

The KDE DocBook Authors guide

Lauri Watts

Revision 1.00.00 (2004-08-22)

Copyright © 2000, 2001, 2002, 2003 Lauri Watts

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in [the section entitled "GNU Free Documentation License"](#).

A quick reference guide to the current KDE DocBook markup standards

Table of Contents

- [1. General KDE markup style guide](#)
- [2. Purpose of this document](#)
 - [Other reference material](#)
- [3. The Prologue](#)
 - [book and the bookinfo section](#)
- [4. Chapters and Sections](#)
- [5. The linking elements](#)
- [6. Lists](#)
- [7. Tables](#)
- [8. The GUI elements, menus, toolbars and shortcuts.](#)
- [9. Describing actions and commands](#)
- [10. Questions and Answers](#)
- [11. Images and Examples](#)
- [12. General markup \(not covered elsewhere\)](#)
- [13. Admonitions: Tips, hints, and Warnings.](#)
- [14. The synopsis elements](#)
- [15. Markup for programming](#)
- [16. Making Callouts](#)
- [17. References, indexes, and glossaries](#)
 - [Making a glossary](#)
 - [Making an Index](#)
 - [Other Reference Sections](#)
- [18. Tags we do not use](#)
- [19. Alphabetical List of all elements](#)
- [20. Credits and License](#)
- [A. Entities](#)

List of Examples

- 1.1. [Managing translatable entities](#)
- 3.1. [Setting up the global ?kappname? entity](#)
- 3.2. [A KDE User Manual Prolog](#)
- 3.3. [The bookinfo section from the KDE template](#)
- 6.1. [A Segmented List](#)
- 7.1. [An <informaltable> template](#)
- 7.2. [A <table> template](#)
- 8.1. [An example from a menu reference entry](#)
- 10.1. [<qandaset> Template](#)
- 11.1. [A screenshot example](#)
- 14.1. [How to markup a command synopsis](#)
- 14.2. [How to markup a function synopsis](#)
- 16.1. [Marking up callouts with <screenco>.](#)
- 16.2. [Marking up callouts by embedding directly in text](#)
- 17.1. [How to markup a glossary](#)
- 17.2. [Index](#)

Chapter 1. General KDE markup style guide

- Format for readability, and content, not for a formatted document.

It is not your job or responsibility to make sure the final documentation looks good. If you use appropriate markup tags for the content of your documentation, the processing tools will ensure your document looks good. Do not substitute an inappropriate DocBook XML tag because you do not like the ?look? of the correct tag.

You should use white space to make the DocBook source more readable to the writer. Please do not indent unless it is absolutely necessary.

- Do what you can to ensure you turn in a *valid* DocBook file. The reviewers will correct any DocBook errors you create, but please try to reduce errors by checking your work before it is turned in. If you have the KDE tools installed, you can use the command **checkXML index.docbook** to check for syntax errors. No result from **checkXML** is a good result - it means there are no problems.
- Non-English words should be tagged with `<foreignphrase lang="de">Wort</foreignphrase>`.
- Underlining and CAPITALIZING entire words are leftovers from the days of typewriters. They are no longer appropriate for today's documents.
- Do not use quotation marks in your documentation. If you want a word to appear within quotation marks, simply enclose it between `quote` tags.

This software is provided `<quote>as is</quote>`.

- There are three different ‘dashes’ that are commonly found in documentation.
 - The hyphen combines two or more words into one. For example, ‘mother-in-law’. The hyphen can be entered directly from the keyboard.
 - The en-dash is used to separate numbers/dates/etc.. For example, ‘Sections 1-3 review basic concepts’. The en-dash can be encoded using `&ndash`.
 - The em-dash is used to separate sentences, or to show that something is missing. This is rarely used in technical documentation, but it can be used to show that one sentence is interrupted by another. The em-dash can be encoded using `&mdash`.
- When trying to decide between an ordered and unordered list, simply ask yourself the following question: ‘Does the order of the listed items matter?’ or ‘If I change the order of the listed items, does that change the meaning of the list?’. If you answer ‘No’ to either question, then an unordered list is likely the logical choice.
- All `<chapter>` and `<sectN>` tags must have an id. The id must be in all lower case, and with dashes separating words. For example, `<sect1 id="how-to-obtain-kapp">`.
- All elements must have a full closing tag unless they are empty elements. Empty elements must still be closed with a `/`.

Incorrect: `<para>Blah blah/` or `</para>Blah blah</>`

Correct: `<para>Blah blah</para>`

- No attribute minimization.

Incorrect: `attribute=value` or `attribute='value'`

Correct: `attribute="value"`

- All entities must end with a semi-colon:

Incorrect: `%parameterentity` or `&generalentity`

Correct: `%parameterentity;` or `&generalentity;`

- Element GIs (the first word in a tag) must be written in lower case only.

Incorrect: `<MediaObject>` or `<MEDIAOBJECT>`

Correct: `<mediaobject>`

Entities are also case sensitive, and will result in validation errors if the case is wrong.

- Specify date and application’s version in the format:

`<date>2000-12-31</date>`

`<releaseinfo>1.02.03</releaseinfo>`

The `<date>` is the date of the last update. The `<releaseinfo>` always matches the version number of the application that is described in the documentation (if any). A translated version of a documentation always has the same `<date>` and `<releaseinfo>` as the English original. Please respect this, it is the only way to manage efficiently both the writing and the translation processes.

- The list of entities for applications is maintained centrally. Entity names are the application name completely in lower case. In case the name you need does not exist yet, send a mail to `<kde-docbook@kde.org>` to have it added. You may add it in the prologue for validation purposes (in case it's new), but don't forget to remove it when you submit the document, because there should not be any 'extra' entities defined in the document prologue.
- For language-independent entities, use `kdelibs/kdoctools/customization/entities/general.entities` and for language-specific entities, use `kdelibs/kdoctools/customization/lang/user.entities`. Try to avoid clashes with existing KDE entities.
- The `en/user.entities` file should be updated keeping in mind that translation must be possible. Here is an example of how this translation can be managed:

Example 1.1. Managing translatable entities

`&LMB;` is an entity which stands for 'Left Mouse Button'

When translating to French for example, do not translate only the entity contents, please also translate the entity name to `&BGS;` (or `&bgs;`), to reflect the change in the initials:

```
<!ENTITY "LMB" "left mouse button"> becomes
<!ENTITY "BGS" "bouton gauche de la souris">
```

Languages that decline nouns like German and Russian can use something like the following:

```
<!ENTITY "LMB" "linke Maustaste">
<!ENTITY "LMBn" "linken Maustaste">
```

- If you feel that some elements don't make fine enough a distinction, feel free to use the attribute `role` (but please tell the DocBook team, as otherwise you may find your document to be suddenly invalid).
- Use `<qandaset>` for FAQs, not an `<itemizedlist>`. Please split up a FAQ into several chapters or sections if it gets big. The HTML files get too big otherwise, which the users may not like.
- Abbreviations and acronyms should be marked up as well.

Use the DocBook tags `<abbrev>` and `<acronym>` respectively.

Please keep them apart: acronyms are things like GUI, KDE, GPL, while abbreviations are things like etc., i.e., e.g..

There are entities for the most common ones.

- Use `<glossterm>` or `<firstterm>` each time you introduce a technically significant new word.
- Keep in mind that the `??` sign is introduced by the shell, and is not part of an environment variable's name:

`ls -l $KDEDIR` is marked up as

```
<userinput><command>ls</command>
<option>-l</option>
<parameter>${envar}KDEDIR</envar></parameter>
</userinput>
```

`export $KDEDIR=/usr/local/kde` is marked up as:

```
<userinput>
<command>export</command>
<parameter>${envar}KDEDIR</envar>=<filename>
/usr/local/kde</filename></parameter></userinput>
```

- Only use `<ulink>` for URL's and not for files, unlike `<A>` in HTML. Don't use it for email addresses either, they have their own element, `<email>`.
- The elements `<beginpage>` and `<bridgehead>` are disallowed and have been removed from the KDE customized DTD. (They are not meant for new technical documentation.) `<revisionhistory>` has been removed also: we are using SVN already.

Chapter 2. Purpose of this document

Table of Contents

Other reference material

The purpose of this document is to describe how markup has been standardized within KDE documentation only.

This document is *not* to be considered more authoritative than the DocBook documentation, including the O'Reilly Duck book. However, there are places where the KDE DTD is more restrictive than, or just differs from, the OASIS DTD, and these are noted in this document. In these cases, follow the instructions here.

Please read and make use of the other documentation available to you, which is much more comprehensive. This document is not intended to be more than a quick reference for KDE authors, to clarify how the DocBook XML elements are used within the KDE Documentation.

Other reference material

Please take a look at the following reference material, rather than relying on this document to answer all your questions.

The Duck book

The complete DocBook SGML (and now XML) reference. Available as a download in several formats, so you can keep a copy on your hard drive for reference. Also available for sale in hard copy - if you see yourself doing a lot of DocBook Authoring, you definitely ought to consider buying it.

The Crash Course to Docbook

A non-KDE specific crash course to marking up documentation. This is the starting point for all KDE documents, including the markup issues discussed here. Note that the current version is written for SGML, but the concepts are still correct for XML.

The KDE Documentation Template

Covers many things not mentioned here, including required and optional chapters, the preferred way to mark up the prologue and bookinfo sections, and how to deal with licensing and credits. It can be found in `kdelibs/kdoctools/template.docbook` in CVS.

DocBook-XML (in German)

A very nice book, in German only unfortunately, but comes highly recommended.

Chapter 3. The Prologue

Table of Contents

book and the bookinfo section

```
<?xml version="1.0" ?>
<!DOCTYPE book PUBLIC "-//KDE//DTD DocBook XML V4.2-Based Variant V1.1//EN" "dtd/kdex.dtd" [
  <!-- Define an entity for your application if it is not part of KDE
        CVS -->
  <!ENTITY kmyapplication "<application>KMyApp</application>">
  <!ENTITY kappname "&kmyapplication;"><!-- replace kmyapplication here
                                do *not* replace kappname-->
  <!ENTITY package "kde-module"><!-- kdebase, kdedadmin, etc. Leave
                                this unchanged if your
                                application is not maintained in KDE CVS -->

  <!ENTITY % addindex "IGNORE">
  <!ENTITY % English "INCLUDE"> <!-- ONLY If you are writing non-English
                                original documentation, change
                                the language here -->
]>
```

In general, this needs minimal changing from the template. The items you *must* change are the entities `?kappname?`, `?package?`, and `?English?`.

