

RichDoc Framework XML Format Specification

Michal Šev enko <sevcenko@vc.cvut.cz>, October 9, 2006

This document is specification of the RichDoc Framework XML scheme. Since the scheme is quite large, elements and attributes are divided into several packages, according to their logical contribution to the format.

The specification contains definitions of three kinds of entities: XML elements, XML attributes, and classes of XML elements. Classes are used to group together similar elements that share attributes, children, or parents.

Each class or element definition includes list of attributes, list of allowed children, and list of allowed parents. Attribute definitions include allowed values, and elements or classes for which they may be used. Note that several identifiers may be used for two or more elements or attributes, in which case they are disambiguated by further text in parentheses, for instance, `td(array)` is different from `td(table)`.

Some elements, attributes, or particular occurrences thereof, may be marked as deprecated. This means that this version of the specification discourages their use, although the RichDoc Framework compatible with this or any later version should be able to process files containing deprecated constructs. If you load and save a document with up-to-date version of the RichDoc Framework, you should get document that does not contain any deprecated constructs.

This XML scheme may be extended by third parties, either from the top, i.e. it may be embedded into some superordinate XML format, or from the bottom, i.e. user may embed his objects into the RichDoc document model, or both. This scheme is internally kept in an interactive XML Schema Management Tool, which you may use to add your specific elements to this scheme, and generate the corresponding XML Schema or DTD. Please contact us to obtain the XML Schema Management Tool and an appropriate representation of this scheme.

Note that this specification in fact describes a family of mutually interconnected schemes, which differ only in the root element. The following elements may be used as XML document root:

document

Specifies a RichDoc document or its part, if multi-section multi-file mode is used.

styleFile

Specifies one cascade of the style sheet mechanism. May be stored in `viewStyles.xml` or `documentStyles.xml` files, depending on what type of style information is provided.

userIndex

Describes user index data structure associated with a document, if multiple-file serialization mode is used. The user index definition is usually stored in file named `userIndex.xml`.

node

Describes the table of contents of document stored in multiple-file serialization mode. It is usually stored in file named `toc.xml`.

bibliography

Specifies the bibliography list associated with a document, if multiple-file serialization mode is used. It is usually stored in file named `bibliography.xml`.

idIndex

Specifies the cross-reference targets index. It is stored in file `idIndex.xml`.

1 Package core

Class styleElement

A class of elements that may contain custom style definition. This includes contentElements, i.e. elements that correspond to visible material, or for example paragraphFragments, that only group visible material together.

Attributes: class id userId language

Content: style? indexEntryRef?

Class contentElement

Elements that represent visible parts of the document, such as words, paragraphs or sections.

Attributes: class id userId language

Content: style? indexEntryRef?

Class numbered

Elements that are automatically numbered, such as sections, floatings or displayedMath equations. The numbering may be suppressed by the hasNumber element.

Attributes: hasNumber class id userId language

Content: style? indexEntryRef?

Class titled

Element that may have title, in the form of the title child element. Any element may have at most one title, and the title must be the first block element of the enclosing element.

Attributes: class id userId language

Content: style? indexEntryRef? title? *block**

Class other

Attribute id

This attribute is used throughout the RichDoc framework format to provide globally-unique identifiers for particular elements of the framework. This identification is usually synthesized automatically, and is not intended to be displayed to the user. There is no requirement on possible values of the ID, but it is sometimes used as a file name, so it should not contain any values that prevents such usage. Also note that although it is intended to be interpreted case-sensitive, using it in case-insensitive contexts, such as MS Windows-based file systems, may cause problems. Currently, the framework synthesizes the id as twelve-byte random number, encoded using modified BASE-64 encoding. The modification replaces the slash character '/' with the hyphen character '-'.

Attribute userId

Identification of element that is intended to be provided and referenced by end users. There is, however, no guarantee that this identifier is unique.

Attribute class

Specifies a variant of the element. The usage is similar to HTML class mechanism.

Attribute language

Specifies the language of this element and its children, unless the children override the value. The value should be Java-compatible locale specification, such as `cs`, `en`, or `en_US`.

Attribute hasNumber

Supresses numbering of the enclosed numbered element.

Attribute title

Attribute width(bitmap)

Attribute height(bitmap)

2 Package document

Element document

This is the root element of RichDoc document file. It primarily contains single section child, representing the body of the document. It may optionally contain other structures, such as custom styles, index, or bibliography, or list of embedded files. Occurrence of these auxiliary children depends on the serialization mode as follows:

Single-file mode

Styles, index and bibliography, if present, are stored in place as children of the document element. The document does not contain the list of embedded files.

Multiple-file mode

Styles, index and bibliography are stored in separate files, as well as any embedded binary data. The `embeddedFiles` child element contains the list of embedded files.

Attributes: class language id level

Content: `embeddedFiles?` `viewStyles?` `documentStyles?` `bibliography?` `userIndex?` (section | note)

Element embeddedFiles

List of embedded files, such as raster image data, that are stored externally to the containing XML document, if multiple-file serialization mode has been used. This element is not used in single-file serialization mode.

Parents: document

Content: `file(embeddedFiles)+`

Element file(embeddedFiles)

Parents: embeddedFiles

Attributes: name(file)

Element explicitTitle

Sections usually use title element to both to display their title as a block material, AND use this title for various bookkeeping operations, such as building the table of contents. If you want the displayed title to be different from the logical title, or in particular, you do not want to display any title but still want to provide logical title, you may use the `explicitTitle` attribute to define the logical title.

Parents: section

Mixed Content: (*inline* | *paragraphFragment*)*

Element viewStyles

Specifies document-specific view styles, that override the view styles inherited from the default context. They may be used for example for changing various visual aspects, or for adding document-specific class definitions.

Parents: document

Content: styles*

Element documentStyles

Parents: document

Content: styles*

Attribute level

In multiple-file multiple-section serialization mode, the level attribute is used to specify how deep is the enclosing document element from the global document root. For instance, for a parted book, the root element has level 0, and thus omits the level attribute, parts have level 1, and chapters have level 2.

Attribute name(file)

3 Package style

Element styleFile

Style file is a container for styles structures. One style file is one link in a cascading style sheet chain.

Content: styles*

Element styles

Styles structure is a container for style structures. One styles structure provides rules for rendering a document in one context, such as online display, printing, HTML export, or PDF export. Styles struc-

tures may be organized into hierarchies bound using the `basedOn` attribute, which are then used to inherit contained style structures.

Parents: `styleFile` `viewStyles` `documentStyles`

Attributes: `name(style)` `basedOn`

Content: `style*`

Element style

Style structure is a container for `value(style)` structures. One structure is intended to provide rules for particular document elements, such as paragraphs, tables, lists, list items etc. The name of the style structure is important for matching the style structure with the elements to which it should apply. Style structures may be organized into hierarchies using the `basedOn` attribute, so that they inherit `value(style)` structures from each other.

The `style` element may be also used as a container of style values associated with particular document element, when user specified some property of the element in an ad hoc manner. Any `styleElement` therefore may have child `style` element.

Parents: `styles` `styleElement`

Attributes: `name(style)` `basedOn`

Content: `value(style)*`

Element value(style)

Style value defines one visual rendering rule to be applied on associated document element. The rule may provide foreground color, background color, font name, margin size etc.

Parents: `style`

Attributes: `name(style)` `type(styleValue)` `value(style)` `inheritable`

Mixed Content: (*inline* | *paragraphFragment*)*

Attribute name(style)

A name of styles, `style` or `value(style)` structure.

Attribute basedOn

This attribute is used to reference styles or style structure from which definitions should be inherited. The link is made according to the referenced `name(style)` attribute.

Attribute type(styleValue)

Type of `value(style)`, such as `Color`, `String` or `Float`. If the `name(style)` of the enclosing value corresponds to a known style value, defined in the file `styleTemplate.xml`, the `type` attribute may be omitted.

Attribute inheritable

Specifies whether given `value(style)` associated with some document element may be inherited by style children of that document. For instance, values like colors are usually inherited, while values like margins are not.

Attribute value(style)

The value of the value(style) structure, such as "Times" or "1.0". It is interpreted according to the type(styleValue) attribute, which is either explicitly specified, or inferred from the styleTemplate.xml file. If the value type is Xml (XML markup), it may be specified either as the value of this attribute, or in place as the content of the enclosed value(style) element, if this is more convenient.

