lt;section&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcResolution&gt;</code></td>
<td><code>&lt;fig&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcSecurity&gt;</code></td>
<td><code>&lt;fig&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcSkills&gt;</code></td>
<td><code>&lt;p&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcSpecChars&gt;</code></td>
<td><code>&lt;p&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcTask&gt;</code></td>
<td><code>&lt;fig&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcTaskItem&gt;</code></td>
<td><code>&lt;p&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td><code>&lt;lcTechnical&gt;</code></td>
<td><code>&lt;section&gt;</code></td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle</td>
</tr>
<tr>
<td>Element name</td>
<td>Specialized from</td>
<td>Inherits everything from ancestor?</td>
<td>Block/Inline (presentation)</td>
<td>Block/Inline (translation)</td>
<td>Translatable content?</td>
<td>Translatable attributes?</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------</td>
<td>-------------------------------------</td>
<td>----------------------------</td>
<td>----------------------------</td>
<td>------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>&lt;lcValues&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcViewers&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle Footnote</td>
</tr>
<tr>
<td>&lt;lcW3C&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle Footnote</td>
</tr>
<tr>
<td>&lt;lcWorkEnv&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle Footnote</td>
</tr>
<tr>
<td>&lt;lcWorkEnvDescription&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle Footnote</td>
</tr>
<tr>
<td>&lt;learningPlan&gt;</td>
<td>&lt;topic&gt;, &lt;learningBase&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;learningPlanbody&gt;</td>
<td>&lt;body&gt;, &lt;learningBasebody&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

**learning map domain elements (new in DITA 1.2)**

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;learningContentRef&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;learningGroup&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;learningObject&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;learningOverviewRef&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;learningPlanRef&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;learningPostAssessmentRef&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;learningPreAssessmentRef&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
<tr>
<td>&lt;learningSummaryRef&gt;</td>
<td>&lt;topicref&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@navtitle</td>
</tr>
</tbody>
</table>

**Learning domain elements (new in DITA 1.2)**

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;lcAnswerContent&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcAnswerOption&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcAnswerOptionGroup&gt;</td>
<td>&lt;ul&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcArea&gt;</td>
<td>&lt;figgroup&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcAreaCoords&gt;</td>
<td>&lt;ph&gt;</td>
<td>NO</td>
<td>inline</td>
<td>inline</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Element name</td>
<td>Specialized from</td>
<td>Inherits everything from ancestor?</td>
<td>Block/Inline (presentation)</td>
<td>Block/Inline (translation)</td>
<td>Translatable content?</td>
<td>Translatable attributes?</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------------------------------------</td>
<td>-------------------------------------</td>
<td>-----------------------------</td>
<td>-----------------------------</td>
<td>-----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>&lt;lcAreaShape&gt;</td>
<td>&lt;keyword&gt;</td>
<td>NO</td>
<td>inline</td>
<td>inline</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;lcAsset&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcCorrectResponse&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;lcFeedback&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcFeedbackCorrect&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcFeedbackIncorrect&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcHotspot&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcHotspotMap&gt;</td>
<td>&lt;figgroup&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcInstructornote&gt;</td>
<td>&lt;note&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>@spectitle Footnote.</td>
</tr>
<tr>
<td>&lt;lcItem&gt;</td>
<td>&lt;stentry&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes @specentry</td>
</tr>
<tr>
<td>&lt;lcMatching&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes @specentry</td>
</tr>
<tr>
<td>&lt;lcMatchingItem&gt;</td>
<td>&lt;stentry&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes @specentry</td>
</tr>
<tr>
<td>&lt;lcMatchingItemFeedback&gt;</td>
<td>&lt;stentry&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes @specentry</td>
</tr>
<tr>
<td>&lt;lcMatchTable&gt;</td>
<td>&lt;simpletable&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td>removes @spectitle</td>
</tr>
<tr>
<td>&lt;lcMatchingHeader&gt;</td>
<td>&lt;sthead&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcMatchingPair&gt;</td>
<td>&lt;strow&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcMultipleSelect&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcOpenAnswer&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcOpenQuestion&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcQuestion&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcQuestionBase&gt;</td>
<td>&lt;p&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcSequence&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;lcSequenceOption&gt;</td>
<td>&lt;li&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcSequenceOptionGroup&gt;</td>
<td>&lt;ol&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcSequencing&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcSingleSelect&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lcTrueFalse&gt;</td>
<td>&lt;fig&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>
## learning metadata domain (new in DITA 1.2)

<table>
<thead>
<tr>
<th>Element name</th>
<th>Specialized from</th>
<th>Inherits everything from ancestor?</th>
<th>Block/Inline (presentation)</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
<th>Translatable attributes?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;lcLom&gt;</td>
<td>&lt;metadata&gt;</td>
<td>yes</td>
<td>block</td>
<td>block</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>&lt;lomAggregationLevel&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;lomContext&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;lomCoverage&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;lomDifficulty&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;lomInstallationRemarks&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;lomIntendedUserRole&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;lomInteractivityLevel&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;lomInteractivityType&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;lomLearningResourceType&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;lomOtherPlatformRequirements&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;lomSemanticDensity&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;lomStructure&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;lomTechRequirement&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;lomTypicalAgeRange&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>&lt;lomTypicalLearningTime&gt;</td>
<td>&lt;data&gt;</td>
<td>yes</td>
<td>N/A</td>
<td>block</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

### DITAVAL elements

The DITAVAL elements are not specialized, and are not rendered on their own, so related columns are dropped from this table. There are no translatable attributes in the DITAVAL element set.

As of DITA 1.2, the only element that directly contains text for translation is `<alt-text>`.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Block/Inline (translation)</th>
<th>Translatable content?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;alt-text&gt;</code></td>
<td>block</td>
<td>yes</td>
</tr>
<tr>
<td><code>&lt;endflag&gt;</code></td>
<td>block</td>
<td>yes (inside nested elements)</td>
</tr>
<tr>
<td><code>&lt;prop&gt;</code></td>
<td>block</td>
<td>yes (inside nested elements)</td>
</tr>
<tr>
<td><code>&lt;revprop&gt;</code></td>
<td>block</td>
<td>yes (inside nested elements)</td>
</tr>
<tr>
<td><code>&lt;startflag&gt;</code></td>
<td>block</td>
<td>yes (inside nested elements)</td>
</tr>
<tr>
<td><code>&lt;style-conflict&gt;</code></td>
<td>block</td>
<td>N/A (empty element)</td>
</tr>
<tr>
<td><code>&lt;val&gt;</code></td>
<td>block</td>
<td>yes (inside nested elements)</td>
</tr>
</tbody>
</table>
B.7 Implementation reference for DITA modules and document-type shells

(Non-normative) The DITA standard defines a set of base types and vocabulary modules and a set of specialized map types, topic types, and domains.

A significant feature of the DITA implementation is that it places more importance on the modules than on the actual document-type shell. All element and attribute-type declarations are made in modules, which are then integrated into a document type using a document-type shell. Implementers are free to create new document-type shells that introduce, constrain, or remove modules; the new shells also can control whether -- and what kind of -- topics can nest.

B.7.1 Domain modules

The DITA standard provides domain modules that you can integrate into your document-type shells.

The tables in this topic use the following variables:

- `grammar` specifies the type of XML grammar, for example, "dtd", "xsd", "rng", or "rnc".
- `.ext` is the file extension, for example, "ent", "mod", "xsd", "rng", or "rnc".

Comment by Kristen Eberlein, 6 March 2014
I have added the new 1.3 domains to the following table. I have not added the file names, as I cannot verify them against packages yet. Accordingly, the `<filepath>` elements contain "X".