The entity `?kappname?` looks like it's redundant (as the comment in the template notes), but it is important. This allows us to use one global text in all documents, and still refer to the specific application by its correct name. So it should be changed to refer to this new entity, but this time you should only change the part in quotes (`?&kmyapplication;?`) as follow:

Example 3.1. Setting up the global `?kappname?` entity

```
From:
<!ENTITY kappname "&kmyapplication;" -- this only *seems* redundant -->
To:
<!ENTITY kappname "&kate;" -- this only *seems* redundant -->
```

In short: change any occurrence of `?kmyapplication?` to the real name of your application. Do *not* use `?kappname?` or `?kapp?` directly in a document yourself.

The entity `?package?` is used similarly. It allows us to insert a single piece of `?boilerplate?` text into every document, and have the correct package name inserted when the document is compiled. Use the cvs module name, in lower case, e.g. `?kdeedu?` or `?kdebase?`.

The entity `%addindex;` is a toggle. If set to `?INCLUDE?` a document index will be automatically generated. It is normally set instead to `?IGNORE?`, and should not be changed unless you really do want to generate an index. You can find out more about indexes in [Chapter 17, *References, indexes, and glossaries*](#).

Example 3.2. A KDE User Manual Prolog

Here is an example of a completely set up prolog, as it normally looks. This is the prolog from the AMOR documentation

```
<?xml version="1.0" ?>
<!DOCTYPE book PUBLIC "-//KDE//DTD DocBook XML V4.2-Based Variant V1.1//EN" "dtd/kdex.dtd" [
  <!ENTITY kappname "&amor;">
  <!ENTITY package "kdetoys">
  <!ENTITY % addindex "IGNORE">
  <!ENTITY % English "INCLUDE">
]>
```

Note

The entity `?English?` should be changed to reflect your language, if you are either writing original documentation in another language, or you are translating a document. For KDE the original documentation should always be in English, so you should not need to change this when writing. For informational purposes, the currently supported languages are:

- Afrikaans

- British-English
- Bulgarian
- Catalan
- Czech
- Danish
- German
- Greek
- English
- Spanish
- Estonian
- Finnish
- Faroese
- French
- Hebrew
- Hungarian
- Indonesian
- Italian
- Japanese
- Dutch
- Norwegian (Note, this is only for compatibility, either Norwegian-Bokmal or Norwegian-Nynorsk should be used in preference.)
- Norwegian-Bokmal
- Norwegian-Nynorsk
- Polish
- Portuguese

- Brazilian-Portuguese
- Romanian
- Russian
- Slovak
- Slovenian
- Serbian
- Swedish
- Turkish
- Ukrainian
- Walloon
- Xhosa
- Continental-Chinese
- Traditional-Chinese

book and the bookinfo section

The `bookinfo` section is most easily prepared by copying the KDE template.

```
<book lang="&language">
```

Contains the entire document. Most important thing to remember is the `lang` attribute, which must contain exactly `&language;`, and must not be changed. To set the language for the document, change the entity as described in the [prologue](#) section.

```
<bookinfo>
```

Wraps the `?meta?` information ? information about the document, not about the application it is documenting. Required in KDE documentation. No attributes.

```
<authorgroup>
```

Wraps the author information, and may also contain `<othercredit>` information. Required in KDE documentation. No attributes.

```
<author>
```

Required element in the header section of all KDE documentation. Use this element *only* for the author(s) of the document. Other contributors (developers, translators, and so on) should be credited in the <othercredit> section. No attributes.

<personname>

Used to wrap a person's name. You can use this directly in the text as well, but here it should be used to contain each author or contributor name.

<firstname>

The contributor's first name.

<othername>

If the author normally uses more than a first and surname, you can add further names here.

<surname>

The author's surname.

<email>

An email address for the maintainer of the document is required for KDE documentation. You do not have to use your primary private address, and you may be able to arrange for someone else (the developer perhaps) to receive the email regarding the document. In any case, there must be an address for users and translators to contact regarding errors and document bugs.

Note

In previous versions of DocBook, <email> could not be used directly inside <author>. Since DocBook XML V 4.2 (used by KDE for documents after KDE 3.1.x), this is possible, which simplifies this markup considerably.

In other contexts in the document, <email> is used to contain any email address, and is not used inside the address element.

<othercredit role="">

Similar to author, this is a wrapper around information describing other contributors to the document. Include here the contributor's name and email address as you do for the author. See the template for more details.

The role attribute is required, and can contain any one of the following:

- Translator

- Developer
- Reviewer
- Graphist
- Musician

The `othercredit` element also includes the `contrib` element.

`<contrib>`

The role this contributor played in the document or application preparation. This could contain something like:

- Developer
- *Deutsche Übersetzung*
- Reviewer
- *Traduction française*

`<corpauthor>`

This is used in very specific circumstances, where an organization (e.g. "The KDE Team") is being credited with authorship of a document. Authors writing about applications should not use this and should credit themselves. If you do find a need to use this, please be sure to include a maintainer's name and email address in the credits chapter of the document.

`<copyright>`

This is a wrapper for copyright information. `copyright` must contain these elements:

`<year>`

Add one `year` element for each year in which the document was changed or added to. Don't put more than one year in each tag, rather add more `year` elements, and use the 4 digit "YYYY" format.

`<holder>`

The usual full name of the copyright holder(s). If there is more than one copyright holder (the document was previously maintained by another person, or is written collaboratively), then add more `copyright` sections, rather than trying to fit multiple names in the one section.

Copyright is automatically held by the author of the document, but the `copyright` element is still required for all KDE documentation. None of the elements contained have any attributes.

Please do *not* add more names or years to existing <holder> or <year> elements. Add more, if they are required, or have multiple copyright sections.

<legalnotice>

This contains, of course, a legal notice. This is absolutely required for any KDE document. In the context of this section, it should contain the &FDLNotice; entity, which inserts some information into the document about the document's license (and *not* the license of the application you are describing.)

<date>

The date is very important. It is used not only by scripts for automatic processing of documentation, but is also central to revision control and co-ordination of translations. You must change the date if you have changed the *original* document, and you must *not* change the date if you are a translator. The format of the date is very important. It *must* be in the ISO, with *literal* delimiters, in the form *yyyy-mm-dd*?. Please be extremely careful about this, and triple check it before you send in the document.

<releaseinfo>

This should match exactly the version of the *application* you are documenting. It should normally conform to the format X.x.xx (where X is a major version number and x are minor version numbers, however, you no longer have to pad the content to this length. That is to say, if the application has released version *1.4*?, you may write <releaseinfo>1.4</releaseinfo>, and you do not need to make it <releaseinfo>1.04.00</releaseinfo>

This is *not* the version of the document. There are no attributes, and this element is required in KDE documentation.

<abstract>

In KDE Documentation, the abstract is required. It should be a short one- or two-sentence summary of the document. The abstract is not the place to put version or contact information, but it should say something about the application and its purpose. For example *KFoo is a small fast network enabled foo generator, suitable for both beginner and advanced foo users.*?

The abstract is your chance to sum up the application in a small paragraph ? in KHelpCenter it shows up on the first page as your document is selected, and the abstract frequently shows up in the summary of your document in web searches. A short overview of the application you are writing about is very valuable in this situation, *This is the KFoo handbook and describes KFoo 1.2.*? on its own, is not.

<keywordset>

A wrapper for a set of keywords suitable for search engines. Required for KDE Documentation, and there are no attributes. The keywordset should contain several <keyword>s.

<keyword>

Add one <keyword> inside the <keywordset> for each search term. You must include at a minimum the terms ?KDE?, the name of the application you are documenting, and the name of the package it is found in, for example ?kdegames?. The keywords should be in order from most general first (that is, KDE) through less general, to the most specific. Add two or three more relevant words that people might search with, e.g., for the application KWrite you might add ?editor? and ?text?. This is required for KDE Documentation, and there are no attributes.

<!-- TRANS:ROLES_OF_TRANSLATORS -->

This line is specific to KDE documentation. Although it's a comment, it is *absolutely* required in documents. It is used by the translation system as a placeholder for the translation teams to add their own role info. Translators should add more `othercredit` sections here as appropriate.

Example 3.3. The bookinfo section from the KDE template

```
<bookinfo>
<title>The &kmyapplication; Handbook</title>

<authorgroup>
<author>
<!-- This is just put in as an example. For real documentation, please
define a general entity in entities/contributor.entities, e.g.
<!ENTITY George.N.Ugnacious "<personname><firstname>George</firstname><othername>N.</othername><surname>Ugnacious</surname></personname>">
<!ENTITY George.N.Ugnacious.mail "<email>gnu@kde.org</email>">
and use '&George.N.Ugnacious; &George.N.Ugnacious.mail;' in the author element.
-->
<personname>
<firstname>George</firstname>
<othername>N.</othername>
<surname>Ugnacious</surname>
</personname>
<email>gnu@kde.org</email>
</author>
</authorgroup>

<!-- TRANS:ROLES_OF_TRANSLATORS -->

<copyright>
<year>2002</year>
<holder>George N. Ugnacious</holder>
</copyright>
<!-- Translators: put here the copyright notice of the translation -->
<!-- Put here the FDL notice. Read the explanation in fdl-notice.docbook
and in the FDL itself on how to use it. -->
<legalnotice>&FDLNotice;</legalnotice>

<!-- Date and version information of the documentation
Don't forget to include this last date and this last revision number, we
need them for translation coordination !
Please respect the format of the date (YYYY-MM-DD) and of the version
(V.MM.LL), it could be used by automation scripts.
Do NOT change these in the translation. -->
<date>2003-01-10</date>
<releaseinfo>1.1.</releaseinfo>

<!-- Abstract about this handbook -->

<abstract>
<para>
&kmyapplication; is an application specially designed to do nothing you would
ever want.
</para>
</abstract>

<!-- This is a set of Keywords for indexing by search engines.
Please at least include KDE, the KDE package it is in, the name
of your application, and a few relevant keywords. -->

<keywordset>
<keyword>KDE</keyword>
<keyword>kdeutils</keyword>
<keyword>Kapp</keyword>
<keyword>nothing</keyword>
```

```
<keyword>nothing else</keyword>
</keywordset>

</bookinfo>
```

Chapter 4. Chapters and Sections

```
<chapter id="">
```

Use chapters to break up the document into smaller chunks. A chapter break should occur when a major subject change happens. Use sections within the chapter when the subject changes, but you are still discussing a particular aspect of a larger subject.