4 Package block

Class block

This is class of elements that may be used wherever block elements are expected, such as paragraphs, lists or tables. There is no assumption about children of this element.

Parents: *blockContainer*

Attributes: class id userId language

Content: style? indexEntryRef?

Class blockContainer

This class prescribes that children of this element are of class block, that is paragraphs, lists, tables etc. There is no assumption about the parent of this element, that is, it may be used in various contexts. Examples of direct subclasses are list items or table cells, which may contain block material, but themselves must be used within special contexts (lists or table rows).

Content: *block**

Class blockParent

Element that is both block and blockContainer.

Parents: *blockContainer*

Attributes: class id userId language

Content: style? indexEntryRef? *block**

Element page

Generic block element, something like HTML's div element. It may be used e.g. to create variants using the class mechanism.

Parents: box linDim *blockContainer*

Attributes: class id userId language

Content: style? indexEntryRef? *block**

Element section

Document section. Sections define hierarchical structure of a document. Note that parent of a section may be only another section, or the document, which is parent of the root section. There may be only one root section. Note that if this section has subsections, they must be the last block children.

Parents: document section none

Attributes: showInToc hasNumber class id userId language

Content: style? indexEntryRef? explicitTitle? title? *block** section*

Element floating

A LaTeX-like floating block. It has two primary functions:

1. To provide title and number to the enclosed material, which is mostly a table or a figure.
2. To declare the floating status, that is, to ensure that the whole block is moved to the next page if necessary during the page breaking algorithm.

Note that this element is expected to contain exactly two block children: the title (which must be always the first child, even if it is displayed below the second child), and the actual contained material. The contained material, however, may be a page containing more elements, e.g. a sequence of paragraphs.

The type of the floating is usually indicated by the class attribute, which is usually table or figure. This information is used to correctly update the table and figure counters.

Parents: *blockContainer*

Attributes: class id userId language

Content: style? indexEntryRef? title *block*

Element p(block)

This is block element representing regular document paragraphs.

Parents: *blockContainer*

Attributes: class id userId language

Mixed Content: style? indexEntryRef? (*inline* | *paragraphFragment*)*

Element pre

A variant of paragraph that preserves white spaces, and is usually (not necessarily) associated with fixed width font. Note that in RichDoc framework, unlike HTML, even regular paragraph p preserves whitespace.

Parents: *blockContainer*

Attributes: class id userId language

Mixed Content: style? indexEntryRef? (*inline* | *paragraphFragment*)*

Element hr

Horizontal rule.

Parents: *blockContainer*

Attributes: class id userId language

Content: style? indexEntryRef?

Element toc

Table of contents page. This element has no children, it is only placeholder for the table of contents material. The table of contents is constructed dynamically when document is built from XML source, and is not written back to the XML.

Parents: *blockContainer*

Attributes: level hasNumber class id userId language

Content: style? indexEntryRef?

Element indexPage

Index page. This element has no children, it is only placeholder for the index material. The index is constructed dynamically when document is built from XML source, and is not written back to the XML.

Parents: *blockContainer*

Attributes: *hasNumber class id userId language*

Content: *style? indexEntryRef?*

Element bibPage

Bibliography page. This element has no children, it is only placeholder for the bibliography material. The bibliography is constructed dynamically when document is built from XML source, and is not written back to the XML.

Parents: *blockContainer*

Attributes: *hasNumber class id userId language*

Content: *style? indexEntryRef?*

Element glossaryPage

Glossary page. This element has no children, it is only placeholder for the glossary material. The glossary is constructed dynamically when document is built from XML source, and is not written back to the XML.

Parents: *blockContainer*

Attributes: *hasNumber class id userId language*

Content: *style? indexEntryRef?*

Element group(block)

A group block element. It is used to group together several *displayedMath* equations, to provide single major number and sequential minor numbers.

Parents: *blockContainer*

Attributes: *class id userId language*

Content: *style? indexEntryRef? block**

Element sideways

Special block element that rotates its content 90 degrees counter-clockwise. Note that it provides no wrapping mechanism, so the contained material gets maximum possible width. It is useful for example for table row headings.

Parents: *blockContainer*

Attributes: *class id userId language*

Content: *style? indexEntryRef? block**

Element title

A paragraph that is considered a title of the enclosing titled element. It is displayed as normal paragraph (perhaps with special style), and besides is used for various bookkeeping operations, such as building table of contents or list of figures. Note that the enclosing element may have at most one title child, which must be the first block element.

Parents: *titled node*

Attributes: *class id userId language*

Mixed Content: style? indexEntryRef? (*inline* | *paragraphFragment*)*

Element table

A table. It has very similar semantics to HTML tables, i.e. it contains cells in `td(array)` elements, which are organized in rows in `tr(table)` elements. The cells may span rows and/or columns. Note that unlike HTML, there is a requirement that the table structure is consistent. The following requirements must be fulfilled:

1. The cell structure must have rectangular shape.
2. The cells must not overlap.
3. At least one cell must start in each row and in each column.

Figure 1 shows examples of violations of these rules.

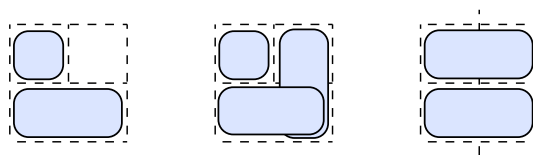


Figure 1 Violations of Table Structure Requirements

Parents: `tabBox` *blockContainer*

Attributes: `class` `id` `userId` `language`

Content: style? indexEntryRef? `tr(table)`+

Class tr

Definition of a row of a table or array. It is a list of cells (`td` elements) that start in this row, but may span also to rows below this row.

Attributes: `class` `id` `userId` `language`

Content: style? indexEntryRef?

Element tr(table)

A table row (`tr`).

Parents: `table`

Attributes: `class` `id` `userId` `language`

Content: style? indexEntryRef? `td(table)`+

Class td

Definition of table or array cell. The cell may span several columns or rows, as defined by the `colspan` and `rowspan` attributes.

Attributes: `rowspan` `colspan` `class` `id` `userId` `language`

Content: style? indexEntryRef?

Element td(table)

A table cell (`td`).

Parents: `tr(table)`

Attributes: `horizontalAlignment` `verticalAlignment` `valign` `fill` `rowspan` `colspan` `class` `id` `userId` `language`

Content: style? indexEntryRef? *block**

Class list

Different types of lists, that are used to contain items defined in li elements.

Parents: *blockContainer*

Attributes: class id userId language

Content: style? indexEntryRef? li+

Element ul

Unordered list, which marks its items with equal symbols.

Parents: *blockContainer*

Attributes: class id userId language

Content: style? indexEntryRef? li+

Element ol

Ordered list, which marks its items by a sequence of symbols, such as Arabic numbers, Roman numbers, letters of Latin or Greek alphabet, etc.

Parents: *blockContainer*

Attributes: class id userId language

Content: style? indexEntryRef? li+

Element todo

List that marks its items with checked or unchecked boxes.

Parents: *blockContainer*

Attributes: class id userId language

Content: style? indexEntryRef? li+

Element li

An item of ordered, unordered or todo list. The checked status of todo list item is usually indicated by setting the class to done.

Parents: *list*

Attributes: class id userId language

Content: style? indexEntryRef? *block**

Element dl

A HTML-like definition list. It consist of series of definition terms defined using dt element, and definition descriptions, defined using dd element. Although in most cases these two elements alternate, there is no such requirement; you may, for example, use two terms followed by single definition.

Parents: *blockContainer*

Attributes: class id userId language

Content: style? indexEntryRef? (dt | dd)+

Element dt

Term item of a definition list defined using the dl element. It is of type paragraph, so it may contain inline elements.

Parents: dl

Attributes: class id userId language

Mixed Content: style? indexEntryRef? (*inline* | *paragraphFragment*)*

Element dd

Description item of definition list. It may contain block material.

Parents: dl

Attributes: class id userId language

Content: style? indexEntryRef? *block**

Attribute showInToc

This attribute may be used to prevent section showing in table of contents. By default, all sections show in the table of contents.

Attribute rowspan

Specifies how many rows are occupied by enclosing td cell. The default is 1.

Attribute colspan

Specifies how many columns are occupied by enclosing td cell. The default is 1.