Table 20: Domain modules used in DITA topics

<table>
<thead>
<tr>
<th>Domain</th>
<th>File names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbreviation</td>
<td><code>technicalContent/grammar/abbreviateDomain.ext</code></td>
<td>Provides an element (<code>&lt;abbreviated-form&gt;</code>) that represents a reference to a term that can be displayed in both long and short form</td>
</tr>
<tr>
<td>Equation</td>
<td><code>technicalContent/grammar/X.ext</code></td>
<td>Provides elements for representing equations</td>
</tr>
<tr>
<td>Hazard statement</td>
<td><code>base/grammar/hazardstatementDomain.ext</code></td>
<td>Provides elements for constructing product safety-information labels</td>
</tr>
<tr>
<td>Highlighting</td>
<td><code>base/grammar/highlightDomain.ext</code></td>
<td>Provides elements for highlighting text</td>
</tr>
<tr>
<td>Indexing</td>
<td><code>base/grammar/indexingDomain.ext</code></td>
<td>Provides elements for use with indexing</td>
</tr>
<tr>
<td>Learning</td>
<td><code>learning/grammar/learningDomain.ext</code></td>
<td>Provides elements for X</td>
</tr>
<tr>
<td>Learning interaction base</td>
<td><code>learning/grammar/learningInteractionBaseDomain.ext</code></td>
<td>Provides elements to describe learning interactions such as questions and responses</td>
</tr>
<tr>
<td>Learning metadata</td>
<td><code>learning/grammar/learningMetadataDomain.ext</code></td>
<td>Provides elements to describe the characteristics of learning content</td>
</tr>
<tr>
<td>Markup</td>
<td><code>technicalContent/grammar/X.ext</code></td>
<td>Provides an element that can be used a base for specializing elements to represent markup elements</td>
</tr>
<tr>
<td>Domain</td>
<td>File names</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MathML</td>
<td>technicalContent/grammar/X.ext</td>
<td>Provides elements for including or referencing MathML markup</td>
</tr>
<tr>
<td>Programming</td>
<td>technicalContent/grammar/programmingDomain.ext</td>
<td>Provides elements for documenting programming concepts and syntax</td>
</tr>
<tr>
<td>Release management</td>
<td>technicalContent/grammar/X.ext</td>
<td>Provides elements for human-authored information about the changes that have been made to a DITA topic</td>
</tr>
<tr>
<td>Software</td>
<td>technicalContent/grammar/softwareDomain.ext</td>
<td>Provides elements for documenting software applications</td>
</tr>
<tr>
<td>SVG</td>
<td>technicalContent/grammar/X.ext</td>
<td>Provides elements for including or referencing SVG markup</td>
</tr>
<tr>
<td>Task requirements</td>
<td>technicalContent/grammar/taskreqDomain.ext</td>
<td>Provides elements for use in tasks that involve machines or other pieces of hardware</td>
</tr>
<tr>
<td>User interface</td>
<td>technicalContent/grammar/uiDomain.ext</td>
<td>Provides elements for documenting user interfaces</td>
</tr>
<tr>
<td>Utilities</td>
<td>base/grammar/utilitiesDomain.ext</td>
<td>Provides elements for common features of a language that might not be semantic, such as image maps</td>
</tr>
<tr>
<td>XML mention</td>
<td>technicalContent/grammar/X.ext</td>
<td>Provides elements for describing XML applications and document types</td>
</tr>
</tbody>
</table>

Table 21: Domain modules used in DITA maps

<table>
<thead>
<tr>
<th>Domains</th>
<th>Description</th>
<th>File names</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification</td>
<td>Provides elements for associating topic references with subjects</td>
<td>subjectScheme/grammar/classifyDomain.ext</td>
</tr>
<tr>
<td>Delayed conref</td>
<td>Provides elements for use in environments when conref resolution is delayed or occurs at run time</td>
<td>base/grammar/delayResolutionDomain.ext</td>
</tr>
<tr>
<td>resolution</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Glossref</td>
<td>Provides convenience elements that are used to reference glossary topics or to reference multiple topics in a single collection</td>
<td>technicalContent/grammar/glossrefDomain/ext</td>
</tr>
<tr>
<td>Learning map</td>
<td>Provides elements to group topics as learning objects</td>
<td>learning/grammar/learningMapDomain.ext</td>
</tr>
<tr>
<td>Map group</td>
<td>Provides elements for use in addressing, defining, or grouping topic references</td>
<td>base/grammar/mapGroup.ext</td>
</tr>
<tr>
<td>xNAL</td>
<td>Provides elements that represent a subset of the Extensible Name and Address Standard</td>
<td>base/grammar/xnalDomain.ext</td>
</tr>
</tbody>
</table>
B.7.2 XML schema organization

(Non-normative) The OASIS DITA document types are implemented with a set of schema modules. Some of these modules are used by every DITA schema document, others are only used by topics or by maps, and some are only used in specific specializations.

XML schemas versus modules

Description of DITA modules

The tables below describe the modules that are defined by the DITA standard.

Table 22: Description of common modules

<table>
<thead>
<tr>
<th>Common module files</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common elements (commonElements)</td>
<td>Defines all content elements that may appear in both maps and topics.</td>
</tr>
<tr>
<td>Metadata elements (metaDecl)</td>
<td>Defines meta elements that may appear in both maps and topics</td>
</tr>
<tr>
<td>Table elements (tblDecl)</td>
<td>Defines the complex tables used within DITA, based on the OASIS Exchange Table model.</td>
</tr>
<tr>
<td>DITA Architecture attribute (ditaarch.xsd)</td>
<td>XML Schema only - Defines the attribute that defines DITA's architectural version</td>
</tr>
<tr>
<td>XML namespace attributes (xml.xsd)</td>
<td>XML Schema only - Defines the attributes with the XML namespace</td>
</tr>
</tbody>
</table>

Table 23: Description of domain modules

<table>
<thead>
<tr>
<th>Domains</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexing domain</td>
<td>The indexing domain provides several new elements for use with indexing. The new elements allow authors to define &quot;See&quot; and &quot;See also&quot; references, and to override the default sort order for a term.</td>
</tr>
<tr>
<td>Highlight domain</td>
<td>The highlighting elements are used to highlight text with styles (such as bold, italic, and monospace). Never use these elements when a semantically specific element is available. These elements are not intended for use by specializers, and are intended solely for use by authors when no semantically appropriate element is available and a formatting effect is required.</td>
</tr>
<tr>
<td>Programming domain</td>
<td>The programming domain elements are used to define the syntax and to give examples of programming languages.</td>
</tr>
<tr>
<td>Software domain</td>
<td>The software domain elements are used to describe the operation of a software program.</td>
</tr>
<tr>
<td>UI domain</td>
<td>The user interface domain elements are used to describe the user interface of a software program.</td>
</tr>
<tr>
<td>Utilities domain</td>
<td>The utilities domain elements represent common features of a language that may not necessarily be semantic, such as image maps.</td>
</tr>
<tr>
<td>Map Group domain</td>
<td>The mapgroup domain elements define, group, or reference content.</td>
</tr>
</tbody>
</table>
### Domains

<table>
<thead>
<tr>
<th>Domains</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>xNAL domain</td>
<td>The xNAL domain elements represent a subset of the Extensible Name and Address Standard. The domain can be included in any topic type or map, although the implementations provided by OASIS only include it in the bookmap specialization. It is used to encode information about the author or authors of DITA information.</td>
</tr>
<tr>
<td>Hazard Statement domain</td>
<td>The hazard statement domain elements represent labeling for product safety hazards that readers need to be aware of. The domain can be included in any topic type or map. Its elements are used to inform readers about potential hazards, consequences, and avoidance strategies</td>
</tr>
<tr>
<td>Machine Industry domain</td>
<td>The machine-industry task domain contains elements for use describing tasks that involve machines or other pieces of hardware.</td>
</tr>
<tr>
<td>Delay Resolution domain</td>
<td>The delayed conref resolution domain provides several elements for use when using DITA in situations that enable delayed or run time resolution of conref. The elements allow users to resolve some conref values statically, while delaying others for later resolution.</td>
</tr>
<tr>
<td>Abbreviate domain</td>
<td>The abbreviate domain element is used to represent a reference to a term that may appear in an abbreviated form (often an acronym).</td>
</tr>
<tr>
<td>Glossref domain</td>
<td>The glossref domain elements are convenience elements that are used to reference glossary topics or to reference multiple topics in a single collection.</td>
</tr>
<tr>
<td>Classify domain</td>
<td>The classification domain elements are used to identify subjects covered by the content.</td>
</tr>
<tr>
<td>Learning domain</td>
<td>The learning domain elements represent the base elements that are used to create learning and training content.</td>
</tr>
<tr>
<td>Learning Interaction Base domain</td>
<td>The learning interaction base domain defines an &quot;abstract&quot; base type for all learning assessments.</td>
</tr>
<tr>
<td>Learning Map domain</td>
<td>The learning and training map domain organizes groups of topics as learning objects</td>
</tr>
<tr>
<td>Learning Metadata domain</td>
<td>The learning and training metadata domain describes specific characteristics of the learning content.</td>
</tr>
</tbody>
</table>