For example, going from discussing how to use the application, to how to configure the application would be worthy of a new chapter. Moving from discussing how to specifically configure the application on SuSE, to how to specifically configure the application on Red Hat®, would be a new section in a larger *Configuration* chapter.

Chapters must have an `id`. This is the only attribute used in KDE documentation. For KDE Documents, this `id` must be in lower case, and with a hyphen (-) to separate words. Please don't use spaces, underscores, or run the words together. For HTML generation, the chapter `id` and most `<sect1>` `id`'s are used to name the separate HTML pages, so take care to make them sensible and descriptive. For translators, these `id`'s should be translated, but you will need to take care to also translate references to the `id`'s in `<link>` and `<xref>` elements in other parts of the document.

```
<title>
```

Titles are used in many places, but the most common is the Chapter and Section headings. Make sure to use sensible titles, as these will also be that chapter's (or section's) entry in the table of contents, so people will rely on these to find the part of the document they are interested in.

```
<sect1 id="">, <sect2>, <sect3>, <sect4>, <sect5>
```

Use sections to break chapters up into smaller pieces. Use similar criteria on where to divide them as you would for chapters.

Sections require a `<title>`. Sections are nested according to the number - a `<sect2>` can contain any number of `<sect3>`, which can contain `<sect4>`, but a `<sect2>` can't directly contain a `<sect4>`.

`<sect1>` requires an `id` attribute, and you can use `id`'s on the other section tags if you want to later link directly to them from other parts of the document. `id` is the only attribute used in KDE Documentation.

```
<sect1info>, <sect2info>, <sect3info>, <sect4info>, <sect5info>
```

The section `info` elements are rarely used in KDE Documentation. They are appropriate for documents where some smaller sections are contributed by third parties, or where the document covers multiple applications. The contents are more or less the same as those of the `<bookinfo>` section, although they tend to be briefer.

Please ensure if you use these elements that you add the translation placeholder comments as you do in the prolog.

```
<appendix>
```

The standard installation instructions for all applications are contained in an `<appendix>`, and are normally required for KDE documents. Although the installation instructions as found in the template are reasonably complete, and need no customization for most applications, authors are very strongly encouraged to expand on them. For example, links to web pages, where to find libraries, plugins, screenshots of the application in a particular configuration, or any other information you can think of.

If the application is only distributed with KDE, there is little use in repeating the same installation instructions for every manual. You may leave it out entirely, unless you have further information to add.

For other purposes, appendices are used infrequently in KDE Documentation. An appendix can be found, for example, in the KPPP document, containing such things as Hayes Modem commands. Only use an appendix if you think it's very necessary. In most cases, the information it would contain would be better moved to the main document. In the example of KPPP, this information is vital to a few people, but extremely uninteresting to the majority, so it was placed in an appendix.

Chapter 5. The linking elements

```
<link linkend="">
```

The most common link. Use this to turn a word or phrase into a link to another part of the document. `linkend` is the only attribute we use.

```
<ulink url="">
```

A link that refers to a document using it's URI. Use this for websites and ftp sites, but not for email addresses, which have their own specific tag. Please do *not* use this to link to other documents on the local system.

```
<anchor id="" />
```

Marks a place in the document, which you can use to link to. Note that the `id` attribute on any other element where it is valid, will automatically generate an HTML anchor in generated HTML, so you do not need to duplicate these. Use anchors only when you need to jump into the middle of a longer page, for example, to a particular menu item, or to a particular option in a preference dialog.

Note

`<anchor />` is an empty element, and must be closed with a `/`.

```
<xref linkend="" />
```

A cross reference to another part of the document. Use this when you want to refer to the section without the name. This is one of very few unclosed elements allowed. `linkend` is the only attribute we currently use.

Note

`<xref />` is an empty element, and must be closed with a `/`.

`<email>`

Use this to enclose an email address. Don't add `?mailto:?` to the email address, and don't use `<ulink url=" ">` for email addresses. No attributes required.

Chapter 6. Lists

`<listitem>`

`<listitem>` is the main building block of almost all the lists. It should always contain some other markup, usually a `<para>`

`<orderedlist>`

Use this type of list when the order of the items matters, but they are not a set of steps that are carried out to achieve something. A good example is a list of things in order of importance.

`<itemizedlist>`

Use an itemized list when the order of the items is not important.

`<variablelist>`

A list that has two sections for each entry. Examples: A menu item, and what the menu item does, An action, and its result, or a term and its definition. This is a very common type of list. (Almost this entire document is composed of variable lists.)

`<variablelist>` contains the following elements:

`<varlistentry>`

A `<varlistentry>` is a wrapper around each pair in the variable list.

`<term>`

To reuse the above examples, the `<term>` for each pair would be the menu item you are describing, the action, or the term you are defining. You can use the `id` attribute for this element, which is quite convenient in long lists such as a menu reference, enabling you to link directly to a particular menu item from another part of the document.

`<listitem>`

As described above the `<listitem>` is used inside a `<varlistentry>` to hold the second part of the pair: The result of choosing that menu item, for example, the consequences of an action, or the definition of the term.

`<procedure>`

Use a procedure list when you are listing a sequence of steps which are performed in a particular order.

A procedure contains only one tag:

`<step>`

A step is one of the sequence of events that make up a procedure.

`<substeps>`

A step can contain substeps

`<simplelist>`

A simple list is just that - a simple list, with no formatting required. A simple list can contain only one type of element:

`<member>`

Members of a simple list.

`<segmentedlist>`

A Segmented list is a very particular type of list. Use sparingly, as it's very difficult to get these right, and most content appropriate for a segmented list could just as well fit the table model.

Example 6.1. A Segmented List

```
<segmentedlist>
<segtitle>Name</segtitle>
<segtitle>Occupation</segtitle>
<segtitle>Favorite Food</segtitle>
<seglistitem>
<seg>Tux</seg>
<seg>Linux Mascot</seg>
<seg>Herring</seg>
</seglistitem>
<seglistitem>
<seg>Konqui</seg>
<seg>The KDE Dragon</seg>
<seg>Gnomes</seg>
</seglistitem>
</segmentedlist>
```

Name: Tux
Occupation: Linux Mascot
Favorite Food: Herring
Name: Konqui
Occupation: The KDE Dragon
Favorite Food: Gnomes

The segmented list contains the following elements:

`<segtitle>`

The title each segment will have

`<seglistitem>`

A set of entries in the list

`<seg>`

The contents of the entries in the list. In each `<seglistitem>` there is one `<seg>` for each `<segtitle>`.

Chapter 7. Tables

`<informaltable>`

This is the table type used most in KDE Documentation. Please be very sure that what you are marking up as a table, is actually tabular data, as in many cases a `<variablelist>` is more appropriate. Please do not use any of the presentation attributes to make tables "look nice?". The only attribute currently allowed in KDE Documents is `pgwide`.

An `<informaltable>` must contain a `<tgroup cols=" ">` entry. Informal tables have no specific title, if you wish the table to be titled and to have an entry in the table of contents, you should use `<table>`. Do not use any attributes other than `pgwide` on tables or informal tables for KDE documentation.

`<table>`

A formal table with a title. Tables will have their own separate entry in the table of contents. Other than the addition of a title, they are marked up the same as an `<informaltable>`.

`<tgroup cols=" ">`

A `<tgroup>` is a required element in a table. The `cols` attribute is required, and should be completed with the number of columns the table is to hold. No other attributes used in KDE Documentation.

A `tgroup` must contain a `tbody`

`<tbody>`

A `tbody` is a required element in a table. There are no attributes. The `tbody` contains rows.

`<row>`

A row corresponds directly with the rows of the table. Rows contain `<entry>` tags, one for each column in the table, as specified by the `cols` attribute on the `<tgroup>` tag.

`<entry>`

The entry is the basic building block of a table. Each entry corresponds to one ?data cell? in the table. There must be as many `<entry>` tags in each row as the `cols` attribute on the `<tgroup>` tag. There are no attributes used in KDE Documentation.

`<thead>`

`<thead>` can be used to create a heading row for the table. It must appear before the `tbody` element, and should normally contain one row and as many `entry` elements as the rest of the table.

`<tfoot>`

`<tfoot>` is not currently used in KDE Documentation. If you want to use it, please see the Duck book for information.

Example 7.1. An `<informaltable>` template

```
<informaltable>
<tgroup cols="2">
<tbody>
<row>
<entry></entry>
<entry></entry>
</row>
</tbody>
</tgroup>
</informaltable>
```

Example 7.2. A `<table>` template

```
<table>
<title></title>
<tgroup cols="2">
<tbody>
<row>
<entry></entry>
<entry></entry>
</row>
</tbody>
</tgroup>
</table>
```

Chapter 8. The GUI elements, menus, toolbars and shortcuts.

`<action>`

The result of a user action. This does not need to be a complete sentence, or even more than a single word. For example, `?This button <action>closes the dialog</action>.` The main place you will find this in KDE Documentation is in the Menu and Command reference chapters of the manuals.

`<guibutton>`

The text on a button that you click on. Icons, Radio buttons and check boxes are not considered buttons in this sense.

`<guiicon>`

The name or description of an icon.

`<guilabel>`

The text of anything that is labelled on screen, and isn't a button, icon, menu, or menu item. For example, the name of a dialog box, the name of a tab in that dialog box, and the name of a label by a checkbox.

Take care that the text exactly matches the label on screen. If it has a `:` on the dialog box, put the `:` into your documentation. Match the capitalization. There is a script in the `kde-i18n` module called `check-gui-texts` which you can use to help check that your text matches exactly what is in the application. During translation, the translators can use this script to generate translations from their translations of the GUI itself, but this will only work if the English text matches precisely.

`<guimenu>`

The top level name of a menu (that is, the name you can see on the menu bar when the menu isn't open).

`<guimenuitem>`

The final item you select on the menu, that actually performs an action.

`<guisubmenu>`

A submenu. That is, a menu which has items both above and below it in the hierarchy.