Attribute fill

Specifies the weight of this cell when a group of cells should be enlarged. Assume that cells c_1, c_2, \dots, c_n have total width W , and we want them to have them some larger width $W + \Delta$. In such case, the width of each cell is enlarged by the amount of $\Delta \frac{\omega_i}{\Omega}$, where ω_i is the fill weight of the i -th cell, and $\Omega = \sum_i \omega_i$. If all cells have zero weight, i.e. $\Omega = 0$, we assume that $\omega_i = 1$ for each i . The default value of this attribute is 0.

The need for enlargement occurs for example when we force full width of the table, or if we have group of cells that are in column with column-spanning cell requiring more room.

5 Package paragraph

Class p

This class represents elements that may contain inline material, like text, inline math, or paragraph fragments. No assumption is made about parents of element represented by this class.

Mixed Content: (*inline* | *paragraphFragment*)*

Class inline

Inline elements are elements that may appear in a paragraph (any element based on the class p) such as inline math, icons, or inline bitmaps. Note that paragraphs may also contain other elements, namely *paragraphFragments*, which do not correspond to document elements, but provide style inheritance for contained inline elements or paragraph fragments.

Parents: *paragraphFragment* p

Attributes: class id userId language

Content: style? indexEntryRef?

Class paragraphFragment

Paragraph fragments are containers of inline elements, other paragraph fragments, or text. They usually provide style information for the contained material, like **b** or *i* elements.

Parents: *paragraphFragment p*

Attributes: class id userId language

Mixed Content: style? indexEntryRef? (*inline | paragraphFragment*)*

Element i

Paragraph fragment for *italic* style text.

Parents: *paragraphFragment p*

Attributes: class id userId language

Mixed Content: style? indexEntryRef? (*inline | paragraphFragment*)*

Element b

Paragraph fragment for **bold** style text.

Parents: *paragraphFragment p*

Attributes: class id userId language

Mixed Content: style? indexEntryRef? (*inline | paragraphFragment*)*

Element u

Paragraph fragment for underline style text.

Parents: *paragraphFragment p*

Attributes: class id userId language

Mixed Content: style? indexEntryRef? (*inline | paragraphFragment*)*

Element code

Paragraph fragment for underline style text.

Parents: *paragraphFragment p*

Attributes: class id userId language

Mixed Content: style? indexEntryRef? (*inline | paragraphFragment*)*

Element tt

Paragraph fragment for typewriter style text.

Parents: *paragraphFragment p*

Attributes: class id userId language

Mixed Content: style? indexEntryRef? (*inline | paragraphFragment*)*

Element span

Generic paragraph fragment.

Parents: *paragraphFragment p*

Attributes: class id userId language

Mixed Content: style? indexEntryRef? (*inline | paragraphFragment*)*

Element strike

Render ~~strike-through~~ style text.

Parents: *paragraphFragment p*

Attributes: class id userId language

Mixed Content: style? indexEntryRef? (*inline | paragraphFragment*)*

Element sf

Paragraph fragment for sans-serif style text.

Parents: *paragraphFragment p*

Attributes: class id userId language

Mixed Content: style? indexEntryRef? (*inline | paragraphFragment*)*

Element tiny

Paragraph fragment for tiny style text.

Parents: *paragraphFragment p*

Attributes: class id userId language

Mixed Content: style? indexEntryRef? (*inline | paragraphFragment*)*

Element Large

Parents: *paragraphFragment p*

Attributes: class id userId language

Mixed Content: style? indexEntryRef? (*inline | paragraphFragment*)*

Element Huge

Parents: *paragraphFragment p*

Attributes: class id userId language

Mixed Content: style? indexEntryRef? (*inline | paragraphFragment*)*

Element hyperLink

Parents: *paragraphFragment p*

Attributes: morph cap(capitalization) includeTargetName i class id userId language

Mixed Content: style? indexEntryRef? (*inline | paragraphFragment*)*

Element uriRef

Parents: *paragraphFragment p*

Attributes: uri class id userId language

Mixed Content: style? indexEntryRef? (*inline | paragraphFragment*)*

Element br

Line break. Forces typeset on next line, without splitting a paragraph, thus preserving line spacing and indentation.

Parents: *paragraphFragment p*

Attributes: class id userId language

Content: style? indexEntryRef?

Element icon

One of predefined icons. Icons are specialized using the type(icon) attribute.

Parents: *paragraphFragment p*

Attributes: alt type(icon) class id userId language

Content: style? indexEntryRef?

Element cite

Parents: *paragraphFragment p*

Attributes: refId class id userId language

Content: style? indexEntryRef?

Element inlineBitmap

Parents: *paragraphFragment p*




Attributes: class id userId language height(bitmap) width(bitmap) data scale

Content: style? indexEntryRef? sourceInfo?

Attribute refId

Attribute type(icon)

Type of predefined icon. The following icons are currently recognized:

Type	Alt	Icon
happy	:-)	
confused	:-	
sad	:-(

Attribute alt

Attribute uri

Attribute morph

Attribute cap(capitalization)

Attribute includeTargetName

Attribute i

Integer reference to some object representing the hyperlink target. Used when hyperlinks point to custom object that are serialized outside the RichDoc document.

6 Package math

Class mathBox

Class of elements that form mathematic formulas. The basic property is that they may appear as children of mathContainers.

Parents: *mathContainer*

Attributes: class id userId language

Content: style? indexEntryRef?

Class mathContainer

Class of elements that may contain mathBoxes. There is no assumption about parents of these elements.

Content: *mathBox**

Class mathParentBox

Class of elements that are both mathBoxes and mathContainers.

Parents: *mathContainer*

Attributes: class id userId language

Content: style? indexEntryRef? *mathBox**

Class mathText

Class of mathBoxes that provide text with various semantics, such as numbers, variables, constants, vectors etc.

Parents: *mathContainer*

Attributes: text class id userId language

Content: style? indexEntryRef?

Element math

Inline equation, which may be placed in a paragraph.

Parents: *paragraphFragment p*

Attributes: class id userId language

Content: style? indexEntryRef? *mathBox**

Element displayedMath

This block element represents a “displayed” math, that is, mathematic formula that is typeset within the full width of the display, and is separated by vertical space from its neighborhood. It consists by one or more rows, each defining one visual row of the formula (e.g. one equation).

Parents: *blockContainer*

Attributes: numberPerRow numbered hasNumber class id userId language

Content: style? indexEntryRef? ((row | p(displayedMath))* | $(displayed)$)

Element row

One row of a displayedMath block. The contained material may be aligned using the tab and glue elements.

Parents: displayedMath

Attributes: hasNumber class id userId language

Content: style? indexEntryRef? *mathBox**

Element p(displayedMath)

Paragraph interleaved with rows of enclosing displayedMath. May be useful to insert paragraph material into existing displayedMath without interrupting it, to preserve numbering or alignment.

Parents: displayedMath

Attributes: class id userId language

Mixed Content: (*inline* | *paragraphFragment* | indexEntryRef | style)*

Element tab

Defines position that should be aligned to certain horizontal position for all rows of displayedMath. The alignment attribute specifies how the material before the tab is aligned to the tab.

Parents: *mathContainer*

Attributes: alignment class id userId language

Content: style? indexEntryRef?

Element glue

Under construction.

Parents: *mathContainer*

Attributes: class id userId language

Content: style? indexEntryRef?

Element op

Mathematical operator. The type is defined using the text attribute. For valid values see Operators and Symbols in RichDoc User Manual.

Parents: *mathContainer*

Attributes: text class id userId language

Content: style? indexEntryRef?

Element sym

Mathematical symbol. The type is defined using the text attribute. For valid values see Operators and Symbols in RichDoc User Manual.

Parents: *mathContainer*

Attributes: text class id userId language

Content: style? indexEntryRef?

Element hspace

Horizontal space used in math typesetting.

Parents: *mathContainer*

Attributes: name(hspace) negative repeat class id userId language

Content: style? indexEntryRef?

Element frac

A fraction, consisting of numerator and denominator parts, such as $\frac{a}{b}$.

Parents: *mathContainer*

Attributes: class id userId language

Content: style? indexEntryRef? num(numerator) den

Element num(numerator)

Numerator part of a fraction.

Parents: frac

Content: *mathBox**

Element den

Denominator part of a fraction.

Parents: frac

Content: *mathBox**

Element script

A script symbol with optional subscript in sub element and superscript in sup element, the main body being in the body element. For example, x_1 , x^n , or x_1^2 .