**Table 24: Description of topic and its specialization modules**

<table>
<thead>
<tr>
<th>Topic types</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic</td>
<td>Topic modules</td>
</tr>
<tr>
<td>UI domain</td>
<td></td>
</tr>
<tr>
<td>Software domain</td>
<td></td>
</tr>
<tr>
<td>Highlight domain</td>
<td></td>
</tr>
<tr>
<td>Programming domain</td>
<td></td>
</tr>
<tr>
<td>Utilities domain</td>
<td></td>
</tr>
<tr>
<td>Indexing domain</td>
<td></td>
</tr>
<tr>
<td>Hazard Statement domain</td>
<td></td>
</tr>
<tr>
<td>Abbreviate domain</td>
<td></td>
</tr>
<tr>
<td>Base Topic</td>
<td>Topic modules</td>
</tr>
<tr>
<td>Highlight domain</td>
<td></td>
</tr>
<tr>
<td>Topic types</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
|             | Utilities domain  
|             | Indexing domain  
|             | Hazard Statement domain  |
| DITABase    | Concept modules  
|             | Topic modules  
|             | Reference modules  
|             | Task modules  
|             | General Task modules  
|             | Glossary Entry modules  
|             | GlossGroup modules  |
|             | UI domain  
|             | Software domain  
|             | Highlight domain  
|             | Programming domain  
|             | Utilities domain  
|             | Indexing domain  
|             | Hazard Statement domain  
|             | Abbreviate domain  |
| Concept     | Concept modules  
|             | Topic modules  |
|             | UI domain  
|             | Software domain  
|             | Highlight domain  
|             | Programming domain  
|             | Utilities domain  
|             | Indexing domain  
|             | Hazard Statement domain  
|             | Abbreviate domain  |
| Glossary    | Glossary Entry modules  
|             | Concept modules  
|             | Topic modules  |
|             | UI domain  
|             | Software domain  
|             | Highlight domain  
|             | Programming domain  
|             | Utilities domain  
|             | Indexing domain  
|             | Hazard Statement domain  
|             | Abbreviate domain  |
| GlossGroup  | GlossGroup modules  
|             | Concept modules  
|             | Topic modules  |
|             | UI domain  
<p>|             | Software domain  |</p>
<table>
<thead>
<tr>
<th>Topic types</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highlight domain</td>
<td>Programming domain</td>
</tr>
<tr>
<td>Utilities domain</td>
<td>Indexing domain</td>
</tr>
<tr>
<td>Hazard Statement domain</td>
<td>Abbreviate domain</td>
</tr>
<tr>
<td>Reference</td>
<td>Reference modules</td>
</tr>
<tr>
<td>Topic modules</td>
<td>UI domain</td>
</tr>
<tr>
<td>Software domain</td>
<td>Highlight domain</td>
</tr>
<tr>
<td>Programming domain</td>
<td>Utilities domain</td>
</tr>
<tr>
<td>Indexing domain</td>
<td>Hazard Statement domain</td>
</tr>
<tr>
<td>Abbreviate domain</td>
<td>Task</td>
</tr>
<tr>
<td>Topic modules</td>
<td>UI domain</td>
</tr>
<tr>
<td>Strict Taskbody module</td>
<td>Software domain</td>
</tr>
<tr>
<td>General Task</td>
<td>Highlight domain</td>
</tr>
<tr>
<td>Task modules</td>
<td>Programming domain</td>
</tr>
<tr>
<td>Topic modules</td>
<td>Utilities domain</td>
</tr>
<tr>
<td>Machinery Task</td>
<td>Indexing domain</td>
</tr>
<tr>
<td>Task modules</td>
<td>Hazard Statement domain</td>
</tr>
<tr>
<td>Topic modules</td>
<td>Abbreviate domain</td>
</tr>
<tr>
<td>Topic types</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| Learning Assessment | Abbreviate domain  
<p>|                   | Machine Industry domain |
|                   | Learning Assessment modules |
|                   | Topic modules |
|                   | Highlight domain |
|                   | Utilities domain |
|                   | Indexing domain |
|                   | Learning domain |
|                   | Learning Interaction Base domain |
|                   | Learning metadata domain |
| Learning Content  | Learning Content modules |
|                   | Learning Assessment modules |
|                   | Learning Summary modules |
|                   | Concept modules |
|                   | Topic modules |
|                   | Reference modules |
|                   | Task modules |
|                   | Highlight domain |
|                   | Utilities domain |
|                   | Indexing domain |
|                   | Learning domain |
|                   | Learning Interaction Base domain |
|                   | Learning metadata domain |
| Learning Overview | Learning Overview modules |
|                   | Topic modules |
|                   | Highlight domain |
|                   | Utilities domain |
|                   | Indexing domain |
|                   | Learning domain |
|                   | Learning Interaction Base domain |
|                   | Learning metadata domain |
| Learning Plan     | Learning Plan modules |
|                   | Topic modules |
|                   | Highlight domain |
|                   | Utilities domain |
|                   | Indexing domain |
|                   | Learning domain |
|                   | Learning Interaction Base domain |
|                   | Learning metadata domain |
| Learning Summary  | Learning Summary modules |
|                   | Topic modules |</p>
<table>
<thead>
<tr>
<th>Topic types</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highlight domain</td>
<td></td>
</tr>
<tr>
<td>Utilities domain</td>
<td></td>
</tr>
<tr>
<td>Indexing domain</td>
<td></td>
</tr>
<tr>
<td>Learning domain</td>
<td></td>
</tr>
<tr>
<td>Learning Interaction Base domain</td>
<td></td>
</tr>
<tr>
<td>Learning metadata domain</td>
<td></td>
</tr>
</tbody>
</table>