`<keycap>`

A keycap is a key as it is labelled on your keyboard. **Home** is a keycap on a standard English keyboard. **Alt Gr** is a standard key on many European keyboards.

<keycode>

The internal identifier for a key on the keyboard. Used very infrequently, but you may find need for it, for example when describing entries in rc files.

<keysym>

Right arrow is the <keysym> for the <keycap> that looks like ->. Please note this is a KDE specific use of <keysym>, and does not precisely follow the examples in the Duck Book.

<menuchoice>

A menuchoice describes a menu entry. You should use <menuchoice> anywhere you are describing how to reach a menu item. In normal text, there are no particular requirements. In a menu reference, the <menuchoice> should also contain a <shortcut> element describing the keyboard shortcut, and the contents should also be marked up with <accel> as appropriate.

<shortcut>

A key combination that is a shortcut for a menu item. This is *only* used inside <menuchoice> and contains <keycombo> or <keycap> that is defined as the keyboard shortcut in the menu. In the markup, it appears before the actual menu entries inside the <menuchoice> You do not need to describe the shortcut every time the menu item is mentioned in the text, although it may be appropriate to do so on some occasions.

<mousebutton>

The normal name of a mouse button. It will be normally be one of:

- <mousebutton>left</mousebutton> or the entity &LMB ;
- <mousebutton>middle</mousebutton> or the entity &MMB ;
- <mousebutton>right</mousebutton> or the entity &RMB ; .
- <mousebutton>wheel</mousebutton>

Wheel is used only in specific instructions for applications that support it, of course.

Use the entities where possible, they are a lot less typing and are simple to remember (which is why we have provided them.) If you are translating, check with your team leader, as the entities above are *not* translated, but you may have your own language specific ones to use in their place.

<keycombo action=" ">

A keycombo is a sequence or combination of keypresses that are performed together. A keycombo can contain <keycap>, <keysym> or <mousebutton>, or any combination of these, in any order.

It is normal to have them in the order modifier, Alpha-numeric, Mouse. That is, **Ctrl-A**, not **A-Ctrl**, unless pressing **A** then **Ctrl** actually is the shortcut.

Keycombo requires an `action` attribute, describing exactly how the keys (or mouse buttons) are combined. The choices are:

- Click
- Double-Click
- Other
- Press
- Seq
- Simul

You will most likely need to use `Seq` (for a sequence of keys that are pressed one after the other), or `Simul` for a combination of keys that are pressed at the same time.

`<accel>`

The accelerator key that can be used to access a GUI menu without a mouse. This is indicated in the menu by an underlined letter. Although we previously used this in the menu references, we have since decided not to, the maintenance is too high, and it causes an enormous amount of work during translation.

Example 8.1. An example from a menu reference entry

```
<varlistentry>
<term><menuchoice>
<shortcut>
<keycombo action="simul">&Ctrl;
<keycap>C</keycap></keycombo>
</shortcut>
<guimenu>Edit</guimenu>
<guimenuitem>Copy</guimenuitem>
</menuchoice></term>
<listitem><para><action>Copy the selected text</action> to the
clipboard</para></listitem>
</varlistentry>
```

```
<varlistentry>
<term><menuchoice>
<shortcut><keycombo action="simul">&Ctrl;
<keycap>V</keycap>
</keycombo></shortcut>
<guimenu>Edit</guimenu>
<guimenuitem>Paste</guimenuitem>
```

```
</menuchoice></term>
<listitem><para><action>Paste</action> the contents of
the clipboard at the cursor.</para></listitem>
</varlistentry>
```

Please note, this is very complicated markup, and until you have written a few it's very hard to follow, but it does get much easier with practise! Although indenting is discouraged in general, this is one place where you might want to use some indenting and white space to make it clearer while writing, at least when you are beginning. There are also no rules as to when you must start a new line for a new element, so format the markup to suit your own taste while you are writing, if that makes it easier for you to follow.

Chapter 9. Describing actions and commands

```
<replaceable>
```

Use this for placeholder or sample text, that a user would not actually type, but would instead replace with the correct text for their environment. For example, Edit the file `<filename><replaceable>/usr/local/foo/bar</replaceable></filename>`, because it may already be established that `/usr/local` is only the default location of this file, and the user may have it installed to e.g. `/opt/` instead.

```
<application>
```

Use this to mark up the name of any software program mentioned in the text. Don't use this to mark up the actual command issued to execute the application. For example, `<application>Kate</application>` is the name of the editor, but `<command>kate</command>` is the name of the command that starts the Kate application.

Note

All KDE applications, and several non-KDE but very common applications, are provided as entities.

For the KDE applications, using the entities will save you much typing, and will ensure that applications are always referred to with their correct name across all documentation. The entity is always the application's executable name, in lower case, e.g. `&kcontrol;`, `&konqueror;` or `&kmail;`.

For non-KDE applications, one of the major reasons to use the entities is that there are legal implications, so far as we are required to acknowledge trademarks and copyrights held by others outside our organisation. You will find in [Appendix A, *Entities*](#) a list containing a list of the more common non-KDE application entities.

```
<interface>
```

Catch all element for gui interface items that do not have a more specific tag. You can use this to markup things like the `?View pane?` in `KHelpCenter`, or the `?Board?` in `KJumpingCube`.

<userinput>

Any text that the user must type, including commands and data entry.

<screen>

Used to represent the computer screen (usually to represent a terminal or console.) Text contained in <screen> is considered to be literal text ? line breaks and white space are honored and it will be rendered with a mono-spaced font. Don't use screen when what you really want is an example, or an informal example.

<command>

Text the user enters to instruct the computer or an application to do something. **ls -al** is a command (it's also userinput, and has options.) / **join #kde** in an irc client is a command (and again, is userinput.)

Commands are not userinput when you are not expecting the user to actually type them, for example in the sentence ?The output from the **ls** command should show you...?, the text ?ls? is a command, but is not userinput in this context.

Applications not marked up with the <application> tag are also considered commands, for example, **gcc**, **automake** and **autoconf**.

<prompt>

The prompt at which a user types input. For most KDE Documentation, this has been standardised as <prompt>&percent; </prompt> (which is the % character).

<option>

An optional parameter to a command. Since we write about UNIX® platforms, an option on the commandline is almost always indicated by a ?-?, but there are exceptions (e.g., **tar zxvf filename.tar.gz** or **ps ax**, which are marked up as <userinput><command>tar</command> <option>zxvf</option> <replaceable>filename.tar.gz</replaceable></userinput> and <userinput><command>ps</command> <option>ax</option></userinput> respectively.

<envar>

An environment variable. Note that the variable indicator (usually \$ for UNIX®) is not part of the name of the environment variable, so it is correct to do this: \$<envar>KDEDIR</envar>. There are no attributes in use in KDE Documentation.

<errorcode>

A (usually numeric, but not always) error code. SIGSEGV is an errorcode, as is 404 as you might receive when you are web browsing.

`<errorname>`

The actual text of an error message - to reuse the 404 example, the `<errorname>` might be Page not found.

`<errortype>`

The type of error, e.g. fatal or recoverable.

`<filename>`

Use `<filename>` for all occurrences of file names including:

- Directory names ? with the attribute `class="directory"`
- Paths
- File names
- File name placeholders (which should also be tagged with `<replaceable>`)

Do not use `<filename>` for file fragments or extensions (i.e. `*.tgz` which should instead be marked up as `<literal role="extension">`).

`<symbol>`

Symbols are things that are replaced by the computer when they are processed. It's difficult to say when things are a symbol and when they are not - if there is a more specific element to use (e.g. `<envar>` or `<constant>`) then you should use that instead.

Chapter 10. Questions and Answers

`<qandaset>`

A set of questions and answers, suitable for a FAQ. `<qandaset>` must contain `<qandaentry>`.

`<qandaentry>`

Each question and answer pair is a `<qandaentry>`.

`<question>`

The question being asked. It must be inside a `<qandaentry>`, and it must have a matching answer.

`<answer>`

The answer to the matching question in the same `<qandaset>`.

Example 10.1. <qandaset> Template

```
<qandaset>
<qandaentry>
<question>
</question>
<answer>
</answer>
</qandaentry>
</qandaset>
```

Chapter 11. Images and Examples

<screenshot>

Wrapper around screenshots. Use this when you are including a screenshot in your document.

<screeninfo>

Screeninfo is a description of the screenshot. It's common (but not required) to reuse this text in the textobject element, as it saves translation time.

<mediaobject> and <inlinemediaobject>

Use `inlinemediaobject` to insert an inline image (that is, one that is inside a paragraph of text, or is the only item in a table entry). Use `mediaobject` for all other images. If the image is a screenshot, the `mediaobject` should be wrapped with a `screenshot` element. `mediaobjects` contain the following items:

<imageobject>

Imageobject contains information about one specific image. DocBook allows you to add more than one imageobject, in order to provide alternatives if the user is unable to see the preferred image. We don't currently use this functionality in KDE Documentation, but may do at some time in the future.

<imagedata fileref="" format="" />

This element holds the actual image reference. The `fileref` indicates the location of the image. You should always keep images in the same directory as the document itself, so you need only put the filename into the `fileref` attribute. The `format` indicates the type of image you are including. For KDE this should be PNG. Do *not* use gif format images for KDE documents.

This is one of few `?empty?` elements in use in KDE Documentation. This means there is no `</imagedata>`, but you should *always* close the element as shown above, with a final `?/?`.

Keep the images in the same directory as your `index.docbook`, don't create a separate directory to store them in.

`<textobject>`

Encloses the text part of a screenshot, which for KDE Documentation means it contains a `<phrase>` element.

`<phrase>`

A short descriptive phrase about the image contents, this element is contained in the `<textobject>` element.

`<caption>`

If you want the image to have a caption when displayed, you can add this. It's not required for KDE documents, but recommended, especially if there are several images near each other and there could be confusion as to which you are referring in the text.

`<informalexample>`

Use this element to enclose any informal examples you use in your document. There are no attributes. An informal example can contain almost any markup, so feel free to use them liberally. They should generally not be part of a paragraph.

`<example>`

An example is a more formal example, which has a title and an entry in the table of contents. Use sparingly, because having a hundred examples listed in the contents of a 5 page document lessens their usefulness. However, don't hesitate to use when you think it's necessary.