Parents: *mathContainer*

Attributes: class id userId language

Content: style? indexEntryRef? body? sub? sup?

Element body

The body part of certain compound mathematical constructs.

Parents: script sum sqrt comp verticalFence stackrel

Content: *mathBox**

Element sub

Subscript part of a script or comp boxes.

Parents: script sum comp

Content: *mathBox**

Element sup

Superscript part of a script or comp boxes.

Parents: script sum comp

Content: *mathBox**

Element sqrt

Radical symbol, consisting of a body and an optional root, for example, $\sqrt[3]{x}$.

Parents: *mathContainer*

Attributes: class id userId language

Content: style? indexEntryRef? root? body?

Element root

The root part of the sqrt symbol.

Parents: sqrt

Content: *mathBox**

Element verticalFence

A vertical fence structure, consisting of body, and optional top and bottom.

Parents: *mathContainer*

Attributes: bottom top class id userId language

Content: style? indexEntryRef? body top? bottom?

Element top

The top part of the stackrel or verticalFence structures.

Parents: verticalFence stackrel

Content: *mathBox**

Element bottom

Bottom part of verticalFence.

Parents: verticalFence

Content: *mathBox**

Element stackrel

Parents: *mathContainer*

Attributes: class id userId language

Content: style? indexEntryRef? body top

Element comp

One of a number of mathematical structures having subscript, superscript and body, such as sums, limits, integrals, etc. The type of the structure is defined using the type(comp) attribute.

Parents: *mathContainer*

Attributes: type(comp) class id userId language
Content: style? indexEntryRef? sub? sup? body?

Element mpar

A paragraph embedded in math structure.

Parents: *mathContainer*

Attributes: class id userId language

Mixed Content: style? indexEntryRef? (*inline* | *paragraphFragment*)*

Element fence

A fence structure, i.e. structure enclosed in parentheses, brackets, curly brackets etc. The type of left and right fence symbol is specified using the start and end attributes. By default, the fence size scales automatically to accommodate the enclosed material. This may be overridden using the size(fence) attribute.

Parents: *mathContainer*

Attributes: start end size(fence) class id userId language

Content: style? indexEntryRef? *mathBox**

Element not

Not structure, which represents a negation of the operator that is put inside it. For instance, if the operator \models is put inside the not box, it shows up as $\not\models$.

Parents: math *mathContainer*

Attributes: class id userId language

Content: style? indexEntryRef? *mathBox**

Element over

An over structure, such as \dot{x} , \hat{x} , \vec{x} etc. The type is defined by the type(over) attribute.

Parents: *mathContainer*

Attributes: type(over) class id userId language

Content: style? indexEntryRef? *mathBox**

Element array

A mathBox representing rectangular structures like vectors or matrices. It has similar structure as the table element.

Parents: *mathContainer*

Attributes: class id userId language

Content: style? indexEntryRef? tr(array)+

Element tr(array)

An array row (tr).

Parents: array

Attributes: class id userId language

Content: style? indexEntryRef? td(array)+

Element `td(array)`

An array cell (`td`).

Parents: `tr(array)`

Attributes: `rowspan` `colspan` `class` `id` `userId` `language`

Content: `style?` `indexEntryRef?` *mathBox**

Element `var`

`mathText` box with variable semantics.

Parents: *mathContainer*

Attributes: `text` `class` `id` `userId` `language`

Content: `style?` `indexEntryRef?`

Element `num(number)`

`mathText` box with number semantics.

Parents: *mathContainer*

Attributes: `text` `class` `id` `userId` `language`

Content: `style?` `indexEntryRef?`

Element `fun`

`mathText` box with function semantics.

Parents: *mathContainer*

Attributes: `text` `class` `id` `userId` `language`

Content: `style?` `indexEntryRef?`

Element `vec`

`mathText` box with vector semantics.

Parents: *mathContainer*

Attributes: `text` `class` `id` `userId` `language`

Content: `style?` `indexEntryRef?`

Element `const`

`mathText` box with constant semantics.

Parents: *mathContainer*

Attributes: `text` `class` `id` `userId` `language`

Content: `style?` `indexEntryRef?`

Element `dim`

`mathText` box with dimension semantics.

Parents: *mathContainer*

Attributes: `text` `class` `id` `userId` `language`

Content: `style?` `indexEntryRef?`

Element `mat`

`mathText` box with matrix semantics.

Parents: *mathContainer*

Attributes: text class id userId language

Content: style? indexEntryRef?

Element dom

mathText box with domain semantics.

Parents: *mathContainer*

Attributes: text class id userId language

Content: style? indexEntryRef?

Element mathSf

mathText box with generic sans-serif semantics.

Parents: *mathContainer*

Attributes: text class id userId language

Content: style? indexEntryRef?

Element mtext

mathText box with generic text semantics.

Parents: *mathContainer*

Attributes: text class id userId language

Content: style? indexEntryRef?

Attribute numberPerRow

Specifies whether the enclosing displayedMath assigns separate numbers to its rows (if set to true), or if it gets one number collectively (the default).

Attribute alignment

Specifies how the material left to this tab is aligned to this tab. Valid values are *left* (extra space is between the material and this tab) and *right* (extra space is between previous tab or the beginning of the equation, and the material), see Figure 2.

previous tabs

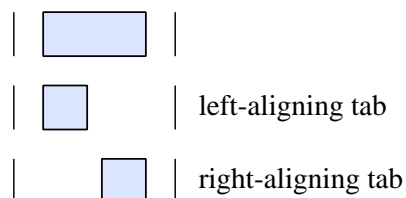


Figure 2 Tabs

Attribute text

Attribute name(hspace)

Specifies the amount of horizontal space of the hspace element. Valid values are thin for 1/16 of the cell size, med for 1/8 of the cell size, thick for 1/4 of the cell size, and quad for 100% of the cell size.

Attribute negative

Whether the enclosing hspace is positive (the default) or negative.

Attribute repeat

Increases the base size of the enclosing hspace by this factor. The default value is 1.

Attribute type(comp)

Specifies the type of the comp structure.

Attribute start

Specifies the type of the opening fence symbol. If omitted, there is no opening fence symbol. Valid values are:

```
(      (
{      {
[      [
|      |
langle <
lfloor  ⌊
lceil  ⌈
```

Attribute end

Specifies the type of the closing fence symbol. If omitted, there is no closing fence symbol. Valid values are:

```
)      )
}      }
]      ]
|      |
rangle >
rfloor  ⌋
rceil  ⌉
```

Attribute size(fence)

Specifies the size of the fence symbols. By default, the fence symbols scale automatically with the enclosed material. You may specify fixed fence size by specifying one of the fixed size values, see below.

auto	0	1	2	3	4
$\left(\int_0^1 x dx\right)$	$\left(\int_0^1 x dx\right)$	$\left(\int_0^1 x dx\right)$	$\left(\int_0^1 x dx\right)$	$\left(\int_0^1 x dx\right)$	$\left(\int_0^1 x dx\right)$

Attribute type(over)

Specifies the type of the enclosing over box.

Attribute bottom

Attribute top

7 Package drawing

Element geometry

Defines box in certain coordinate space, in terms of the anchor point, width, and height. The anchor point refers to the top-left corner of the rectangle, unless otherwise specified. For instance, for box element, the anchor point may be specified by the anchor style value.

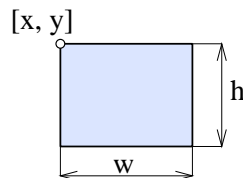


Figure 3 Box Geometry (anchor point refers to top-left corner)

Parents: box tabBox customBox

Attributes: x y w h

Element transform

A drawingContainer that applies specified transformation to its contained drawingElements. The transformation is specified as a series of transformOperations.