**Table 25: Description of map and its specialization modules**

<table>
<thead>
<tr>
<th>Map types</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>Map modules</td>
</tr>
<tr>
<td></td>
<td>Map Group domain</td>
</tr>
<tr>
<td></td>
<td>Indexing domain</td>
</tr>
<tr>
<td></td>
<td>Delay Resolution domain</td>
</tr>
<tr>
<td></td>
<td>Glossref domain</td>
</tr>
<tr>
<td></td>
<td>UI domain</td>
</tr>
<tr>
<td></td>
<td>Software domain</td>
</tr>
<tr>
<td></td>
<td>Highlight domain</td>
</tr>
<tr>
<td></td>
<td>Programming domain</td>
</tr>
<tr>
<td></td>
<td>Utilities domain</td>
</tr>
<tr>
<td></td>
<td>Indexing domain</td>
</tr>
<tr>
<td></td>
<td>Hazard Statement domain</td>
</tr>
<tr>
<td></td>
<td>Abbreviate domain</td>
</tr>
<tr>
<td>Base Map</td>
<td>Map modules</td>
</tr>
<tr>
<td></td>
<td>Map Group domain</td>
</tr>
<tr>
<td></td>
<td>Indexing domain</td>
</tr>
<tr>
<td></td>
<td>Delay Resolution domain</td>
</tr>
<tr>
<td></td>
<td>Highlight domain</td>
</tr>
<tr>
<td></td>
<td>Utilities domain</td>
</tr>
<tr>
<td></td>
<td>Hazard Statement domain</td>
</tr>
<tr>
<td>Bookmap</td>
<td>Bookmap modules</td>
</tr>
<tr>
<td></td>
<td>Map modules</td>
</tr>
<tr>
<td></td>
<td>Map Group domain</td>
</tr>
<tr>
<td></td>
<td>Indexing domain</td>
</tr>
<tr>
<td></td>
<td>Delay Resolution domain</td>
</tr>
<tr>
<td></td>
<td>UI domain</td>
</tr>
<tr>
<td></td>
<td>Software domain</td>
</tr>
<tr>
<td></td>
<td>Highlight domain</td>
</tr>
<tr>
<td></td>
<td>Programming domain</td>
</tr>
<tr>
<td></td>
<td>Utilities domain</td>
</tr>
<tr>
<td></td>
<td>Indexing domain</td>
</tr>
<tr>
<td></td>
<td>Hazard Statement domain</td>
</tr>
<tr>
<td></td>
<td>Abbreviate domain</td>
</tr>
<tr>
<td>Map types</td>
<td>Purpose</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Classification Map</td>
<td>Map modules</td>
</tr>
<tr>
<td></td>
<td>Map Group domain</td>
</tr>
<tr>
<td></td>
<td>Indexing domain</td>
</tr>
<tr>
<td></td>
<td>Delay Resolution domain</td>
</tr>
<tr>
<td></td>
<td>Classification domain</td>
</tr>
<tr>
<td></td>
<td>Highlight domain</td>
</tr>
<tr>
<td></td>
<td>Utilities domain</td>
</tr>
<tr>
<td></td>
<td>Hazard Statement domain</td>
</tr>
<tr>
<td>Subject Scheme</td>
<td>Subject Scheme modules</td>
</tr>
<tr>
<td></td>
<td>Map modules</td>
</tr>
<tr>
<td></td>
<td>Map Group domain</td>
</tr>
<tr>
<td></td>
<td>Highlight domain</td>
</tr>
<tr>
<td></td>
<td>Utilities domain</td>
</tr>
<tr>
<td></td>
<td>Hazard Statement domain</td>
</tr>
<tr>
<td>Learning Map</td>
<td>Map modules</td>
</tr>
<tr>
<td></td>
<td>Map Group domain</td>
</tr>
<tr>
<td></td>
<td>Indexing domain</td>
</tr>
<tr>
<td></td>
<td>Delay Resolution domain</td>
</tr>
<tr>
<td></td>
<td>Learning Metadata domain</td>
</tr>
<tr>
<td></td>
<td>Learning Map domain</td>
</tr>
<tr>
<td></td>
<td>Highlight domain</td>
</tr>
<tr>
<td></td>
<td>Utilities domain</td>
</tr>
<tr>
<td></td>
<td>Hazard Statement domain</td>
</tr>
<tr>
<td>Learning Bookmap</td>
<td>Map modules</td>
</tr>
<tr>
<td></td>
<td>Bookmap modules</td>
</tr>
<tr>
<td></td>
<td>Map Group domain</td>
</tr>
<tr>
<td></td>
<td>Indexing domain</td>
</tr>
<tr>
<td></td>
<td>Delay Resolution domain</td>
</tr>
<tr>
<td></td>
<td>xNAL Domain</td>
</tr>
<tr>
<td></td>
<td>Learning Metadata domain</td>
</tr>
<tr>
<td></td>
<td>Learning Map domain</td>
</tr>
<tr>
<td></td>
<td>Highlight domain</td>
</tr>
<tr>
<td></td>
<td>Utilities domain</td>
</tr>
<tr>
<td></td>
<td>Hazard Statement domain</td>
</tr>
</tbody>
</table>

Table 26: Description of constraint modules

<table>
<thead>
<tr>
<th>Constraint types</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strict Taskbody</td>
<td>The strict task body has a constrained structure.</td>
</tr>
<tr>
<td>Constraint types</td>
<td>Purpose</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------</td>
</tr>
<tr>
<td>Machinery</td>
<td>The machinery task body makes use of elements prereqs and closereqs instead of prereq and postreq</td>
</tr>
<tr>
<td>Taskbody</td>
<td>and postreq</td>
</tr>
</tbody>
</table>

**B.8 Processing component modularization and naming conventions**

Files that implement processing associated with specific vocabulary or constraint modules should generally be organized by module, such that the implementation components correspond to the vocabulary modules that they implement and are, to the degree the implementation technology provides for it, similarly modular.

For example, the DITA Open Toolkit provides a general plugin mechanism that allows implementation components to be dynamically added to and removed from a Toolkit instance. General practice is to define Toolkit plugins that correspond to modules.

For the specific case of cascading style sheets and XSLTs transformations, two key DITA implementation technologies, the suggested filenaming conventions are:

**CSS override files (recommended convention for tool providers)**
- `typename.css`
- `customization-purpose.css`

**XSLT override files (recommended convention for tool providers)**
- `typename.xsl`
- `customization-purpose.xsl`

**B.9 Processing interoperability considerations**

(Non-normative) The DITA specification does not require processors to perform filtering, content reference resolution, key space construction, and other processing related to base DITA semantics in any particular order. This means that different conforming DITA processors may produce different results for the same initial data set and filtering conditions. DITA users and DITA implementers need to be aware of these potential differences in behavior when DITA content will be processed by different processors.

For conditional elements, an element is "applicable" if it is filtered in by active filtering specification and "inapplicable" if it is filtered out.

In general, in any situation in which two elements interact during processing, applying filtering before or after the processing is done will result in different results when either or both of the elements is conditional.

**Filtering and content reference resolution**

When two elements are merged as result of a content reference, the attributes of the two elements are combined. By default, the attributes of the referencing element take precedence over the referenced element. However, any attribute may specify the value "-dita-use-conref-target", which causes the referenced element's attribute to take precedence. This means that the effective value of filtering attributes may reflect either the referencing element or the referenced element depending on how each attribute is configured on the referencing element. This in turn means that, in certain cases, filtering before resolving content references will produce a materially different result than when filtering is applied after resolving content references.

In two cases, the order in which filtering is applied results in either an element being in the effective result or an element not being in the effective result. There is a third case in which there will be either an empty element (and unresolvable content reference) or no element.

In the case where a referenced element is not applicable and the referencing element is explicitly applicable for the same condition (that is, both elements specify values for the same classification domain and the referencing element is applicable), if content references are resolved before filtering, the content reference is resolved and
the effective value of the referencing element reflects the referenced element. If content referencing is resolved after filtering, the referenced element is filtered out and the content reference cannot be resolved, typically generating an error.

If the referencing element is not conditional and the referenced element is inapplicable, filtering applied before content reference resolution results in an unresolvable content reference. If filtering is applied after content resolution, the explicit condition on the referenced element becomes the effective value for that condition following content resolution and the result is then filtered out. The difference in these two cases is that in the first case the content reference cannot be resolved, resulting in a processing error and a potentially non-sensical element if the referencing element has required subelements (for example, a content reference from a topic to another topic, where the referencing topic must have a title subelement), but in the second case the element is filtered completely out.

**Filtering and key space resolution**

DITA allows processors to apply filtering before or after key space construction. Key space construction is the process of determining which key definitions are effective within a given map tree. Given two definitions of the same key with different applicabilities, the effective key space will be different if filtering is applied before key space construction than if it is applied after. In particular, if filtering is applied after key space construction, the key space is invariant, but if filtering is applied before key space construction, the effective key space will differ depending on the active conditions.