I've used them in this document to make it easy to quickly go to the small `?template?` examples for complex markup, because you can find them directly from the table of contents. Less difficult examples in this document have `<informalexample>` instead. Use your best judgement. As with `<informalexample>`, they can contain almost any markup.

Example 11.1. A screenshot example

```
<screenshot>
<screeninfo>An example image</screeninfo>
<mediaobject>
<imageobject>
<imagedata fileref="example.png" format="PNG" />
</imageobject>
<textobject>
<phrase>An example image</phrase>
</textobject>
</mediaobject>
</screenshot>
```

Chapter 12. General markup (not covered elsewhere)

`<abbrev>`

Abbreviations are shortened forms of longer words.

Abbreviations are not normally pronounced in speech. Examples are e.g. and i.e.. This is a KDE specific distinction, please stick to it.

`<acronym>`

Acronyms are shortened forms of words or phrases, often made up of the initials of the words in a phrase. Acronyms are normally pronounced in speech as well as written. Examples are GUI and KDE. As with `<abbrev>`, this is a KDE specific distinction.

`<attribution>`

If you use `<quote>` or `<blockquote>`, the source of the quote (that is, *who* you are quoting) should be cited with this tag.

`<blockquote>`

Use this when you want to quote a passage of text that should be set off from the main text, for example, an entire paragraph from a book or other source. Use `<quote>` to quote a passage of text that is not to be set off, for example a short sentence or comment from another person. Use both of them as little as you can, there are copyright issues to quoting from other works inside KDE Documentation.

`<emphasis>`

Use this to emphasise text. Don't use it to mark up file names, commands, or anything else. Use it where you might type in all caps in an email, for emphasis of one word or short phrase, and try not to use it too much. Emphasis loses it's power when over used.

`<computeroutput>`

Text the user can see on the computer screen. For example, a listing of a directory as produced after the command `ls` would be `computeroutput`.

`<epigraph>`

A short quote or saying at, sometimes used at the beginning of a chapter as an introduction. Use sparingly, no attributes used by KDE.

`<equation>`

Equation is used if you need to mark up a mathematical equation. You are unlikely to need to use this in KDE Documents.

`<hardware>`

Used when referring to a piece of computer hardware, e.g. Floppy Drive or Monitor.

`<lineannotation>`

A comment, for example in a `<programlisting>`. This is *not* for comments contained in the text, it is for comments by the author (you) *about* the text.

`<literal>`

In KDE Documentation, this is markup of last resort (or ?the least of all evils?) Use it only for things that must be marked up, but have no appropriate tag, and preferably only for the following things (already decided on:)

- `<literal role="extension">*.tar.gz</literal>`

`<literallayout>`

Use very sparingly, when it is absolutely vital that some text is presented exactly as it appears, including white space and line breaks. There is almost always a better tag to use than this (screen and computeroutput together, or even a screenshot).

`<markup>`

Use to wrap markup examples, for text that should be represented literally. Examples are this document, and documents that have HTML markup included literally in them. Other than meta-documentation like this, you probably won't have much need for markup.

`<optional>`

Optional information, usually in user input. Not used to date in KDE Documentation, but it may be appropriate in some circumstances.

`<para>`

A paragraph. This is the most common tag. You do not need to enclose lists, tables, or other markup with `<para>`. Sometimes however, you might want to do so, especially with `<screen>` and some types of lists, when they actually are still part of the current paragraph.

`<quote>`

Use when you are quoting something or someone, inside a sentence. Also use if you want a word or phrase to be ?enclosed in quotes? like this.

`<trademark class=" ">`

Used to denote that a word is a trademark. There is the optional attribute `class` which should contain one of the following, if appropriate:

- `copyright`
- `registered`
- `service`
- `trade`

If there is no `class=" "` attribute, `?trade?` is assumed.

We have provided entities, marked up appropriately, for very commonly met trademarks, including Qt? (`&Qt;`), UNIX® (`&UNIX;`), Linux® (`&Linux;`) and many more.

`<sgmltag>`

An SGML tag. This includes XML and XHTML tags. Use this for marking up individual components, but use `<markup>` when you need to display a block of markup.

`sgmltag` will generate the correct markup characters for you, based on the `class` attribute.

Attribute values available:

- `attvalue`, for the contents of an attribute.
- `attribute`, for attributes.
- `element`, for element names.
- `endtag`, for closing tags (e.g. `</para>`).
- `emptytag`, for tags which are `?empty?`, such as `
` in XHTML.
- `genentity`, for markup up general entities. For example, ` ` in XHTML.
- `numcharref`, to mark up a numbered character reference. ` `, for example, could also be referred to as ` `.
- `parentity`. You are unlikely to need this for any KDE documentation.
- `pi`. Note this is an SGML PI, not an XML one. You are very unlikely to need this for any KDE documentation.
- `xmlpi`. An XML processing instruction, such as

- `starttag`. An opening tag, such as `<para>`. Most of this document is marked up this way.
- `sgmlcomment`.

`<superscript>`

Superscript as in x^2 . Unlikely to be required in most KDE Documentation.

`<msgtext>`

The actual text of an informational message. Use `<errorname>` for error messages.

`<subscript>`

Used to create things like H_2O . Unlikely to be found in most KDE Documents.

`<foreignphrase lang="">`

Use this any time you need to use text in a language different than the main language of the document. This should be rare, but may occur especially in credits information. The `lang` attribute should contain the normal two letter designation of the language. Please be careful with these, the *Country* and *Language* codes are sometimes different, e.g. `?se?` is the country code for Sweden, but the language code is `?sv?`. Using `?uk?` for British English would give you possibly unexpected results, as this is actually the language code for Ukrainian.

Chapter 13. Admonitions: Tips, hints, and Warnings.

Admonitions are set off from the main body of the text. Use these sparingly, as they disturb the flow of the writing, but don't be afraid to use them where necessary. Just make sure they *are* necessary when you do use them.

We have settled on a preliminary order of importance for these elements, which differs from that explained in the Duck Book. Note that this particular order is for KDE Documentation only, and use your own judgement which is the most appropriate element if your situation differs from those outlined.

`<warning>`

Use warning when data loss could occur if you follow the procedure being described.

`<caution>`

A note of caution. Use this for example when the reader may lose easily recovered or replaceable information (e.g. user settings), or when they could cause data loss if they don't correctly follow the procedure being outlined.

`<important>`

When there is no danger of data loss, but you wish to make clear to the reader a consequence that isn't immediately obvious (e.g. when changing the font for one instance of a program also changes the default setting, and this isn't clear from the GUI.)

`<note>`

Information the user should be aware of, but is peripheral to the actual task being described.

`<tip>`

When you're giving a hint to make things easier or more productive for the reader.

`<footnote id=" ">`

Use very sparingly for things that really are footnotes. An example might be to note that the situation being described will be changing at some currently unknown future time. Most footnotes would better be marked up as notes, or tips.

`<footnoteref linkend=" ">`

You can refer to a footnote more than once, by using this element to refer to it's unique id. The footnote does not need to be in the same chapter. Use this very sparingly.

Chapter 14. The synopsis elements

`<cmdsynopsis>`

Example 14.1. How to markup a command synopsis

```
<cmdsynopsis>
<command>more</command>
<group choice="opt"><option>-d</option>
<option>l</option><option>f</option>
<option>p</option><option>c</option>
<option>s</option><option>u</option>
</group>
<arg>-num</arg>
<arg>+/ pattern</arg>
<arg>+ linenum</arg>
<arg rep="repeat"><replaceable>file</replaceable></arg>
</cmdsynopsis>
```

This should generate:

```
more [-dlfpcsu] [-num] [+ pattern] [+ linenum] [file...]
```

There are several very nice examples in the Duck book at www.docbook.org

`<functsynopsis>`

Example 14.2. How to markup a function synopsis

```
<funcsynopsis>
<funcprototype>
<funcdef>void <function>setFile</function></funcdef>
<paramdef>QString <parameter>file</parameter></paramdef>
</funcprototype>
</funcsynopsis>

<funcsynopsis>
<funcprototype>
<funcdef>void <function>setAutoResize</function></funcdef>
<paramdef>bool <parameter><replaceable>val</replaceable></parameter></paramdef>
</funcprototype>
</funcsynopsis>

<funcsynopsis>
<funcprototype>
<funcdef>QString <function>getVideoCodec</function></funcdef><void/>
</funcprototype>
</funcsynopsis>
```

These would generate the following, respectively.

```
void setFile(file);
QString file;
```

```
void setAutoResize(val);
bool val;
```

```
QString getVideoCodec( );
```

A function synopsis can contain the following:

```
<funcprototype>
```

Contains a prototype of the function. It can contain `<void>`, `<varargs>`, `<paramdef>` or most commonly, a `<funcdef>` which actually defines the function.

```
<funcdef>
```

A function and its return type.

```
<funcparams>
```

Contains the list of parameters for the function.

```
<paramdef>
```

Information about the parameters of a function.

`<void>`

An empty element in a function indicating there are no arguments.

`<varargs>`

An empty element in a function indicating there are multiple arguments, without specifically listing them. This is generally represented with an ellipsis (...). For example `int max(...);`

`<funcsynopsisinfo>`

Not used in KDE documentation.

`<arg>`

Used inside `<cmdsynopsis>`. Since most KDE applications are GUI only, you won't see this very often. See the entry for `<cmdsynopsis>` for a full explanation and example.

`<group>`

Group

`<sbr>`

sbr

`<synopfragment>`

synopfragment

`<modifier>`

A modifier modifies a class, field, or method synopsis. Examples are the words `?public?`, `?private?` or `?virtual?`

`<fieldsynopsis>`

A field synopsis.

Chapter 15. Markup for programming

For formally marking up code examples or making a synopsis, you should study the Duck Book and the [Synopsis](#) chapter. The elements described below are mainly for marking up of pieces of source code that appear in the running text. Remember that KDE and KDE applications are written almost exclusively in C++, so our usage may differ in places from the examples in the Duck book, which may be describing other programming languages.

To Developers reading this, remember most of the authors who may be documenting your work are unfamiliar with source code, and many of them like it that way. Therefore, the explanations here are more concerned with how to tell things apart than what they are for, and may make you cringe.

To everyone reading this, this section is very much under construction so to speak. If you already need to use this markup, you can ask questions on the kde-docbook mailing list, which is the most likely place to get correct and up to date answers.