Parents: *drawingContainer*

Attributes: userId id

Content: *transformOperation+* (*drawingElement* | transform)+

Class transformOperation

Parents: transform

Element translate

Specifies translation transforming operation. All objects are transformed using transformation matrix

$$\begin{pmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{pmatrix} \quad (1)$$

Parents: `transform`

Attributes: `x` `y`

Element rotate

Specifies rotate transforming operation. All objects are transformed using transformation matrix

$$\begin{pmatrix} \cos \theta & \sin \theta & x(1 - \cos \theta) - y \sin \theta \\ -\sin \theta & \cos \theta & y(1 - \cos \theta) + x \sin \theta \\ 0 & 0 & 1 \end{pmatrix} \quad (2)$$

Parents: `transform`

Attributes: `x` `y` `theta`

Element scale

Specifies scale transforming operation. All objects are transformed using transformation matrix

$$\begin{pmatrix} sx & 0 & x(1 - sx) \\ 0 & sy & y(1 - sy) \\ 0 & 0 & 1 \end{pmatrix} \quad (3)$$

Parents: `transform`

Attributes: `x` `y` `sx` `sy`

Element shear

Specifies shear transforming operation. All objects are transformed using transformation matrix

$$\begin{pmatrix} 1 & sy & -ysy \\ sx & 1 & -xsx \\ 0 & 0 & 1 \end{pmatrix} \quad (4)$$

Parents: `transform`

Attributes: `x` `y` `sx` `sy`

Class drawingContainer

Any element that may contain drawingElements.

Content: (`drawingElement` | `transform`)*

Class drawingElement

Drawing element is contentElement that may appear as a child of drawingContainer.

Parents: `drawingContainer`

Attributes: `class` `id` `userId` `language`

Content: `style?` `indexEntryRef?`

Element drawing

A block element that serves as a container for drawingElements.

Parents: *blockContainer*

Attributes: class id userId language

Content: style? indexEntryRef? animation? (*drawingElement* | transform)*

Element animation

Parents: drawing

Content: rule*

Element rule

Parents: animation

Simple Content: xsd:string

Element shape

Shape is closed or open stroke comprising segments defined using shapeSegment elements. The segments may be straight, quadratic, Bézier or arc.

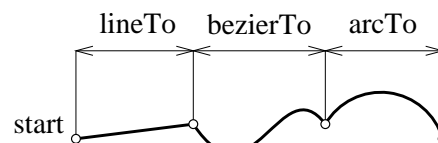


Figure 4 Segmented Shape

Parents: *drawingContainer*

Attributes: closed filled class id userId language

Content: style? indexEntryRef? start *shapeSegment*+

Element start

Defines starting point of the shape. The type attribute controls behavior of the shape when user manipulates the starting point.

Parents: shape

Attributes: x y type(shapeSegment)

Class shapeSegment

Class of elements defining segments of a shape. Each segment has an attribute type that controls the behavior of the shape when user manipulates the point at the end of this segment.

Parents: shape

Attributes: type(shapeSegment)

Element lineTo

Straight shape segment, i.e. straight line from the current point to the point [x, y], see Figure 5.

Parents: shape

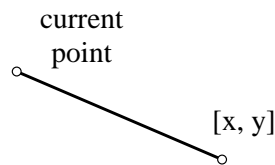


Figure 5 Straight Segment

Attributes: x y type(shapeSegment)

Element quadTo

Quadratic shape segment, i.e. quadratic curve from the current point to the point [x, y], having [cx, cy] as a control point, see Figure 6.

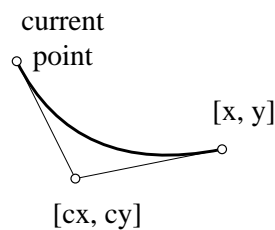


Figure 6 Quadratic Segment

Parents: shape

Attributes: x y cx cy type(shapeSegment)

Element bezierTo

Bézier shape segment, i.e. Bézier curve from the current point to the point [x, y], having [c1x, c1y] and [c2x, c2y] as a control points, see Figure 7.

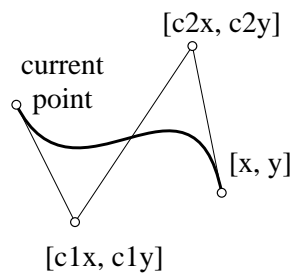


Figure 7 Bézier Segment

Parents: shape

Attributes: x y c1x c1y c2x c2y type(shapeSegment)

Element arcTo

Arc shape segment, i.e. arc segment from the current point to the point [x, y], having [mx, my] as a control point, see Figure 8.

Parents: shape

Attributes: x y my mx type(shapeSegment)

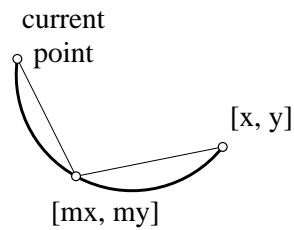


Figure 8 Arc Segment

Element box

drawingElement representing a box with or without content. The general box shape is rectangular, but it may be changed by style to round or oval. Note that the geometry child is required, the omission of it is a deprecated construct, so in the schema it is listed as optional.

Parents: *drawingContainer*

Attributes: x y theta autoLayout width(box) class id userId language

Content: style? indexEntryRef? geometry? page?

Element link

Parents: *drawingContainer*

Attributes: type(link) x1 y1 x2 y2 from to class id userId language

Content: style? indexEntryRef? ((rectangularPoint | bezier))?

Element bezier

Specification of parameters of a Bézier link.

Parents: link

Attributes: length angle baseAngle

Element rectangularPoint

Specifies a hint for rectangular or corner link. For corner link, there are two options how to draw it. The one is chosen for which the real link corner is closer to the hint point, see Figure 9a. In the case of rectangular link, consisting of three segments, the rectangularPoint determines either x or y coordinate of the middle segment of the link. There are also two options, from which the one is chosen for which the first node point is closer to the rectangularPoint, see Figure 9b and c. In the figure, orange point denotes the rectangularPoint, while the two green points denote the two alternatives, from which the closer is taken.

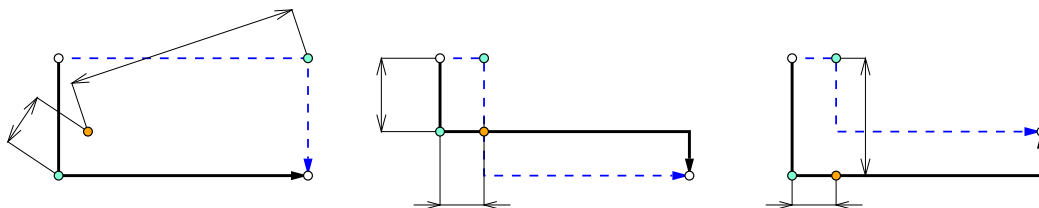


Figure 9 How rectangularPoint influences link shape

Parents: link

Attributes: x y

Element bitmap(drawing)

Parents: *drawingContainer*

Attributes: x y aspect class id userId language height(bitmap) width(bitmap) data scale

Content: style? indexEntryRef? sourceInfo?

Element sourceInfo

Identifies the source from which this bitmap has been imported. It might be used e.g. for updating the bitmap when source changes, or for re-rasterizing the bitmap in another resolution.

Parents: *bitmap*

Attributes: type(sourceInfo) name(sourceInfo) id data width(sourceInfo) height(sourceInfo)

Element tabBox

Parents: *drawingContainer*

Attributes: class id userId language

Content: style? indexEntryRef? geometry table

Element linDim

Linear dimension drawingElement. The elements from and to define endpoints of the dimension arrow. The elements extension1 and extension2 define the length of the extension line. The extension line is always perpendicular to the dimension arrow. If the extension value is positive, the extension extends in the direction at right angle clockwise from the from-to vector. If the value is negative, it extends in counter-clockwise direction. Alternatively, the extension point may be defined using snap child element, in which case it is associated with some point hosted by another object. In this case, the from or to points may be modified to maintain the perpendicular property of the extension line, but they may be moved only along the main dimension line. The text of the dimension is defined using the page element. The position of the page is defined using the alignment element.

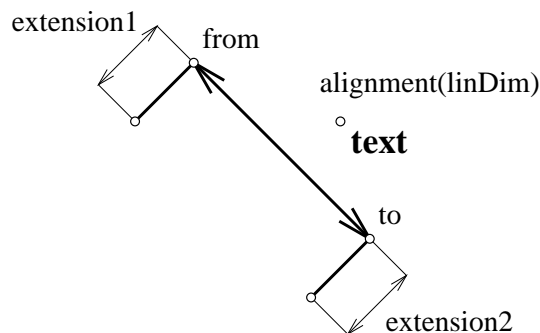


Figure 10 Linear Dimension

Parents: *drawingContainer*

Attributes: class id userId language

Content: style? indexEntryRef? from to extension1 extension2 alignment(linDim) page

Element from

Starting point of linDim drawingElement.