As an implementation detail for key-space-constructing processors, if filtering is applied before constructing the key space, then the set of effective key definitions is simply the first definition of each unique key name within the map tree. However, if filtering is applied after key space construction, and in particular, if a processor needs to allow dynamic resolution of keys based on different filtering specifications applied to the same constructed key space, then the set of effective key definitions is the first definition of each pair of unique key name and unique selection specification set. This second form of constructed key space would be needed by processors such as editors and content management systems that need to quickly provide different filtering-specific key bindings without reconstructing the entire key space for each new set of filtering conditions.

For example, given a map that contains two definitions for the key "topic-01", one with an @audience value of "expert" and one with an @audience value of "novice", a filter-first processor would only have at most one effective key definition for the key name "topic-01", whichever of the two definitions was filtered in by the active filter specification and was the first definition encountered (if both happen to be filtered in). In a processor that supports dynamic key definition filtering, there would be two effective definitions for the key name "topic-01", one for @audience of "expert" and one for @audience of "novice". The processor would also need to maintain knowledge of the definition order of the two key definitions in order to correctly handle the case when both "expert" and "novice" are applicable for a given key access request (in which case, whichever of the two definitions was first in the map tree would be used as the effective value of the key).

**Link resolution**

If a cross reference, link, or other linking element is resolved to its target before filtering and the target is subsequently filtered out, the link would be to a non-existent target but might reflect properties of the target (e.g., a cross reference link text might reflect the target's title). If the link is resolved after filtering is applied and the target is filtered out, the link is to a non-existent target, which will result in a different link text. The rendition effect for the navigation link will be the same: the link cannot be navigated because the target doesn't exist in the rendered result.

**Map metadata propagation**

Filtering applied before propagation can result in different effective values than if it is applied after. In particular, elements filtered before out will never contribute to propagation.

**Topcref resolution**

Resolution of topicrefs before filtering can result in use of topic-provided navigation titles or metadata that would not be used if the target topic was filtered out before resolution. In both cases, the topicref as rendered would be to a missing topic.
Chunking

A topicref subsequently filtered out that generates chunks would create chunks in the output if chunk processing is done before filtering but since the topicref would then be filtered out, the chunks would not be referenced.

Copy-to processing

If copy-to processing is done before filtering, two topicrefs, only one of which is applicable, could specify the same copy-to target, leading to a conflict and a potential ambiguity about which governs. If the topicrefs are filtered before copy-to processing, the conflict does not occur.

B.10 Specialization design, customization, and the limits of specialization

(Non-normative) DITA specialization imposes certain restrictions. An inherent challenge in designing DITA vocabulary modules and document types is understanding how to satisfy markup requirements within those restrictions and, when the requirements cannot be met by a design that fully conforms to the DITA architecture, how to create customized document types that diverge from the DITA standard as little as possible.

DITA imposes the following structural restrictions:

- All topics must have titles.
- Topic body content must be contained within a body element.
- Section elements cannot nest.
- Metadata specific to an element type must be represented using elements, not attributes.

When markup requirements cannot be met within the DITA architecture, there still might be an interest in using DITA features and technology, or a business need for interoperability with conforming DITA documents and processors. In this case, the solution is to create customized document types. Customized document types are document types that do not conform to the DITA standard. To reduce the cost of producing conforming documents from non-conforming documents, custom document types should minimize the extent to which they diverge from the DITA standard.

Typical reasons for considering custom document types include the following:

- Optimizing markup for authoring by excluding unwanted element types and attributes that are included in the OASIS-provided vocabulary modules
- Supporting legacy markup structures that are not consistent with DITA structural rules, for example, footnotes within titles or nested sections within topic bodies
- Defining different forms of existing structures, such as lists, where the DITA-defined structures are too constrained
- Providing attributes required by specific processors, such as CMS-defined attributes for maintaining management metadata
- Embedding tool-imposed markup in places that do not allow the <foreign> or <unknown> elements

In many of these cases, it is possible to define document types that conform to the DITA standard. Explore this fully before developing customized document types.

Optimizing document types for authoring

The base DITA element types provide relaxed content models in order to avoid imposing unnecessary constraints on specializers. However, these open content models are often inappropriately open for authoring purposes, where authors are better served by more restrictive content models. To address this requirement, DITA provides two configuration mechanisms for defining and configuring document types: document type shells and constraint modules.

Document type shells pull together sets of vocabulary and constraint modules in order to define a working document type. The OASIS-provided document-type shells typically include all available vocabulary modules, which often is not what is needed or desired. When the requirement is to eliminate unneeded domain modules or
topic types, simply define new document-type shells that omit unneeded domains or structural types and include
any locally-defined or third-party modules that users might need. See Configuration (Document type shells) (see
page 118) for details.

The DITA constraint mechanism, introduced with DITA 1.2, makes it possible to configure the content models of
individual elements from document type shells. This means you can optimize content models to meet authoring
requirements without directly modifying the base vocabulary modules, as long as you only need to add
constraints or eliminate optional elements.

Most authoring optimization requirements can be met with a combination of document type shells and constraint
modules, when the markup requirements are satisfied by available vocabulary modules. When markup
requirements are not satisfied by existing vocabulary modules, you can specialize new vocabulary modules and
integrate them into your document-type shells.

If the markup requirements include content models that are less constrained than those defined by DITA, you
have no choice but to define custom document types.

Where interchange and interoperation with other DITA systems and information sets are required, you must first
transform non-conforming documents into conforming documents. See Map from a customized document type to
DITA during preprocessing (see page 733).

Specialization design considerations

Requirements for new markup often appear to be inconsistent or incompatible with DITA architectural rules or
existing markup, especially when mapping existing non-DITA markup practice to DITA, where the existing
markup may have used structures that cannot be directly expressed in DITA. For example, you might need
markup for a specialized form of list where the details are not consistent with the base model for DITA lists.

In this case you have two alternatives, one that conforms to DITA and one that does not.

• Specialize from more generic base elements or attributes.
• Define non-conforming structures and map them to conforming DITA structures as necessary for processing
  by DITA-aware processors or for interchange as conforming DITA documents.

Specializing from more generic base elements, such as defining a list using specializations of <ph> or <p>, while
technically conforming, might still impede interchange of such documents because generic DITA processors will
have no way of knowing that what they see as a sequence of phrases or paragraphs or whatever is really a list
and should be rendered in a listy way. However, your documents will be reliably interchangeable with conforming
DITA systems.

Defining non-conforming markup structures means that the documents that use those structures cannot be
conforming DITA documents as authored and therefore cannot be reliably processed by generic DITA-aware
processors or interchanged with other DITA systems. However, as long as the documents can be transformed
into conforming DITA documents without undue effort interchange and interoperation requirements can be
satisfied as needed. This approach will often be needed when using content management systems to manage
what is nominally DITA content where the content management system imposes requirements onto the content
for whatever reason, such as to add its own markup for management metadata or because of implementation
limitations.

In addition, non-conforming document types can use the basic specialization mechanism used by the DITA
document types, with the same re-use and interoperation benefits, only restricted to the specific domain within
which the new document types apply. Such document types are not conforming DITA document types but may
be quite useful because of the general benefits of specialization as an enabling technology.

Note that even if one uses the DITA-defined types as a starting point, any change to those base types not
accomplished through specialization or the constraint feature defines a completely new document type that has
no normative relationship to the DITA document types, and cannot be considered in any way to be a conforming
DITA application. In particular, the use of DITA specialization from non-DITA base types does not produce DITA-
conforming vocabularies.
Specialize from generic elements or attributes

Most DITA element types have relaxed content models that are specifically designed to allow a wide set of options when specializing from them. However, some DITA element types do impose constraints that might not be acceptable or appropriate for a specific markup application. In this case, consider specializing from a more generic base element or attribute.

Generic elements are available in DITA at every level of detail, from whole topics down to individual keywords, and the generic @base attribute is available for attribute domain specialization.

For example, if you want to create a new kind of list but cannot usefully do so specializing from `<ul>`, `<ol>`, `<sl>`, or `<dl>`, you can create a new set of list elements by specializing nested `<ph>` elements. This new list structure will require specialized processing to generate appropriate output styling, because it is not semantically tied to the other lists by ancestry. Nevertheless, it will remain a valid DITA specialization, with the standard support for generalization, content referencing, conditional processing, and more.