`<classname>`

Used to identify the name of a class in a programming language. In KDE Documentation, you won't see this much in the user documentation, except for those applications which contain an API reference chapter, and occasionally in others. You will find it used a lot in the KDevelop documentation.

For non-programmers, as we're almost exclusively discussing KDE applications written in C++ and using Qt?, classnames are fairly easy to distinguish: They start with a capital Q or K, and are usually one word only, in the form of `KApplication` or `QListBox`.

`<function>`, `<methodname>`

A function or ?subroutine?. In C++, a function generally looks something like this: `foo() ;`. The semi-colon may not always be present and there may or may not be content inside the braces.

If you see things that have the form `Kfoo : : bar ()` these are not just functions, but also methods, so you would use the `<methodname>` for these.

Constructors are methods where the parts before and after the `::` are the same, e.g. `Kfoo : : Kfoo ()`. Destructors look like Constructors, but have a `~` after the `::`: e.g. `Kfoo : : ~Kfoo ()`. The same things apply as with functions and methods: there may or may not be a `;` at the end, and there may or may not be content inside the braces of a constructor (there is never content for a destructor).

These are normally marked up as `<methodname>`, but if you need to make a synopsis of a method, there are specific elements available: `<constructorsynopsis>` and `<destructorsynopsis>`

To recap:

Function

```
foo( )
```

Methodname

```
Kfoo : : bar ( )
```

Constructor

`Kfoo() : Kfoo()` These are methods in ordinary text, but when making a synopsis, have a more specific tag to use.

Destructor

`Kfoo() : ~Kfoo()` These are methods in ordinary text, but when making a synopsis, have a more specific tag to use.

Sometimes you really can't tell the difference, especially when they are being mentioned in passing in the text. Also, programmers tend to shorten and make shortcuts when referring to snippets of source. If it's very unclear what something is, mark it up with `<function>` and ask the developer.

Tip

Asking a developer "What is foo?" will likely result in a two page explanation of a finer point of C++ programming, which, if you could understand it, you wouldn't have needed to ask the question in the first place. It saves everyone a lot of time and frustration if you word the question "Out of function, method, constructor and destructor, which is the best fit for foo?".

`<varname>`

The name of a variable.

`<returnvalue>`

The value returned by a function.

`<token>`

A token is a placeholder, something that is replaced by an actual value during processing. (I need to come up with a useful example for a token)

`<constant>`

A constant. In the snippet:

```
enum MyType { Red = 0, Green, Blue, Yellow };
```

Red, Green, Blue and Yellow should be marked up as `<constant>`

`<type>`

Used to classify a value. In the snippet:

```
enum MyType { Red = 0, Green, Blue, Yellow };
```

MyType is a `<type>`

`<programlisting>`

Use this to wrap any source code examples in your document. You don't need to use this for short snippets that are inline in the text, but you should use it for any examples longer than a line or two, or that are a separate block of text.

`<structname>`,`<structfield>`

Not used in KDE Documentation, primarily because they are rare in KDE source code, and are almost certainly never going to require marking up.

`<parameter>`

Parameters can be used for commandlines as well as for code samples.

`<classsynopsis>`

A class synopsis

```
DCOPStub {? not sure about what goes here ? enum Status(CallSucceeded, CallFailed);}
```

`<initializer>`

An initializer

`<exceptionname>`

An exception name

Chapter 16. Making Callouts

Callouts are difficult, so they have their own chapter. Use callouts when you want to refer from text to specific parts of an image, `programlisting`, or synopsis. Using callouts with graphics is currently unused, and is somewhat problematic, so they will not (yet) be described here.

`<calloutlist>`

A list element that contains the callouts themselves. That is, a list of the explanations that belong to the indicated areas in the item being explained.

`<callout arearefs=" ">`

The actual explanation or description of the called out area or line. The `arearefs` attribute should contain the id of the appropriate callout you are referring to.

`<programlistingco>` and `<screenco>`

Callouts applied to a programlisting or a screen element. Although they look more difficult than just embedding the callouts directly in the text, they really aren't too hard. The programlistingco contains one areaspec, and one programlisting. The screenco contains one areaspec and one screen element. The programlisting and screen elements are exactly as you would normally have.

```
<areaspec>
```

The areaspec contains a list of area elements, each of which describes one single callout.

```
<area coords="" id="" />
```

The area is another of the very few empty elements, so there is no `</area>`. The `id` attribute should contain a unique name for the item. The `coords` contains a pair of numbers which indicate first the line and then the column where the `co` should appear. The line and column refer to the position in relation to the container element, *not the entire document!*. That is, in a screenco, the line and column numbers refer to the line *within the screen element*.

Example 16.1. Marking up callouts with `<screenco>`.

```
<screenco>
  <areaspec>
    <area coords="2 65" id="currentdir" />
    <area coords="3 65" id="updir" />
    <area coords="4 75" id="hiddenfile" />
    <area coords="10 75" id="backupfile" />
    <area coords="13 70" id="hiddendir" />

  <screen>
total 864
drwx-----  8 vampyr  vampyr      4096 Oct  2 18:01 ./
drwxr-xr-x  13 root    root        4096 Oct  1 16:32 ../
-rw-----  1 vampyr  vampyr         32 Sep  2 14:21 .MCOP-random-seed
-rw-----  1 vampyr  vampyr         0 Sep  2 14:42 .Xauthority
-rw-r--r--  1 vampyr  vampyr      1899 Aug  6 19:32 .Xdefaults
-rw-----  1 vampyr  vampyr       261 Sep 29 22:59 .bash_history
-rw-r--r--  1 vampyr  vampyr       24 Aug  6 19:32 .bash_logout
-rw-r--r--  1 vampyr  vampyr      285 Aug  6 19:34 .bash_profile
-rw-r--r--  1 root    root        230 Aug  6 19:32 .bash_profile~
-rw-r--r--  1 vampyr  vampyr       559 Aug  6 19:32 .bashrc
-rw-r--r--  1 vampyr  vampyr     4044 Aug  6 19:32 .emacs
drwxr-xr-x  7 vampyr  vampyr     4096 Sep 29 17:31 .kde/
  </screen>
</screenco>
<calloutlist>
<callout arearefs="currentdir1"><para>The current directory.</para>
</callout>
<callout arearefs="updir1">
<para>One directory up in the tree.</para>
</callout>
<callout arearefs="hiddenfile1">
<para>A hidden file, indicated by the . beginning the name.</para>
</callout>
<callout arearefs="backupfile1">
<para>A backup or temporary file, indicated by the ~ ending the name.</para>
```

```

</callout>
<callout arearefs="hiddendir1">
<para>A hidden directory, which, like a hidden file, is indicated by the . at
the start of the name.</para>
</callout>
</calloutlist>

```

All this markup above, while it looks complicated is really quite simple if you study it closely. It would generate the following:

```

total 864
drwx-----  8 vampyr  vampyr      4096 Oct  2 18:01 ./
drwxr-xr-x  13 root    root        4096 Oct  1 16:32 ../
-rw-----  1 vampyr  vampyr         32 Sep  2 14:21 .MCOP-random-seed
-rw-----  1 vampyr  vampyr         0 Sep  2 14:42 .Xauthority
-rw-r--r--  1 vampyr  vampyr      1899 Aug  6 19:32 .Xdefaults
-rw-----  1 vampyr  vampyr       261 Sep 29 22:59 .bash_history
-rw-r--r--  1 vampyr  vampyr        24 Aug  6 19:32 .bash_logout
-rw-r--r--  1 vampyr  vampyr       285 Aug  6 19:34 .bash_profile
-rw-r--r--  1 root    root        230 Aug  6 19:32 .bash_profile~
-rw-r--r--  1 vampyr  vampyr       559 Aug  6 19:32 .bashrc
-rw-r--r--  1 vampyr  vampyr     4044 Aug  6 19:32 .emacs
drwxr-xr-x  7 vampyr  vampyr      4096 Sep 29 17:31 .kde/

```

- ❶ The current directory.
- ❷ One directory up in the tree.
- ❸ A hidden file, indicated by the . beginning the name.
- ❹ A backup or temporary file, indicated by the ~ ending the name.
- ❺ A hidden directory, which, like a hidden file, is indicated by the . at the start of the name.

```
<co>
```

Indicates where a callout is. For KDE HTML documentation, a small numbered graphic will be placed here, and also at the location of the explanation. These numbered graphics are links between the two places. It is entirely possible to embed the callout elements directly in the text you are describing, and this is perhaps the easiest way to do it. It isn't the most specific, but working out the line coordinates to use the more precise elements is difficult, so this way is acceptable for now.

Example 16.2. Marking up callouts by embedding directly in text

```

<screen>
drwxr-xr-x  3 vampyr  vampyr      4096 Aug  6 19:32 .kde2/
lrwxrwxrwx  1 vampyr  vampyr        15 Sep  3 19:46
.kdeinit-whiterabbit.magicians.org-:0 -> /tmp/.kinV4m2iI= <co id="symlink"/>
-rw-r--r--  1 vampyr  vampyr      2096 Aug  6 19:32 .kderc
-r-----  1 vampyr  vampyr        21 Sep  2 14:21 .kxmlrpcd
-rw-r--r--  1 vampyr  vampyr       185 Aug  6 19:32 .mailcap
-rw-----  1 vampyr  vampyr        31 Sep  2 14:21 .mcoprc
drwxr-xr-x  4 vampyr  vampyr      4096 Aug  6 19:32 .netscape/

```

```

-rw----- 1 vampyr vampyr 777947 Sep 2 14:42 .xsession-errors
drwxr-xr-x 5 vampyr vampyr 4096 Sep 2 14:42 Desktop/ <co id="dir"/>
drwx----- 2 vampyr vampyr 4096 Aug 6 19:32 tmp/
-rw-r--r-- 1 vampyr vampyr 3836 Oct 13 16:44 notes.txt <co id="file"/>
</screen>

```

```

<calloutlist>
<callout arearefs="symlink">
<para>A symbolic link, indicated by the ->, and showing the location it is
linked to.</para>
</callout>
<callout arearefs="dir">
<para>An ordinary directory.</para>
</callout>
<callout arearefs="file">
<para>An ordinary file.</para>
</callout>
</calloutlist>

```

Again it's really not as hard as it looks on first glance. This markup would generate the following:

```

drwxr-xr-x 3 vampyr vampyr 4096 Aug 6 19:32 .kde2/
lrwxrwxrwx 1 vampyr vampyr 15 Sep 3 19:46
.kdeinit-whiterabbit.magicians.org--0 -> /tmp/.kinV4m2iI= ❶ -rw-r--r-- 1 vampyr vampyr 2096 Aug 6 19:32 .kderc
-r----- 1 vampyr vampyr 21 Sep 2 14:21 .kxmlrpcd
-rw-r--r-- 1 vampyr vampyr 185 Aug 6 19:32 .mailcap
-rw----- 1 vampyr vampyr 31 Sep 2 14:21 .mcoprc
drwxr-xr-x 4 vampyr vampyr 4096 Aug 6 19:32 .netscape/
-rw----- 1 vampyr vampyr 777947 Sep 2 14:42 .xsession-errors
drwxr-xr-x 5 vampyr vampyr 4096 Sep 2 14:42 Desktop/ ❷ drwx----- 2 vampyr vampyr 4096 Aug 6 19:32 tmp/
-rw-r--r-- 1 vampyr vampyr 3836 Oct 13 16:44 notes.txt ❸

```

- ❶ A symbolic link, indicated by the ->, and showing the location it is linked to.
- ❷ An ordinary directory.
- ❸ An ordinary file.