Parents: linDim

Attributes: x y

Element to

Ending point of linDim drawingElement.

Parents: linDim

Attributes: x y

Element alignment(linDim)

Specifies the position of the linDim's annotation.

Parents: linDim

Attributes: type(linDimAlignment)

Content: position

Element position

Parents: alignment(linDim)

Attributes: x y

Element extension1

Defines the direction and length of the first extension point of linDim, if the value(style) is present. Alternatively, the extension point may be defined using nested snap point.

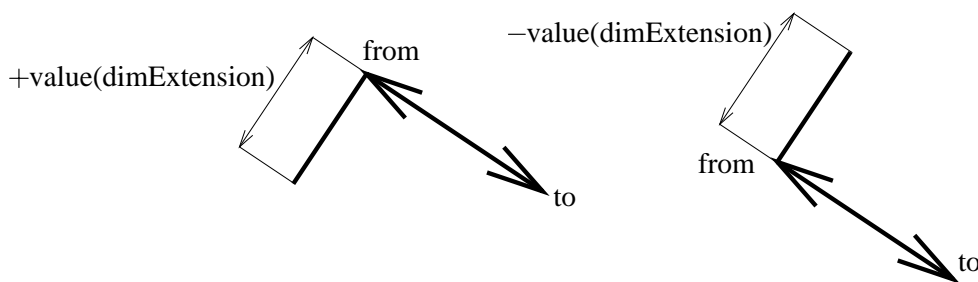


Figure 11 Beginning of Linear Dimension

Parents: linDim

Attributes: value(dimExtension)

Content: snap?

Element extension2

Defines the direction and length of the second extension point of linDim, if the value(style) is present. Alternatively, the extension point may be defined using nested snap point.

Parents: linDim

Attributes: value(dimExtension)

Content: snap?

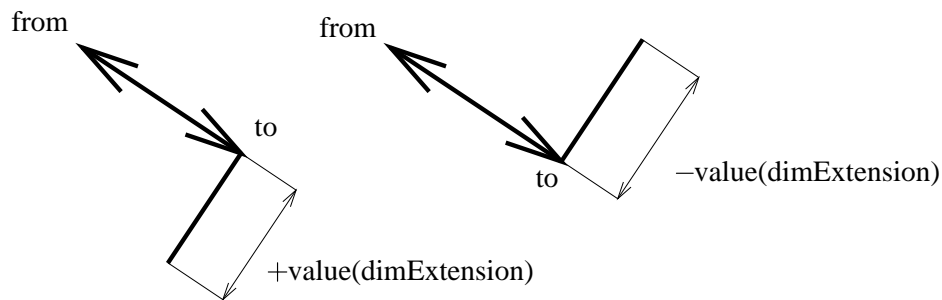


Figure 12 Beginning of Linear Dimension

Element snap

Defines that extension point of `linDim` is associated with certain point of other object. The object is identified by its identifier using the `id` attribute, and its point is identified by the `type(snap)` attribute.

Parents: `extension1 extension2`

Attributes: `id type(snap)`

Element customBox

Custom box is a special box whose content is drawn by user-supplied code. The code is supplied in the form of a bytecode of a class that is expected to be of type `org.ksmsa.richDoc.drawing.view.CustomButtonCodeBase` and override the method `draw(double width, double height)`. The element should provide the bytecode in the compiled child element, and may optionally provide the fragment of the source code in the child element `source`.

Parents: *drawingContainer*

Attributes: `class id userId language`

Content: `style? indexEntryRef? geometry source? compiled`

Element compiled

The compiled part of the enclosing `customBox`, in the form of BASE64-encoded bytecode.

Parents: `customBox`

Simple Content: `xsd:string`

Element source

Contains the fragment of the source code that is used to compile the bytecode of the enclosing `customBox`. The fragment is the body of the method `draw(double width, double height)` of the class extending `org.ksmsa.richDoc.drawing.view.CustomButtonCodeBase`.

Parents: `customBox`

Simple Content: `xsd:string`

Class bitmap

Attributes: `height(bitmap) width(bitmap) data scale class id userId language`

Content: `style? indexEntryRef? sourceInfo?`

Attribute x

Attribute y**Attribute w****Attribute h****Attribute theta****Attribute sx****Attribute sy****Attribute filled****Attribute closed****Attribute type(shapeSegment)**

Controls the behavior of shape when user drags one of its manipulation points.

corner

only the selected manipulation point is affected; this may be used to create discontinuous (corner) points of adjacent bezier curves, see Figure 13 *a*.

straight

ensures first-order continuity when control point is dragged; that is the adjacent control vectors must have the same direction, but may differ in length, see Figure 13 *b*.

smooth

ensures second-order continuity when control point is dragged; that is the adjacent control vectors must be equal in both direction and length, see Figure 13 *c*.

auto

sets the control points automatically when the node point is dragged, see Figure 13 *d*.

Attribute cx

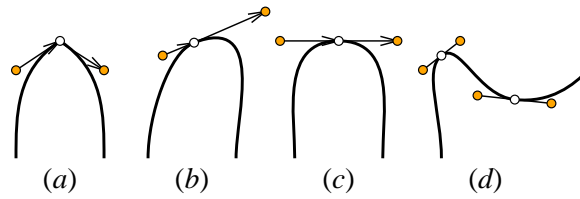


Figure 13 (a) corner, (b) straight, (c) smooth, (d) auto

Attribute cy

Attribute c1x

Attribute c1y

Attribute c2x

Attribute c2y

Attribute mx

Attribute my

Attribute autoLayout

Turns this box to auto-layout mode. In this mode, all boxes that are reachable from this box via links, are positioned out automatically.

Attribute from

Specifies the object associated with the starting point of the enclosing link.

Attribute to

Attribute x1

Attribute x2

Attribute y1

Attribute y2

Attribute type(link)

Attribute length

Distance of the link's control point from its node point, relative to the length of the link, see Figure 14. If the link length is zero (i.e. the link is a loop), the length is relative to the value of 1.

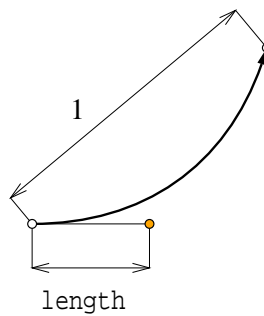


Figure 14 Bézier Link

Attribute baseAngle

Specifies the angle of rotation of Bézier link around its collapsed point, see Figure 15.

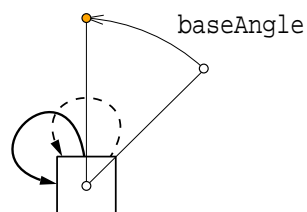


Figure 15 Bézier Link

Attribute angle

Specifies the angle of the control point relative from the link vector for Bézier link, see Figure 16. The angle is relative to the angle of link's base, from-to vector. If the link base vector has zero length, i.e. the link is a loop, the angle is relative to the baseAngle attribute.

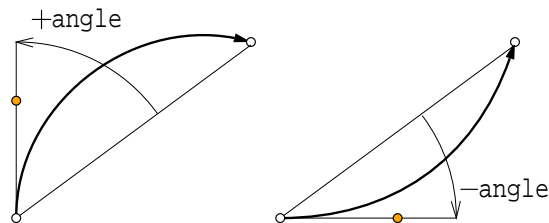


Figure 16 Bézier Link

Attribute aspect

Specifies the aspect ratio of a bitmap, if not equal to 100%. The height of a bitmap is equal to $\text{width}(\text{bitmap}) * \text{aspect}$. If the value is 1, it may be omitted.

Attribute type(linDimAlignment)

Attribute value(dimExtension)

Specifies length and direction of linDim's extension lines, extension1 and extension2.

Attribute type(snap)

Defines which point within particular drawingElement should be associated with enclosing extension point of linDim. The following values are legal, depending on the type of the associated drawingElement referenced by snap's id attribute:

shape

1-start, n -center, n -end, where n is one-based index of shape's segment. That is, 1-start is the starting point of the shape, 1-center is the middle of the first segment, 1-end is the end of the first segment and the beginning of the second segment, and so on.

box

topLeft, top, topRight, left, center, right, bottomLeft, bottom, bottomRight

Attribute type(sourceInfo)

Type of the source used to initialize the enclosing bitmap. The valid values are:

jar

The bitmap has been imported from java archive file. The file is identified by the name(sourceInfo) attribute, which specifies fully-qualified path of the bitmap file within the java archive, for example, org/kmsa/richDoc/view/bold.png.

eps

The bitmap was created by rasterizing Encapsulated Postscript image. The source is identified by binary data, either using the `id` attribute in case of multi-file serialization mode, or `data` attribute in single-file serialization mode. These attributes reference binary stream containing deflated content of the eps file. The `width(sourceInfo)` or `height` attributes may be used if other than native size of the EPS file has been requested.