The following base elements in `<topic>` are generic enough to support almost any structurally-valid DITA specialization:

- `<topic>`
  - Any content unit that has a title and associated content
- `<section>`
  - Any non-nesting division of content within a topic, titled or not
- `<p>`
  - Any non-titled block of content below the section level
- `<fig>`
  - Any titled block of content below the section level
- `<ul>`, `<ol>`, `<dl>`, `<simpletable>`
  - Any structured block of content that consists of listed items in one or more columns
- `<ph>`
  - Any division of content below the paragraph level
- `<text>`
  - Text within a phrase
- `<keyword>`
  - Any non-nesting division of content below the paragraph level
- `<data>`
  - Any content that acts as metadata rather than core topic or map content
- `<foreign>`
  - Any content that already has a non-DITA markup standard, but still needs to be authored as part of the DITA document. Processors should attempt to render this element, if at all possible.
- `<unknown>`
  - Any non-standard markup that does not fit the DITA model, but needs to be managed as part of a DITA document. Processors should not attempt to render this element.
- `<bodydiv>`
  - A generic, untitled, nestable container for content within topic bodies
- `<sectiondiv>`
  - A generic, untitled, nestable container for content within sections
- `<div>`
  - A generic, untitled, nestable container for content within topic bodies or sections
The following attributes in topic are suitable for domain specialization to provide new attributes that are required throughout a document type:

@props
- Any new conditional processing attribute

@base
- Any new attribute that is universally available, has a simple syntax (space-delimited alphanumeric values), and does not already have a semantic equivalent

Whenever possible, specialize from the element or attribute that is the closest semantic match.

**Map from a customized document type to DITA during preprocessing**

Specializations of elements or attributes might not be sufficient for some authoring requirements. In particular, specialization cannot split or rename attributes, and an element cannot be renamed without also specializing its containing element. In such cases, it might be possible to transform a customized document type to a standard-DITA document type during the publishing process.

For example, if an authoring group requires the `<p>` element to be spelled out as `<paragraph>`, the document type could be customized to change `<p>` to `<paragraph>` for authoring purposes. Such documents then could be preprocessed to rename `<paragraph>` back to `<p>` before then are fed into a standard publishing process.

The structural modules should not be edited to create a customized document type. Instead, a customized document-type shell can provide new definitions of DITA entities, including entities for attributes and content models. The new definitions override the entity definitions in the module files before they are imported.

Customized document types do not conform to the DITA standard. Preprocessing can ensure compatibility with existing publishing processes, but it does not ensure compatibility with DITA-supporting authoring tools or content management systems. However, when an implementation is being heavily customized, a customized document type can help isolate and control the consequences of non-standard design.
-dita-use-conref-target attribute value 666

A
abbreviated-form 457
abbreviation list 462
abbrevlist 462
abstract
  Bookmap elements
    abstract 465
addressdetails
  xNAl elements
    addressdetails 537
administrativearea 537
align attribute 657
alt (alternate text) 258
amendments 462
anchor 305
anchorid 346
anchorkey 350
anchorref 313
anchorref attribute 662
apiname 495
appendices
  Bookmap elements
    appendices 463
appendix
  Bookmap elements
    appendix 463
appid attribute (resourceid) 331
application programming interface name 495
appname attribute (resourceid) 331
approved 476
Architectural attribute group 654
area 360
attributedef 384
attributes
  appid (resourceid) 331
  appname (resourceid) 331
  id (resourceid) 331
Attributes
  align 657
  anchorref 662
  audience 652
  base 652
  char 658
  charoff 658
  chunk 657
  class 651
  collection-type 655
  colsep 658
  compact 662
  conaction 651
  conkeyref 651
  conref 651
  conrefend 651
  copy-to 663
  datatype 659
  dir 653
  DITAArchVersion 654
  domains 655

Attributes (continued)
  expanse 660
  expiry 659
  format 661
  frame 660
  golive 660
  href 661
  id (common) 651
  importance 652
  keycol 662
  keyref 661
  linking 656
  locktitle 656
  mapkeyref 662
  name 659
  navtitle 663
  otherprops 652
  outputclass 681
  platform 652
  print 657
  processing-role 655
  product 652
  props 651
  query 663
  refcols 663
  relcolwidth 662
  rev 653
  rowsep 658
  scale 661
  scope 661
  search 657
  specentry 663
  spectitle 663
  status 653
  toc 656
  translate 653
  type 661
  valign 659
  value 659
  xml:lang 653
  xml:space 662
  xmlns:ditaarch 654
  xtrc 654
  xtrf 654
  audience 321
  audience attribute 652
  author 322
  authorinformation 536

B
b 357
backmatter 464
base attribute 652
base sort phrase, definition 111
bibliolist 464
body 255
body elements
  alt 258
cite 258
dd 258
ddhd 259
body elements (continued)
  desc 259
div 260
dlentry 262
dlhead 263
dt 263
dthd 265
fig 266
dlentry 266
image 268
keyword 270
li (list item) 271
lines 271
lq 273
note 274
object 274
ol 278
p 278
param 279
ph 280
pre 280
q 281
sl (simple list) 282
sl (simple list item) 283
ul (unordered list) 285
xref (cross reference) 285
bookmeta 475
booktitle 467
booktitlealt 467
toc 474
revisionid 486
started 486
summary 487
tested 487
trademarklist 474
volume 488
year 488
bookchangehistory 477
bookevent 477
bookeventtype 478
bookid 478
booklibrary 465
booklist 466
booklists 466
bookmap 461
bookmeta 475
booktitle 467
booktitlealt 467
branch filtering
  examples 97, 366
brand 323

cascading
  map-to-map
    attributes 51
    exceptions 53
    metadata elements 53
  cascading, definition 48
  category 323
  cause 446
  change-completed 509
  change-historylist 509
  copyright © OASIS Open 2005, 2014. All Rights Reserved.
change-item 510
change-organization 510
change-person 511
change-request-id 511
change-request-reference 511
change-started 512
change-summary 513
chapter 467
char attribute 658
charoff attribute 658
chdesc (choice option description) 433
chdeschd (choice description alternate heading) 431
chhead (choicetable heading) 430
choice 434
choices 434
choicetable 430
choption (choice option) 433
choptionhd (choice option alternate heading) 431
chrow (choice table row) 432
chunk attribute 657
cite 258
class attribute 651
classification domain elements
  subjectCell 405
  subjectref 395
  topicapply 396
  topicCell 403
  topicsubject 397
  topicSubjectHeader 400
  topicSubjectRow 401
  topicSubjectTable 398
closereqs 521
cmd (step command) 428
cmdname (command name) 514
code phrase (codeph) 497
codeblock 496
codeph (code phrase) 497
coderef 496
collation 111
collection-type attribute 655
colophon 468
colsep attribute 658
colspec 292
command name (cmdname) 514
Common attributes group 661
Common map attribute group 655
Commonly referenced attribute descriptions
  Architectural attributes 654
  Common attributes 661
  Common map attributes 655
  Complex-table attributes 657
  Data element attributes group 659
  Date attributes 659
  Debug attributes 654
  Display attributes 660
  ID attributes 651
  Link relationships 661
  localization-atts 653
  Metadata attributes 651
  Simpletable attributes 662
Commonly referenced attribute descriptions (continued)
  Specialization attributes 663
  Topicrof element attributes group 663
  Universal attributes 650
compact attribute 662
completed 480
Complex-table attribute group 657
component 324
Composite topics 257
conaction attribute 651
conbody (concept body) 422
conbodydiv 422
correct 421
correct elements
  conbody 422
  concept 421
correct attribute 445
conditional processing
  subset of a map 366
conkeyref attribute 651
conref attribute 651
conrefend attribute 651, 670
consequence 353
contactnumber
  xNAL elements
    contactnumber 538
  contactnumbers
    xNAL elements
      contactNumbers 538
context 425
context-sensitive help 311
cords 360
copy-to attribute 663
copyrfirst 481
copyrholder 325
copyright 324
copyrlast 481
copyyear 325
country
  xNAL elements
    country 539
created 326
critdates (critical dates) 326
critical dates (critdates) 326
cross-reference (xref) 285

data 407
data about 409
datatype attribute 659
date attribute group 659
day 481
dd (description description) 258
ddhd (description descriptions heading) 259
debug attribute group 654
dedication 468
defaultSubject 385
definition description (dd) 258
definition list (dl) 261
definition list entry (dIentry) 262
frame attribute 660
frontmatter 470