<imageobjectco>

Currently unused in KDE Documentation.

<mediaobjectco>

Currently unused in KDE Documentation.

<areaset>

Currently unused in KDE Documentation. This and the above two elements will be used eventually (just as soon as I figure out how they work).

<graphicco>

Not to be used in KDE Documentation at all.

Chapter 17. References, indexes, and glossaries

Table of Contents

[Making a glossary](#)

[Making an Index](#)

[Other Reference Sections](#)

These elements are very underused in KDE Documentation up to this point, and we will probably make an effort to implement them more fully at some point. In the meantime, you may use them if you wish, so they are explained here.

Making a glossary

`<glossterm>`

Use this inline to identify words in the text that are explained further in a `<glossary>` or `<glosslist>`. When it's placed inside a `<glossentry>` it contains the term that glossary entry is defining (see the example below to see this in action.)

`<glossary>`

Put this where you have the glossary appearing. This is usually at the end of the document, perhaps last before the credits section, or before an index. A glossary will become a separate section in the book.

`<glosslist>`

Use this if the glossary is fairly short and simple. It can appear anywhere a normal list could appear. For KDE Documentation, a proper glossary is preferred, so keep use of `<glosslist>` to a minimum, where your glossary would only contain a small handful of entries. Use your own judgement which is most appropriate. You might use a glosslist for example, to explain a list of terms which only appear in one section, but are very important to understanding that section and occur several times there, so you want the explanations to appear close to the text.

`<glossdiv>`

Divides a glossary into several smaller sections. A good use of this in a very large glossary could be to break it up into separate sections for each letter in the alphabet.

`<glossentry id="">`

Contains the actual entries in the glossary or glosslist, where you explain the terms you have marked up with `glossterm` in the text. You should give these an `id`, so they can be linked to from the text, and crossreferenced between glossary entries.

A glossentry always contains one `<glossterm>`. It also contains one `<glossdef>`, or one `<glosssee>`, or a `<glossdef>` and a `<glossseealso>`.

Tip

I would suggest a consistent naming scheme, so glossary entries are easy to reference without having to go look them up all the time. For example, I use the form `id="gloss-word"`, where *word* is the term that is being explained.

```
<glossdef>
```

Contains the actual definitions of the terms

```
<glosssee otherterm=" ">
```

You can use this to save duplicating entries in the glossary. Instead of a `<glossdef>` you can put `<glosssee>` with the id of another `<glossentry>`.

```
<glossseealso otherterm=" ">
```

This is very similar to `<glosssee>`, but instead of replacing the `<glossdef>` it is in addition to it.

If you compare a glossary entry to a variable list entry, you'll see the structure is quite similar, with a `glossterm` taking the place of the term, and a `glossdef` taking the place of the `listitem`. Since variable lists get heavy use in KDE Documents, it shouldn't take you long to pick up how to do a glossary.

Example 17.1. How to markup a glossary

Say you have in the text of the document the following sentence:

KWord is a graphical, wysiwyg word processor, and is part of KOffice.

You want to have the words KWord and koffice in the index, and KWord, wysiwyg, ?word processor? and KOffice explained in a glossary.

Many of these terms also need to be marked up with other tags, such as application, and acronym.

The eventual markup would look like this:

```
<para><glossterm linkend="gloss-kword">KWord</glossterm>
<indexterm><primary>KWord</primary></indexterm> is a
graphical <glossterm linkend
="gloss-wysiwyg"><acronym>WYSIWYG</acronym></glossterm>
<glossterm linkend="gloss-word-processor">word
processor</glossterm>, and is part of <glossterm
linkend="gloss-koffice">KOffice</glossterm>.
<indexterm><primary>KOffice</primary></indexterm></para>
```

The next part is shown here as a `<glosslist>`, and if there were really only this many entries in it, that could be entirely appropriate. In reality, if you are going to make a glossary, it would have many more entries and so would warrant its own `<glossary>` section. The syntax inside `<glossary>` and `<glosslist>` are otherwise the same.

```
<glosslist>
<glossentry id="gloss-kword">
<glossterm>KWord</glossterm>
<glossdef><para>The name of the KDE word
processor</para></glossdef>
</glossentry>

<glossentry id="gloss-koffice">
<glossterm>KOffice</glossterm>
<glossdef><para>A collection of office productivity tools, designed
by and for <acronym>KDE</acronym>, including presentation software,
a word processor, a spreadsheet, a <acronym>PIM</acronym>, and a
vector illustration application.</para></glossdef>
</glossentry>

<glossentry id="gloss-word-processor">
<glossterm>word processor</glossterm>
<glossdef><para>An application for handling text, typically more
concerned with formatting visually than a plain text
editor.</para></glossdef>
</glossentry>

<glossentry id="gloss-wysiwyg">
<glossterm>WYSIWYG</glossterm>
<glossdef><para>Stands for <quote>What You See Is What You
Get</quote>, indicating that you can visually format the presentation of
your data onscreen, and when you print the document, it will look exactly as you
see on the screen.</para></glossdef>
</glossentry>
</glosslist>
```

And the result of all this would be as follows:

KWord is a graphical *WYSIWYG word processor*, and is part of *KOffice*..

KWord

The name of the KDE word processor

KOffice

A collection of office productivity tools, designed by and for KDE, including presentation software, a word processor, a spreadsheet, a PIM, and a vector illustration application.

word processor

An application for handling text, typically more concerned with formatting visually than a plain text editor.

WYSIWYG

Stands for 'What You See Is What You Get', indicating that you can visually format the presentation of your data onscreen, and when you print the document, it will look exactly as you see on the screen.

Making an Index

For KDE Documentation, indexes will in the future be generated automatically, so many of these elements are not to be used directly when authoring. At this stage, indexes are not generated, but if you want to you can mark up words that should be indexed with the `<indexterm>` element, to save work for later.

`<indexterm>`

Use this to note places in the main text of the document that should have an entry in the index. Don't over use it - not every single occurrence of a word needs to be noted in the index, but every occurrence where that term is significant should be.

`indexterm` should contain a `<primary>`, which contains the text that the entry will appear under in the index.

Place the `indexterm` directly before the word you want to index, and place the word itself inside the primary element. If the word should also be listed under a secondary heading, place that term inside a secondary element.

Example 17.2. Index

Say the document contains the following sentence:

KWord is a graphical, wysiwyg word processor, and is part of KOffice.

You want KWord to have an index entry of it's own, and to also be noted under KOffice in the index.

```
<para><application>KWord</application>
<indexterm><primary>KWord</primary><secondary>KOffice</secondary></indexterm>
is a graphical, <acronym>WYSIWYG</acronym> word processor, and is part of
KOffice.</para>
```

The fact that an index entry exists is not normally indicated by a change in appearance.

If you think it should also be added under a third heading in the index, you can use tertiary to indicate this. Most terms you would find in KDE Documentation will only need a primary index heading, so use the others sparingly, if at all.

`<tertiary>`

tertiary

<seealso>

seealso

The following elements are used to create the actual index, but they are automatically generated, if required. You should not use them when authoring documents.

- <index>
- <indexdiv>
- <indexentry>
- <primaryie>
- <secondaryie>
- <see>
- <seealsoie>
- <seeie>
- <tertiaryie>

Other Reference Sections

<firstterm>

Mark up the first occurrence of a technically significant term with this element. If you are creating a glossary or an index, the first occurrence of a term will probably also warrant being an entry in one or both.

<refsynopsisdivinfo>

refsynopsisdivinfo

<refnamediv>

refnamediv

<refclass>

refclass

<refmeta>

refmeta

<refsect1>,<refsect2> and <refsect3>

refsect1, refsect2 and refsect3

<refmiscinfo>

refmiscinfo

<refsect1info>,<refsect2info> and <refsect3info>

refsect1info, refsect2info and refsect3info

<refdescriptor>

refdescriptor

<setindex>

Not Used in KDE Documentation

<refpurpose>

refpurpose

<reference>

reference

<refentrytitle>

refentrytitle

<refname>

refname

<refentry>

refentry

<refsynopsisdiv>

refsynopsisdiv

Chapter 18. Tags we do not use

These are tags that are available for DocBook XML, but we have decided they will not (at this time) be used for KDE Documentation. They are included here for completeness, and so nobody can say "I didn't know I wasn't supposed to use that!"

They fall into two categories: Tags we have definitely decided to not use, in which case we have made a decision to use another tag instead, and tags that are just irrelevant to the documentation we are doing, which you hopefully will never want. Should we write new documentation that can sensibly be marked up with any of these elements, this list will be revised.

If you think you have a use for one of these elements, please, check with the DocBook team first, and be prepared to justify your case.