Attribute name(`sourceInfo`)

Attribute width(`sourceInfo`)

For the bitmap imported from EPS, specifies the desired width of the image in pts (as specified by the LaTeX `\epsfxsize` macro)

Attribute height(`sourceInfo`)

For the bitmap imported from EPS, specifies the desired height of the image in pts (as specified by the LaTeX `\epsfysize` macro)

Attribute scale

Attribute data

8 Package toc

Element node

Specifies one node in the table of contents, corresponding to a section.

Parents: node

Attributes: `id` `hasFile` `decoratedNumber` `localNumber` `locale` `show` ~~`title`~~(~~deprecated~~)

Content: `title?` `node*`

Attribute `hasFile`

Specifies whether the enclosing node corresponds to a section that has its own file, i.e. is the root section of certain document. If a node does not have `hasFile` set, it is interpreted as follows. If it has a node ancestor that has `file`, the node corresponds to a section within the ancestor's file. This scheme is used in XML multi-section serialization mode, see Figure 17a. If the node has not such ancestor, it is only used to provide a hierarchical structure for its children, such as in a SCORM course, where nodes corresponding to activities have no associated content.

Attribute `decoratedNumber`

Decorated number of corresponding section, that is, prefix of the title paragraph besides the user-supplied title.

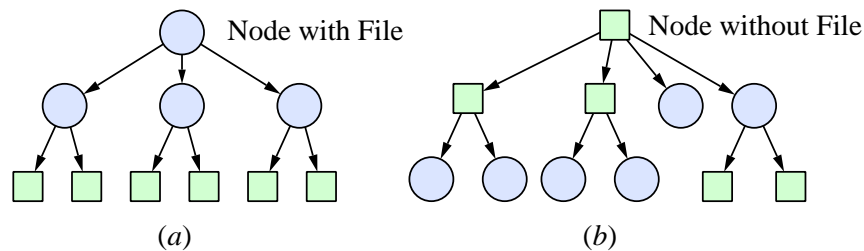


Figure 17 Different Node Structures: (a) Book with Chapters, (b) SCORM Course

Attribute localNumber

Specifies the local number of the corresponding section. For example, section 1.2.4 has local number 4. Note that the text may not be only an integer number but also for example Latin letter or Roman number.

Attribute locale

Specifies the language of the corresponding section.

Attribute show

9 Package idIndex

Element idIndex

Root element of the `idIndex.xml` file. This file specifies list of elements in the associated document which may be used as hyperlink targets using the `hyperLink` element, such as sections, floatings or `displayedMaths`.

Content: `entry(idIndex)`

Element entry(idIndex)

One entry in the `idIndex.xml` file. Holds information about one associated `contentElement`

Parents: `idIndex`

Attributes: `type(idIndexEntry)` `outerId` `innerId` `id` `localNumber` `number(idIndexEntry)` `decorated-Number` `targetNumber` `name(idIndexEntry)` `targetNumberKey` `typeName` `sectionId` `title(deprecated)`

Mixed Content: `(inline | paragraphFragment)*`

Attribute type(idIndexEntry)

Attribute number(idIndexEntry)

Attribute targetNumber

Attribute name(idIndexEntry)

Attribute targetNumberKey

Attribute typeName

Attribute sectionId

10 Package index

Element userIndex

A container corresponding to all definitions of the indexEntries of a document.

Parents: document

Content: indexEntry*

Element indexEntry

One record of the index section of a document, defining one concept. The record may consist of one or more alternative entries defined using the element child element. Optionally, a glossary definition may be provided in a glossary element.

Parents: userIndex

Attributes: id

Content: currentMaster? lastMaster? element+ glossary? target(indexEntry)*

Element element

Defines one of alternative names for concept defined in enclosing indexEntry. The name may consist of several components, from the most important to the less important ones, such as graph – connected. Each component is defined by separate paragraph in p(indexEntryElement).

Parents: indexEntry lastMaster currentMaster

Content: p(indexEntryElement)+

Element p(indexEntryElement)

Defines one component of an index entry element. Since it is of type p, it may contain arbitrary inline material.

Parents: element

Mixed Content: (*inline* | *paragraphFragment*)*

Element glossary

Glossary definition associated with an indexEntry.

Parents: indexEntry

Content: *block**

Element indexEntryRef

This is a reference from styleElement to indexEntry. This link ensures that the referencing style element is mentioned in the generated index section. Note that there should be a corresponding reverse link in the target indexEntry element, defined using the target(indexEntry) element.

Parents: *styleElement*

Attributes: id

Element glossaryRef

Reference from styleElement to indexEntry, similar to indexEntryRef. While the indexEntryRef association is not visible in the document, the glossaryRef association provides text corresponding to the primary indexEntry textual representation. This may be used, for example, to create hyperlinks among glossary definitions.

Parents: *paragraphFragment p*

Attributes: class id userId language

Content: style? indexEntryRef?

Element target(indexEntry)

Specification of one target of the enclosing indexEntry. The target provides three identifiers: the identifier of the actual target id, the identifier of its innermost enclosing section innerId, and the identifier of its innermost section having file. As a content, the element contains paragraph material corresponding to the fragment in the neighborhood of the target element. The actual target element is enclosed in the `` element. Note that each target instance has corresponding indexEntryRef instance.

Parents: indexEntry

Attributes: id outerId innerId title(indexEntryTarget)

Mixed Content: (*inline | paragraphFragment*)*

Element lastMaster

For localized document, specifies the associated master index entry, corresponding to the current translation of this entry. If this entry is up to date, the last master is equal to the currentMaster, and thus the lastMaster element is omitted. If the two fields differ, it indicates that this translation is out of date and needs to be updated. The difference of the current master and the last master may be displayed to the user as a hint for translation.

Parents: indexEntry

Content: element+

Element currentMaster

For localized document, specifies the associated master index entry, in its last known version. This field is updated each time the localized document is updated with the master.

Parents: indexEntry

Content: element+

Attribute outerId

Attribute innerId

11 Package bibliography

Element bibliography

Parents: document

Content: item(bibliography)*

Element item(bibliography)

Parents: bibliography group(bibliography)

Attributes: title type(bibliography) id userId

Content: value(bibliographyItem)*

Element group(bibliography)

Content: item(bibliography)

Element value(bibliographyItem)

Parents: item(bibliography)

Attributes: key(bibliographyValue)

Mixed Content: (*inline* | *paragraphFragment*)*

Attribute type(bibliography)

Attribute key(bibliographyValue)

12 Package deprecated

Element note

Parents: document

Attributes: hasNumber class id userId language

Content: style? indexEntryRef? title? *block** section*

Element sum

Parents: *mathContainer*

Attributes: class id userId language

Content: style? indexEntryRef? body sub sup

Element math(displayed)

The math element was used to denote body of a displayedMath. Now the row element should be used instead.