G
generationidentifier
  xNAL elements
    generationidentifier 541
Gglossentry elements
  glossdef 449
  glossAbbreviation 449
  glossAcronym 449
  glossAlt 449
  glossAlternateFor 449
  glossarylist 470
  glossBody 451
  glossdef 449
  glossentry 447
  glossentry elements
    glossentry 447
      glossTerm 448
    glossgroup 460
    glossPartOfSpeech 451
    glossProperty 451
    glossref 458
    glossScopeNote 453
    glossStatus 453
    glossSurfaceForm 454
    glossSymbol 454
    glossSynonym 456
    glossTerm 448
    glossUsage 456
  golive attribute 660
  groupchoice (group choice) 501
  groupcomp (group composite) 501
  grouping 111
  grouping elements
    bodydiv 256
    div 260
    sectiondiv 282
  groupseq (group sequence) 500

H
hasInstance 378
hasKind 379
hasNarrower 379
hasPart 380
hasRelated 381
hazard statement domain
  consequence 353
  hazardstatement 352
  hazardsymbol 354
  howtoavoid 355
  messagepanel 356
  typeofhazard 356
  hazardstatement 352
  hazardsymbol 354
  highlighting domain
    b 357
    i 357
  highlighting domain (continued)
    line-through 359
    overline 359
    sub 358
    sup 357
    tt 358
    u 359
  honorific
    xNAL elements
      honorific 541
  howtoavoid 355
  href attribute 661, 663

I
i 357
id attribute
  resourceid 331
  id attribute (common) 651
  ID attribute group 651
  image 268
  imagemap 361
  importance attribute 652
  index-base 411
  index-see 340
  index-see-also 341
  index-sort-as 111, 342
  indexing domain
    index-see 340
    index-see-also 341
    index-sort-as 342
    indexterm 336
    indextermref 340
  indexlist 471
  indexterm 336
  indextermref 340
  info (step information) 428
  isbn 482
  itemgroup 411

K
keycol attribute 662
keydef 315
keyref attribute 661, 665
keys attribute 665
keyword 270
keywords 327
kwd (syntax diagram keyword) 505

L
lastname
  xNAL elements
    lastname 541
  lcAge 618
  lcAnswerContent 605
  lcAnswerContent2 569
  lcAnswerOption 605
  lcAnswerOption2 570
  lcAnswerOptionGroup 606
  lcAnswerOptionGroup2 570
learning and training elements (continued)

- lcIntro 629
- lcItem 612
- lcItem2 579
- lcJtaltem 629
- lcKnowledge 630
- lcLearnStrat 630
- lcLMS 631
- lcLom 595
- lcMatching 594
- lcMatching2 579
- lcMatchingHeader 612
- lcMatchingItem 613
- lcMatchingItem2 580
- lcMatchingItemFeedback 613
- lcMatchingItemFeedback2 581
- lcMatchingPair 614
- lcMatchingPair2 581
- lcMatchTable 614
- lcMatchTable2 581
- lcModDate 631
- lcMotivation 632
- lcMultipleSelect 592
- lcMultipleSelect2 582
- lcNeeds 632
- lcNeedsAnalysis 632
- lcNextSteps 634
- lcNoLMS 634
- lcObjective 634
- lcObjectives 635
- lcObjectivesGroup 635
- lcObjectivesStem 636
- lcOJT 636
- lcOpenAnswer 615
- lcOpenAnswer2 583
- lcOpenQuestion 590
- lcOpenQuestion2 583
- lcOrganizational 637
- lcOrgConstraints 637
- lcPlanAudience 638
- lcPlanDescrip 638
- lcPlanObjective 639
- lcPlanPrereqs 639
- lcPlanResources 640
- lcPlanSubject 640
- lcPlanTitle 640
- lcPlayers 641
- lcProcesses 642
- lcProject 642
- lcQuestion 615
- lcQuestion2 584
- lcQuestionBase 590
- lcQuestionBase2 584
- lcResolution 643
- lcResources 643
- lcReview 644
- lcSecurity 644
- lcSequence 616
- lcSequence2 584
- lcSequenceOption 616
- lcSequenceOption2 585
- lcSequenceOptions 586, 617
learning and training elements (continued)