- `<ackno>`
- `<alt>`
- `<appendixinfo>`
- `<arthead>`
- `<article>`
- `<articleinfo>`
- `<artpagenums>`
- `<audiodata>`
- `<audioobject>`
- `<authorblurb>`
- `<authorinitials>`
- `<beginpage>`
- `<bibliodiv>`
- `<biblioentry>`
- `<bibliography>`
- `<bibliographyinfo>`
- `<bibliomisc>`

- <bibliomixed>
- <bibliomset>
- <biblioset>
- <bookbiblio>
- <bridgehead>
- <chapterinfo>
- <citation>
- <citerefentry>
- <citetitle>
- <city>
- <collab>
- <collabname>
- <colophon>
- <colspect>
- <comment>
- <confdates>
- <confgroup>
- <confnum>
- <confsponsor>
- <conftitle>
- <contractnum>
- <contractsponsor>
- <corpname>
- <country>

- <database>
- <dedication>
- <docinfo>
- <edition>
- <editor>
- <entrytbl>
- <fax>
- <figure>
- <formalpara>
- <sgmltag>
- <graphic>
- <highlights>
- <honorific>
- <indexinfo>
- <informalequation>
- <informalfigure>
- <inlineequation>
- <inlinegraphic>
- <interfacedefinition>
- <interfacename>
- <invpartnumber>
- <isbn>
- <issn>
- <issuenum>

- <itermset>
- <jobtitle>
- <lineage>
- <lot>
- <lotentry>
- <manvolnum>
- <medialabel>
- <modespec>
- <msg>
- <msgaud>
- <msgentry>
- <msgexplan>
- <msginfo>
- <msglevel>
- <msgmain>
- <msgorig>
- <msgrel>
- <msgset>
- <msgsub>
- <objectinfo>
- <olink>
- <orgdiv>
- <orgname>
- <otheraddr>

- <pagenums>
- <part>
- <partintro>
- <phone>
- <pob>
- <postcode>
- <preface>
- <prefaceinfo>
- <printhistory>
- <productname>
- <productnumber>
- <property>
- <pubdate>
- <publisher>
- <publishername>
- <pubsnumber>
- <qandadiv>
- <refentryinfo>
- <referenceinfo>
- <remark>
- <revdescription>
- <revhistory>
- <revision>
- <revnumber>

- <revremark>
- <secondary>
- <section>
- <sectioninfo>
- <seriesinfo>
- <seriesvolnums>
- <set>
- <setindexinfo>
- <setinfo>
- <shortaffil>
- <sidebar>
- <sidebarinfo>
- <simpara>
- <simplemsgentry>
- <simplesect>
- <spanspec>
- <state>
- <street>
- <subject>
- <subjectset>
- <subjectterm>
- <subtitle>
- <systemitem>
- <titleabbrev>

- <toc>
- <tocback>
- <tocchap>
- <tocentry>
- <tocfront>
- <toclevel1>
- <toclevel2>
- <toclevel3>
- <toclevel4>
- <toclevel5>
- <tocpart>
- <videodata>
- <videoobject>
- <volumenum>
- <wordasword>

Chapter 19. Alphabetical List of all elements

This is a list of all the markup elements contained in DocBook XML 4.1.2. Choose the element you are interested in to go directly to the section of this document which describes it.

Note

We don't use all these elements in KDE Documentation - they are here for completeness. Elements we don't use are listed in [Chapter 18, *Tags we do not use*](#).

- [<authorinitials>](#)
- [<beginpage>](#)
- [<bibliodiv>](#)
- [<biblioentry>](#)

- <bibliographyinfo>
- <bibliomset>
- <bibliomisc>
- <bibliomixed>
- <biblioset>
- <bibliography>
- <blockquote>
- <book>
- <bookbiblio>
- <bookinfo>
- <bridgehead>
- <co>
- <callout>
- <calloutlist>
- <caption>
- <caution>
- <chapter>
- <chapterinfo>
- <citation>
- <citerefentry>
- <citetitle>
- <city>
- <classname>
- <classsynopsis>

- <classsynopsisinfo>
- <cmdsynopsis>
- <colspec>
- <collab>
- <collabname>
- <colophon>
- <command>
- <comment>
- <computeroutput>
- <confdates>
- <confgroup>
- <confnum>
- <confsponsor>
- <conftitle>
- <constant>
- <constructorsynopsis>
- <contractnum>
- <contractspnosor>
- <contrib>
- <copyright>
- <corpauthor>
- <corpname>
- <country>
- <database>

- <date>
- <dedication>
- <destructorsynopsis>
- <docinfo>
- <edition>
- <editor>
- <email>
- <emphasis>
- <envar>
- <entry>
- <entrytbl>
- <epigraph>
- <equation>
- <errorcode>
- <errorname>
- <errortype>
- <example>
- <exceptionname>
- <fax>
- <figure>
- <fieldsynopsis>
- <filename>
- <firstterm>
- <footnote>

- <footnoteref>
- <foreignphrase>
- <formalpara>
- <funcdef>
- <funcparams>
- <funcprototype>
- <funcsynopsis>
- <funcsynopsisinfo>
- <function>
- <gUIButton>
- <guiicon>
- <guilabel>
- <guimenu>
- <guimenuitem>
- <guisubmenu>
- <glossdef>
- <glossdiv>
- <glossentry>
- <glosslist>
- <glosssee>
- <glossseealso>
- <glossterm>
- <glossary>
- <glossaryinfo>

- <graphic>
- <graphicco>
- <group>
- <hardware>
- <highlights>
- <holder>
- <honorific>
- <isbn>
- <issn>
- <itermset>
- <imagedata>
- <imageobject>
- <imabeobjectco>
- <important>
- <index>
- <indexdiv>
- <indexentry>
- <indexinfo>
- <indexterm>
- <informalequation>
- <informalexample>
- <informalfigure>
- <informaltable>
- <initializer>

- <inlineequation>
- <inlinegraphic>
- <inlinemediaobject>
- <interface>
- <interfacedefinition>
- <interfacename>
- <invpartnumber>
- <issuenum>
- <itemizedlist>
- <jobtitle>
- <keycap>
- <keycode>
- <keycombo>
- <keysym>
- <keyword>
- <keywordset>
- <legalnotice>
- <lineannotation>
- <lineage>
- <link>
- <listitem>
- <literal>
- <literallayout>
- <lot>

- <lotentry>
- <manvolnum>
- <markup>
- <medialabel>
- <mediaobject>
- <mediaobjectco>
- <member>
- <menuchoice>
- <methodname>
- <methodparam>
- <methodsynopsis>
- <modespec>
- <modifier>
- <mousebutton>
- <msg>
- <nmsgaud>
- <msgentry>
- <msgexplan>
- <msginfo>
- <msglevel>
- <msgmain>
- <msgorig>
- <msgrel>
- <msgset>

- <msgsub>
- <msgtext>
- <note>
- <olink>
- <objectinfo>
- <option>
- <optional>
- <orderedlist>
- <orgdiv>
- <orgname>
- <otheraddr>
- <othercredit>
- <othername>
- <pob>
- <pagenums>
- <para>
- <paramdef>
- <parameter>
- <part>
- <partintro>
- <phone>
- <phrase>
- <postcode>
- <preface>

- <prefaceinfo>
- <primary>
- <primaryie>
- <printhistory>
- <procedure>
- <productname>
- <productnumber>
- <programlistingco>
- <prompt>
- <property>
- <pubdate>
- <publisher>
- <publishername>
- <pubsnumber>
- <qandadiv>
- <qandaentry>
- <qandaset>
- <question>
- <quote>
- <refclass>
- <refdescriptor>
- <refentry>
- <refentryinfo>
- <refentrytitle>

- <referenceinfo>
- <refmeta>
- <refmiscinfo>
- <refname>
- <refnamediv>
- <refpurpose>
- <refsect1>
- <refsect1info>
- <refsect2>
- <refsect2info>
- <refsect3info>
- <refsect3info>
- <refsynopsisdiv>
- <refsynopsisdivinfo>
- <reference>
- <releaseinfo>
- <remark>
- <replaceable>
- <returnvalue>
- <revdescription>
- <revhistory>
- <revnumber>
- <revremark>
- <revision>

- <row>
- <sbr>
- <sgmltag>
- <screen>
- <screenco>
- <screeninfo>
- <screenshot>
- <secondary>
- <secondaryie>
- <sect1>
- <sect1info>
- <sect2>
- <sect2info>
- <sect3>
- <sect3info>
- <sect4>
- <sect4info>
- <sect5>
- <sect5info>
- <section>
- <sectioninfo>
- <see>
- <seealso>
- <seealsoie>

- <seeie>
- <seg>
- <seglistitem>
- <segmentedlist>
- <seriesinfo>
- <seriesvolnums>
- <set>
- <setindex>
- <setindexinfo>
- <setinfo>
- <shortaffil>
- <shortcut>
- <sidebar>
- <sidebarinfo>
- <simpara>
- <simplelist>
- <simplemsgentry>
- <simplesect>
- <spanspec>
- <state>
- <step>
- <street>
- <structfield>
- <structname>

- <substeps>
- <subject>
- <subjectset>
- <subjectterm>
- <subscript>
- <subtitle>
- <superscript>
- <surname>
- <symbol>
- <synopfragment>
- <synopsis>
- <systemitem>
- <tbody>
- <tfoot>
- <tgroup>
- <thead>
- <table>
- <term>
- <tertiary>
- <tertiaryie>
- <textobject>
- <tip>
- <title>
- <titleabbrev>

- <toc>
- <tocback>
- <tocchap>
- <tocentry>
- <tocfront>
- <toclevel1>
- <toclevel2>
- <toclevel3>
- <toclevel4>
- <toclevel5>
- <tocpart>
- <token>
- <trademark>
- <type>
- <ulink>
- <userinput>
- <varargs>
- <varlistentry>
- <varname>
- <variablelist>
- <videodata>
- <videoobject>
- <void>
- <volumenum>

- <warning>
- <wordasword>
- <xref>
- <year>

Chapter 20. Credits and License

Document copyright 2000, 2001 Lauri Watts <lauri@kde.org>

This reference was written with substantial help and input from the following people who definitely deserve credit:

- Frederik Fouvry
- Eric Bischoff
- Michael McBride
- Lee Wee Tiong
- Philip Rodrigues
- Eyal Lotem <GNUPeaker@yahoo.com>
- Malte Starostik <malte.starostik@t-online.de>
- Antonio Larossa Jiminez

This documentation is licensed under the terms of the [GNU Free Documentation License](#).

Appendix A. Entities

TODO