Parents: displayedMath

Content: *mathBox**

Element title(deprecated)

Attribute horizontalAlignment

Attribute numbered

Attribute valign

Attribute verticalAlignment

Attribute width(box)

Attribute title(indexEntryTarget)

Attribute title(deprecated)

13 Index

-
- alt – attribute in paragraph 14
 - angle – attribute in drawing 34
 - animation – element in drawing 25
 - arcTo – element in drawing 26
 - array – element in math 19
 - aspect – attribute in drawing 34
 - autoLayout – attribute in drawing 32
 - baseAngle – attribute in drawing 33
 - basedOn – attribute in style 5
 - b – element in paragraph 12
 - bezier – element in drawing 27
 - bezierTo – element in drawing 26
 - bibliography – element in bibliography 39
 - bibPage – element in block 8
 - bitmap(drawing) – element in drawing 28
 - bitmap – element in drawing 30
 - blockContainer – element in block 6
 - block – element in block 6
 - blockParent – element in block 6
 - body – element in math 17
 - bottom – attribute in math 23
 - bottom – element in math 18
 - box – element in drawing 27
 - br – element in paragraph 14
 - c1x – attribute in drawing 32
 - c1y – attribute in drawing 32
 - c2x – attribute in drawing 32
 - c2y – attribute in drawing 32
 - cap(capitalization) – attribute in paragraph 15
 - cite – element in paragraph 14
 - class – attribute in core 3
 - closed – attribute in drawing 31
 - code – element in paragraph 12
 - colspan – attribute in block 11
 - comp – element in math 18
 - compiled – element in drawing 30
 - const – element in math 20
 - contentElement – element in core 2
 - currentMaster – element in index 38
 - customBox – element in drawing 30
 - cx – attribute in drawing 31
 - cy – attribute in drawing 32
 - data – attribute in drawing 35
 - dd – element in block 11
 - decoratedNumber – attribute in toc 35
 - den – element in math 17
 - dim – element in math 20
 - displayedMath – element in math 16
 - dl – element in block 10
 - document – element in document 3
 - documentStyles – element in document 4
 - dom – element in math 21
 - drawingContainer – element in drawing 24
 - drawingElement – element in drawing 24
 - drawing – element in drawing 25
 - dt – element in block 10
 - element – element in index 37
 - embeddedFiles – element in document 3
 - end – attribute in math 22
 - entry(idIndex) – element in idIndex 36

- explicitTitle – element in document 4
- extension1 – element in drawing 29
- extension2 – element in drawing 29
- fence – element in math 19
- file(embeddedFiles) – element in document 4
- fill – attribute in block 11
- filled – attribute in drawing 31
- floating – element in block 7
- frac – element in math 17
- from – attribute in drawing 32
- from – element in drawing 29
- fun – element in math 20
- geometry – element in drawing 23
- glossary – element in index 38
- glossaryPage – element in block 8
- glossaryRef – element in index 38
- glue – element in math 16
- group(bibliography) – element in bibliography 39
- group(block) – element in block 8
- hasFile – attribute in toc 35
- hasNumber – attribute in core 3
- h – attribute in drawing 31
- height(bitmap) – attribute in core 3
- height(sourceInfo) – attribute in drawing 35
- horizontalAlignment – attribute in deprecated 40
- hr – element in block 7
- hspace – element in math 17
- Huge – element in paragraph 13
- hyperLink – element in paragraph 13
- i – attribute in paragraph 15
- icon – element in paragraph 14
- id – attribute in core 2
- idIndex – element in idIndex 36
- i – element in paragraph 12
- includeTargetName – attribute in paragraph 15
- indexEntry – element in index 37
- indexEntryRef – element in index 38
- indexPage – element in block 8
- inheritable – attribute in style 5
- inlineBitmap – element in paragraph 14
- inline – element in paragraph 11
- innerId – attribute in index 39
- item(bibliography) – element in bibliography 39
- key(bibliographyValue) – attribute in bibliography 39
- language – attribute in core 3
- Large – element in paragraph 13
- lastMaster – element in index 38
- length – attribute in drawing 33
- level – attribute in document 4
- li – element in block 10
- linDim – element in drawing 28
- lineTo – element in drawing 25
- link – element in drawing 27
- list – element in block 10
- locale – attribute in toc 36
- localNumber – attribute in toc 36
- mat – element in math 20
- math(displayed) – element in deprecated 40
- mathBox – element in math 15
- mathContainer – element in math 15

- math – element in math 15
- mathParentBox – element in math 15
- mathSf – element in math 21
- mathText – element in math 15
- morph – attribute in paragraph 14
- mpar – element in math 19
- mtext – element in math 21
- mx – attribute in drawing 32
- my – attribute in drawing 32
- name(file) – attribute in document 4
- name(hspace) – attribute in math 22
- name(idIndexEntry) – attribute in idIndex 37
- name(sourceInfo) – attribute in drawing 35
- name(style) – attribute in style 5
- negative – attribute in math 22
- node – element in toc 35
- note – element in deprecated 39
- not – element in math 19
- num(number) – element in math 20
- num(numerator) – element in math 17
- number(idIndexEntry) – attribute in idIndex 36
- numbered – attribute in deprecated 40
- numbered – element in core 2
- numberPerRow – attribute in math 21
- ol – element in block 10
- op – element in math 16
- other – element in core 2
- outerId – attribute in index 39
- over – element in math 19
- p(block) – element in block 7
- p(displayedMath) – element in math 16
- p(indexEntryElement) – element in index 37
- page – element in block 6
- paragraphFragment – element in paragraph 12
- p – element in paragraph 11
- position – element in drawing 29
- pre – element in block 7
- quadTo – element in drawing 26
- rectangularPoint – element in drawing 27
- refId – attribute in paragraph 14
- repeat – attribute in math 22
- root – element in math 18
- rotate – element in drawing 24
- row – element in math 16
- rowspan – attribute in block 11
- rule – element in drawing 25
- scale – attribute in drawing 35
- scale – element in drawing 24
- script – element in math 17
- section – element in block 6
- sectionId – attribute in idIndex 37
- sf – element in paragraph 13
- shape – element in drawing 25
- shapeSegment – element in drawing 25
- shear – element in drawing 24
- show – attribute in toc 36
- showInToc – attribute in block 11
- sideways – element in block 8
- size(fence) – attribute in math 22
- snap – element in drawing 30

- source – element in drawing 30
- sourceInfo – element in drawing 28
- span – element in paragraph 12
- sqrt – element in math 18
- stackrel – element in math 18
- start – attribute in math 22
- start – element in drawing 25
- strike – element in paragraph 13
- styleElement – element in core 2
- style – element in style 5
- styleFile – element in style 4
- styles – element in style 4
- sub – element in math 17
- sum – element in deprecated 40
- sup – element in math 18
- sx – attribute in drawing 31
- sy – attribute in drawing 31
- sym – element in math 17
- tabBox – element in drawing 28
- tab – element in math 16
- table – element in block 9
- target(indexEntry) – element in index 38
- targetNumber – attribute in idIndex 36
- targetNumberKey – attribute in idIndex 37
- td(array) – element in math 20
- td(table) – element in block 9
- td – element in block 9
- text – attribute in math 21
- theta – attribute in drawing 31
- tiny – element in paragraph 13
- title(deprecated) – attribute in deprecated 40
- title(deprecated) – element in deprecated 40
- title(indexEntryTarget) – attribute in deprecated 40
- title – attribute in core 3
- titled – element in core 2
- title – element in block 8
- to – attribute in drawing 32
- toc – element in block 7
- todo – element in block 10
- to – element in drawing 29
- top – attribute in math 23
- top – element in math 18
- tr(array) – element in math 19
- tr(table) – element in block 9
- transform – element in drawing 23
- transformOperation – element in drawing 23
- translate – element in drawing 24
- tr – element in block 9
- tt – element in paragraph 12
- type(bibliography) – attribute in bibliography 39
- type(comp) – attribute in math 22
- type(icon) – attribute in paragraph 14
- type(idIndexEntry) – attribute in idIndex 36
- type(linDimAlignment) – attribute in drawing 34
- type(link) – attribute in drawing 33
- type(over) – attribute in math 23
- type(shapeSegment) – attribute in drawing 31
- type(snap) – attribute in drawing 34
- type(sourceInfo) – attribute in drawing 34
- type(styleValue) – attribute in style 5

-
- typeName – attribute in idIndex 37
- u – element in paragraph 12
- ul – element in block 10
- uri – attribute in paragraph 14
- uriRef – element in paragraph 13
- userId – attribute in core 2
- userIndex – element in index 37
- valign – attribute in deprecated 40
- value(bibliographyItem) – element in bibliography 39
- value(dimExtension) – attribute in drawing 34
- value(style) – attribute in style 6
- value(style) – element in style 5
- var – element in math 20
- vec – element in math 20
- verticalAlignment – attribute in deprecated 40
- verticalFence – element in math 18
- viewStyles – element in document 4
- w – attribute in drawing 31
- width(bitmap) – attribute in core 3
- width(box) – attribute in deprecated 40
- width(sourceInfo) – attribute in drawing 35
- x1 – attribute in drawing 32
- x2 – attribute in drawing 33
- x – attribute in drawing 30
- y1 – attribute in drawing 33
- y2 – attribute in drawing 33
- y – attribute in drawing 31