lcSequencing 593
lcSequencing2 587
lcSingleSelect 591
lcSingleSelect2 587
lcSkills 645
lcSpecChars 645
lcSummary 646
lcTask 646
lcTaskItem 647
lcTechnical 647
lcTime 648
lcTrueFalse 591
lcTrueFalse2 588
lcValues 648
lcViewers 649
lw3c 649
lwWorkEnv 650
lwWorkEnvDescription 650
learningAssessment 552
learningAssessmentbody 553
learningBase 556
learningBasebody 557
learningContent 550
learningContentbody 551
learningContentComponentRef 567
learningContentRef 566
learningGroup 559
learningGroupMap 557
learningGroupMapRef 562
learningObject 561
learningObjectMap 558
learningObjectMapRef 562
learningOverview 549
learningOverviewbody 550
learningOverviewRef 565
learningPlan 553
learningPlanbody 556
learningPlanRef 563
learningPostAssessmentRef 568
learningPreAssessmentRef 564
learningSummary 551
learningSummarybody 552
learningSummaryRef 567
legacy conversion elements
required-cleanup 413
li (list item) 271
line-through 359
lines 271
link 297
Link relationship attribute group 661
linkinfo 301
linking attribute 656
linklist 298
linkpool 299
linktext 301
List elements
definition list 261
ordered list 278
simple list 282
unordered list 285
list item (li) 271
locality 542
localityname 542
Localization attribute group 653
locktitle attribute 656
lomAggregationLevel 596
lomContext 597
lomCoverage 597
lomDifficulty 598
lomInstallationRemarks 598
lomIntendedUserRole 599
lomInteractivityLevel 600
lomInteractivityType 600
lomLearningResourceType 601
lomOtherPlatformRequirements 601
lomSemanticDensity 602
lomStructure 603
lomTechRequirement 603
lomTypicalAgeRange 604
lomTypicalLearningTime 604
machine industry task 520
maintainertitle 483
map 302
map elements
  anchor 305
  keywords 327
  map 302
  navref 306
  reicell 309
  reicspec 309
  relheader 309
  relrow 308
  reltable 306
  shortdesc 251
  topicmeta 304
  topicref 303
  ux-window 311
map reference anchor point 305
map-to-map cascading
  attributes 51
  exceptons 53
  metadata elements 53
mapgroup elements
  anchorref 313
  keydef 315
  mapref 315
  topicgroup 316
  topichead 317
  topicset 318
  topicsetref 319
mapkeyref attribute 662
mapref 315
markup domain
  markupname 492
  overview 492
markupname 492
MathML domain
  mathml 493
  mathmlref 493
  overview 492
mathmlref 493
menucascade 531
messagepanel 356
metadata 304, 320, 328
Metadata attribute group 651
middlename
  xNAL elements
    middlename 543
Miscellaneous elements
  draft-comment 264
  fn 267
  tm (trademark) 284
month 483
msgblock 513
msgnum 514
msgph 513
name attribute 659
namedetails
  xNAL elements
    namedetails 543
navref 306
navtitle 250
navtitle attribute 663
no-topic-nesting 411
noconds 522
nosafety 529
nospare 529
nosupeq 526
nosupply 527
note 274
notices 471
numcharref 533
object 274
ol (ordered list) 278
oper (syntax operator) 506
option 497
ordered list (ol) 278
organization 483
organizationinfo
  xNAL elements
    organizationinfo 544
organizationname
  xNAL elements
    organizationname 544
organizationnamedetails
  xNAL elements
    organizationnamedetails 545
otherinfo
  xNAL elements
    otherinfo 545
othermeta 328
otherprops attribute 652
otherrole attribute 677
outputclass attribute
  example 44
    overview 44
overline 359
p (paragraph) 278
paragraph (p) 278
param 279
parameter description (pd) 499
parameter list (parml) 498
parameter list entry (plentry) 498
parameter term (pt) 499
parameterentry 533
parml (parameter list) 498
parmname 497
part 472
pd (parameter description) 499
permissions 329
perscat 523
perskill 524
person 484
personinfo
  xNAL elements
  personinfo 546
personname
  xNAL elements
  personname 546
personnel 523
ph (phrase) 280
phrase (ph) 280
platform 330
platform attribute 652
plentry (parameter list entry) 498
postalcode 547
postreq 437
pre (preformatted text) 280
preface 473
preformatted text (pre) 280
prereqs 520
prereq 424
print attribute 657
printlocation 484
processing-role attribute 655
prodinfo 330
prodname 330
product attribute 652
prognum 331
programming elements
  apiname 495
    codeph (code phrase) 497
    fragment (syntax fragment) 502
    option 497
    parml 498
    parmname 497
    pd (parameter description) 499
    plentry (parameter list entry) 498
    pt (parameter term) 499
    synbk (syntax block) 504
    synph 499
  codeblock 496
    delim (syntax delimiter) 507
    fragref (fragment reference) 503
    groupchoice 501
    groupcomp 501
    groupsq 500
    kwd (syntax diagram keyword) 505
    oper (syntax operator) 506
    repsep (repeat syntax separator) 508
    sep (syntax separator) 508
    synnotef (syntax note) 504
    synnoteref 505
    syntax diagram 500
    var (syntax diagram variable) 506
  prolog 320
  prolog elements
prolog elements (continued)
  audience 321
  author 322
  brand 323
  category 323
  component 324
  copyholder 325
  copyright 324
  copyyear 325
  created 326
  critdates 326
  featnum 327
  keywords 327
  metadata 328
  othermeta 328
  permissions 329
  platform 330
  prodinfo 330
  prodname 330
  prognum 331
  prolog 320
  publisher 331
  resourceid 331
  revised 333
  series 333
  source 334
  vrmlist 334
  propdesc 443
  propdeschd 442
  properties 440
  property
    data 407
    data-about 409
  prophead
    reference elements
      prophed 441
  props attribute 651
  proptype 443
  proptypehd 441
  propvalue 443
  propvaluehd 442
  pt (parameter term) 499
  published 485
  publisher 331
  publisherinformation 485
  publishtype 485
Q
q (short quote) 281
query attribute 663
quote, long (lq) 273
quote, short (q) 281
R
refbody (reference body) 439
refbodydiv 439
refcols attribute 663
reference 438
reference elements
Reference elements (continued)
  propdesc 443
  propdeschd 442
  property 442
  proptype 443
  proptypehd 441
  propvalue 443
  propvaluehd 442
  refbody 439
  reference 438
  refsyn 440
Reference elements
  properties 440
  refsyn 440
Related links elements
  link 297
  linkinfo 301
  linklist 298
  linkpool 299
  linktext 301
Related-links 256
RelatedSubjects 388
Relcell 309
Relcolspec 309
Relcolwidth attribute 662
Release management domain
  change-completed 509
  change-historylist 509
  overview 509
Release-management domain
  change-item 510
  change-organization 510
  change-person 511
  change-request-id 511
  change-request-reference 511
  change-request-system 512
  change-revisionid 512
  change-started 512
Release-management domain elements
  change-summary 513
Relheader 309
Relrow 308
Reltable 306
Remedy 447
Repsep (repeat syntax separator) 508
Reqcond 522
Reqconds 521
Reqcontp 522
Reppers 523
Required-cleanup 413
Resourceid
  appid 331
  apname 331
  id 331
  overview 331
ResponsibleParty 447
Result 437
Rev attribute 653
Reviewed 486
Revised 333
Revisionid 486
Revisions 414
Role attribute 677
Row 293
Rowsep attribute 658
S
  Safecond 529
  Safety 530
  Scale attribute 661
  Schemeref 376
  Scope attribute 661
  Screen 532
  Search attribute 657
  Searchtitle 249
  Section 281
  Sectiondiv 282
  Sep (syntax separator) 508
  Series 333
  Shape 362
  Short description 251
  Short quote (q) 281
  Shortcut 532
  Shortdesc 251
  Simple list (sl) 282
  Simple list item (sli) 283
  Simpletable 295
  Simpletable attribute group 662
  Sl (simple list) 282
  Sli (simple list item) 283
Software elements
  Cmdname 514
  Filepath 515
  Msgnum 514
  Msgph 513
  Systemoutput 516
  Userinput 515
  Varname (variable name) 514
Software elements
  Msgblock 513
  Sort-as 111, 363
  Sorting 111
  Source 334
  Spare 528
  Spares 528
  Sparesli 528
Speccentry attribute 663
Specialization attributes group 663
Specialization elements
  Boolean 406
  Data 407
  Data-about 409
  Foreign 410
  Index-base 411
  Itemgroup 411
  No-topic-nesting 411
  State 412
  Term 283
  Unknown 412
Speccultimate attribute 663
Started 486
State 412
Status attribute 653
stentry 297
step 427
step command (cmd) 428
stepresult 435
steps 425, 426
steps-informal element 426
stepsection 427
steptroubleshooting 434
stepxmp 429
sthead 296
strow 296
sub 358
subjectCell 405
subjectdef 387
subjectHead 385
subjectHeadMeta 386
subjectref 395
subjectRel 392
subjectRelHeader 390
subjectRelTable 389
subjectRole 393
subjectScheme 371
subjectScheme elements
  attributedef 384
  defaultSubject 385
  elementdef 387
  enumerationdef 382
  hasInstance 378
  hasKind 379
  hasNarrower 379
  hasPart 380
  hasRelated 381
  relatedSubjects 388
  schemeref 376
  subjectdef 387
  subjectHead 385
  subjectHeadMeta 386
  subjectRel 392
  subjectRelHeader 390
  subjectRelTable 389
  subjectRole 393
  subjectScheme 371
subjectScheme map 371
substep 429
substeps 429
summary 487
sup 357
supeqil 525
supequil 525
supplies 526
supply 526
supplyli 527
SVG domain
  svg-container 516
  svgref 518
subjectdef 387
svg-container 516
svgref 518
synbik (syntax block) 504
synnote (syntax note) 504
synnoteref 505
synph 499
syntax delimiter (delim) 507
syntax diagram variable (var) 506
syntax separator (sep) 508
syntaxdiagram (syntax diagram) 500
systemoutput (system output) 516
T
table 287
table elements
colspec 292
entry 293
row 293
simpletable 295
stentry 297
sthead 296
strow 296
table 287
tbody 293
tgroup 292
thead 293
Task elements
  chdesc 433
  chdeschd 431
  chhead 430
  choice 434
  choices 434
  choption 433
  choptionhd 431
  chrow 432
  context 425
  info 428
  postreq 437
  prereq 424
  result 437
  step 427
  stepresult 435
  steps 425, 426
  steptroubleshooting 434
  substeps 429
  task 423
  taskbody 424
  tasktroubleshooting 436
  tutorialinfo 436
thoroughfare 547
title 248
xNAL elements *(continued)*

locality 542
localityname 542
postalcode 547
thoroughfare 547
url 548

xref (cross-reference) 285
xtrc attribute 654
xtrf attribute 654

Y

year 488